Uma Perspectiva Evolutiva dos Sistemas Robóticos

Por
Eduardo José Solteiro Pires

Tese submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
DOUTOR,
de acordo com o disposto no
Decreto-lei 216/92 de 13 de Outubro

Vila Real, 6 de Junho de 2005

Orientador Científico:

Doutor José António Tenreiro Machado

Professor Coordenador com Agregação
Departamento de Engenharia Electrótecnica do
Instituto Superior de Engenharia do Porto

Co-orientador Científico:

Doutor José Paulo Barroso de Moura Oliveira

Professor Auxiliar do
Departamento de Engenharias
Universidade de Trás-os-Montes e Alto Douro

Este trabalho foi co-financiado pelo FSE
através da medida 5, acção 5.3, do PRODEP III

A satisfação está no esforço feito para alcançar o objectivo,
e não em tê-lo alcançado.

Gandhi

Aos meus pais

BENEDITA PIRES

RUI PIRES

Agradecimentos

A realização deste trabalho apenas foi possível devido ao empenhamento, dedicação e apoio de muitas pessoas bem como o apoio material e financeiro de certas instituições. Algumas delas merecem uma menção especial.

O meu orientador Prof. J. A. Tenreiro Machado e co-orientador Prof. J. P. B. de Moura Oliveira, não só pelo apoio, incentivo e disponibilidade constantemente manifestada, mas também pela confiança em mim depositada guiando-me com o seu saber, experiência, competência e rigor profissional, durante a realização dos trabalhos de investigação, sempre na direcção correcta. Sem eles, a realização desta dissertação não teria sido possível.

Alguns colegas da Universidade de Trás-os-Montes e Alto Douro (UTAD) pela ajuda, apoio e motivação.

Alguns colegas do Grupo de Robótica e Sistemas Inteligentes (GRIS) pelo apoio e disponibilidade sempre demonstrada.

A Universidade de Trás-os-Montes e Alto Douro e ao Departamento de Engenharias por todas as facilidades concedidas na realização deste trabalho.

O Programa de Desenvolvimento Educativo para Portugal (PRODEP) pela concessão de uma bolsa que permitiu a dispensa de serviço lectivo e apoio material durante

o período de realização deste trabalho.

O Departamento de Engenharias da Universidade de Trás-os-Montes e Alto Douro, a Fundação Luso-Americana para o Desenvolvimento (FLAD) e a Fundação para a Ciência e a Tecnologia (FCT) pelos apoios financeiros concedidos para participar em algumas conferências.

A todos, incluindo os não mencionados e em especial a todos os meus familiares e amigos, que directa ou indirectamente contribuíram para tornar este trabalho realidade.

Resumo

Esta tese propõe um conjunto de técnicas baseadas em algoritmos genéticos, para optimizar a síntese e o planeamento de trajectórias de manipuladores robóticos. Nas experiências consideradas a optimização é baseada no deslocamento angular e cartesiano, na oscilação residual angular e cartesiana, na energia e no tempo requerido pelo manipulador. O problema é abordado numa perspectiva uni e multi objectivo. A dissertação inclui os resultados da investigação realizada utilizando manipuladores robóticos planares, de dois, três e quatro graus de liberdade.

Neste trabalho é também proposta uma técnica de optimização multi-objectivo de forma a aumentar o desempenho de alguns algoritmos propostos, bem como alguns índices de desempenho com a finalidade de medir a eficiência dos algoritmos genéticos multi-objectivo.

A dinâmica de um algoritmo genético é estudada realizando estudos na sensibilidade de um parâmetro do algoritmo, nomeadamente a probabilidade de mutação. Adicionalmente é investigada a importância da função de aptidão na dinâmica do algoritmo. A evolução dinâmica do sistema foi aproximada por funções de ordem fraccionária. Neste âmbito, foi também analisada a dinâmica de um planeador de trajectórias robóticas.

Para além do trabalho proposto, esta dissertação fornece uma visão global dos algoritmos genéticos padrão e de algoritmos genéticos multi-objectivo. Inclui também a síntese de um conjunto de aplicações robóticas baseados em algoritmos genéticos.

Abstract

This dissertation proposes some techniques based on genetic algorithms, in order to optimize the structure and trajectory planning of robotic manipulators. In the experiments, several indices are considered to qualify the optimization: joint/arm traveling distance, joint/arm ripple in time evolution and energy. The optimization is resolved in two perspective: uni and multi objective. The work includes the investigation results using planar robotic manipulators with two, three and four degrees of freedom.

A new multi-objective technique is proposed in order to increase the performance of some algorithms proposed. In addition, some metrics indices are also proposed to measures the multi-objective genetic algorithms performance.

The dynamic of a genetic algorithm system is studied during its evolution. To investigate the phenomena involved in the GA population evolution, the mutation probability is exposed to excitation perturbations and the corresponding fitness variations are evaluated. Four fitness functions are used to study its influence in the genetic algorithms. The transfer function of the system system is studied revealing a fractional-order dynamic evolution. Additionally, a genetic trajectory planner is also investigated.

The work makes a global review of the basic aspects of genetic algorithms, multi--objective algorithms and includes several robotic approaches using evolutionary algorithms.

Índice

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Lista de símbolos

Lista de acrónimos

AE **A**lgoritmo **E**volutivo 10
AEMO **A**lgoritmo **E**volutivo **M**ulti-**O**bjectivo 54
AG **A**lgoritmo **G**enético 11
AGH **A**lgoritmo **G**enético **H**ierárquico 126
C-NSGA-II *C**lustered** N**on-dominated** S**orting** G**enetic** A**lgorithm* 73
CE **C**omputação **E**volutiva 10
CEIA **C**ritério do **E**rro do **I**ntegral **A**bsoluto 115
CEIAT **C**ritério do **E**rro do **I**ntegral **A**bsoluto multiplicado pelo **T**empo 115
CEIQ **C**ritério do **E**rro do **I**ntegral **Q**uadratico 115
CEIQ **C**ritério do **E**rro do **I**ntegral **Q**uadratico multiplicado pelo **T**empo 115
CF **C**alculo **F**raccional 251
CX Cruzamento Cíclico 34
CSG *C**onstructive** S**olid** G**eometry* 89
DIFs **D**erivadas e **I**ntegrais **F**raccionários 252
DPGA *D**istance-based** P**areto** G**enetic** A**lgorithm* 69
EE **E**stratégia de **E**volução 11
ϵ-MOEA *ϵ-M**ulti-**O**bjective** E**volutionary** A**lgorithm* 74
FA **F**actor de **A**grupamento 50
gdl **g**raus **d**e **l**iberdade 128

HLGA *Weight-Based Genetic Algorithm* 58
MDG Índice do **G**rafo de **D**istâncias **M**ínimas 180
MOGA ***M**ulti-**O**bjective **G**enetic **A**lgorithm* 60
NPGA ***N**iched **P**areto **G**enetic **A**lgorithm* 63
NSGA ***N**on-dominated **S**orting **G**enetic **A**lgorithm* 62
NSGA–II ***N**on-dominated **S**orting **G**enetic **A**lgorithm–II* 66
OPMO **O**ptimização de **P**roblemas **M**ulti-**O**bjectivo 54
OX Cruzamento Ordenado 34
PAES ***P**areto-**A**rchived **E**volution **S**trategy* 72
PBIL Aprendizagem Incremental Baseada na População 241
PMO **P**roblema **M**ulti-**O**bjectivo 54
PMX Cruzamento parcialmente semelhante 34
PE **P**rogramação **E**volutiva 11
PG **P**rogramação **G**enética 11
SBX Cruzamento Binário simulado 40
SA ***S**imulated **A**nnealing* 240
SC **S**istemas de **C**lassificação 11
SP Índice baseado na distância, ***S**pacing **I**ndex* 77
SPEA ***S**trenght **P**areto **E**volutionary **A**lgorithm* 71
TF **T**ransformada de **F**ourier 208
UD ***U**niform **D**istribution* 78
WBGA ***W**eight-**B**ased **G**enetic **A**lgorithm* 58
VEGA ***V**ector **E**valuated **G**enetic **A**lgorithm* 58

1 Introdução

1.1 Introdução

Este capítulo pretende enquadrar o tema da dissertação, ou seja, o uso de algoritmos evolutivos nos sistemas robóticos, de modo a fornecer uma visão breve e geral da tese. Está estruturado da seguinte forma: na secção 1.2 é feita uma breve introdução ao tema da tese. De seguida, na secção 1.3 apresenta-se a motivação e os objectivos da dissertação. Na secção 1.4, são referenciadas as contribuições científicas resultantes deste trabalho. Por último, na secção 1.5 é descrita a estrutura da tese.

1.2 Tema: sistemas robóticos e algoritmos evolutivos

O tema desta dissertação consiste no desenvolvimento de aplicações robóticas utilizando algoritmos evolutivos, incidindo essencialmente no planeamento de trajectórias para manipuladores robóticos.

Os sistemas robóticos são estruturas mecânicas que pretendem simular meios de manipulação e locomoção existentes nos seres vivos. Estes sistemas mecânicos exibem fenómenos cinemáticos e dinâmicos de natureza complexa o que torna difícil

o seu estudo e controlo usando técnicas clássicas, levando a que a utilização deste tipo de sistemas robóticos esteja confinada a situações de baixo desempenho. Alguns algoritmos robóticos desenvolvidos recentemente encontram-se ainda numa fase embrionária, razão pela qual é necessária uma investigação mais profunda dos vários fenómenos em jogo. Assim, aspectos tais como:

- a eficiência da estrutura,
- a robustez e o desempenho dos algoritmos,
- o peso computacional e os métodos para a sua melhoria,
- o desenvolvimento de técnicas baseadas nos fenómenos presentes no sistema biológico "equivalente",

estão ainda, em fase de desenvolvimento. A consideração destes tópicos deverá lançar luz sobre algumas das lacunas mencionadas e, assim, proporcionar o desenvolvimento do conhecimento científico que visa ampliar o campo de aplicação dos sistemas robóticos.

Nas últimas décadas têm sido propostos algoritmos inspirados em certos processos biológicos dando origem à área da inteligência computacional que engloba a computação evolutiva. Esta área está em forte expansão pelo que a sua sistematização não foi ainda totalmente realizada. Por outro lado, a aplicação destes conceitos no campo da robótica está ainda numa fase em que existe um vasto campo de investigação e desenvolvimento de novos algoritmos.

A evolução é o princípio de unificação principal da biologia moderna. A evolução Darwineana clássica juntamente com a selecção de Weismann e a genética de Mendel formam a base para a teoria da evolução actual, ou, por outras palavras, a evolução é o resultado de processos estocásticos interactivos (reprodução, mutação, cruzamento e selecção) sobre populações ao longo de gerações. Todavia, a teoria

da evolução estende-se para além dos sistemas biológicos, quando utilizada por máquinas computadorizadas com o fim de resolver problemas de aprendizagem e optimização. A computação evolutiva abrange uma série de algoritmos, nomeadamente os algoritmos genéticos que são os mais utilizados.

Os princípios básicos dos algoritmos genéticos (AGs) foram propostos inicialmente por Holland [1]. Os AGs são inspirados num mecanismo de selecção natural ou seja, num processo biológico no qual somente os indivíduos mais fortes sobrevivem no seio de um ambiente altamente competitivo. Neste âmbito, os AGs usam directamente a analogia de muitos sistemas naturais.

1.3 Motivação e objectivos

A complexidade da cinemática e da dinâmica cria dificuldades ao planeamento de trajectórias para robôs, que não são resolvidas, de forma eficiente com as técnicas de controlo clássicas. Por outro lado, os algoritmos evolutivos tornaram-se bastante populares pela sua simplicidade e robustez podendo ser aplicados a um vasto número de aplicações. No entanto, os estudos existentes, usando algoritmos genéticos, sobre a síntese e planeamento de trajectórias para manipuladores robóticos são abordados de forma uni-objectivo onde os diversos objectivos são agregados de uma forma "cega". Além disso, as técnicas que promovem a diversidade de soluções ou são pouco eficientes ou têm um peso computacional considerável.

A motivação deste trabalho resultou da necessidade em desenvolver metodologias robustas para os sistemas robóticos utilizando abordagens baseadas na computação evolutiva. Assim, os principais objectivos deste trabalho são enumerados nos pontos seguintes:

- Desenvolver métodos que permitam resolver de forma robusta a síntese e o planeamento de manipuladores robóticos, utilizando:

1. algoritmos genéticos uni-objectivo,
2. algoritmos genéticos multi-objectivo,

nomeadamente nos seguintes problemas:

1. optimização de trajectórias para manipuladores redundantes,
2. optimização de trajectórias em tempo real,
3. síntese de manipuladores robóticos.

- Desenvolver métodos que permitam melhorar o desempenho dos algoritmos genéticos, em particular no que diz respeito aos problemas:
  1. algoritmo de promoção da diversidade genética das soluções, garantido a melhor uniformidade e distribuição,
  2. técnicas de avaliação da convergência dos algoritmos em termos da qualidade das soluções pertencentes à frente óptima de Pareto.
- Analisar a dinâmica evolutiva dos algoritmos genéticos utilizando como ferramenta o cálculo fraccionário.

1.4 Contribuições científicas

A investigação desenvolvida e apresentada nesta tese teve como finalidade atingir os objectivos enunciados na secção anterior. Este trabalho procura dar um contributo para o avanço do estado da arte nas áreas científicas em questão. As principais contribuições inovadoras realizadas são:

- O desenvolvimento de um planeador de trajectórias genético para manipuladores redundantes, resultando nas publicações [2–4];
- O desenvolvimento de uma aplicação em tempo real tendo em atenção dois objectivos principais: a duração da trajectória e a energia requerida pelo manipulador, dando origem às publicações [5–7];

- A síntese de manipuladores robóticos tendo em atenção a execução de diversas tarefas [8, 9] e sendo construído do ponto de vista multi-objectivo [10];
- A resolução do planeamento de trajectórias para manipuladores redundantes do ponto de vista multi-objectivo [11–13];
- Concepção de um algoritmo para promover a diversidade de soluções num algoritmo genético multi-objectivo [14];
- Análise dinâmica de planeadores de trajectórias para manipuladores robóticos utilizando como ferramenta o cálculo fraccionário [15–17].

1.5 Estrutura e organização da tese

Esta tese encontra-se organizada em duas partes principais:

- Na primeira parte, faz-se uma revisão ao estado da arte dando relevância aos conceitos fundamentais que serviram de base no desenvolvimento das aplicações abordadas neste trabalho. Esta parte é composta pelos capítulos 2-4;
- Na segunda parte, são discutidos alguns problemas na robótica e são apresentadas as soluções propostas e resultados que comprovam a sua eficácia. Para obter melhores resultados foram realizadas diferentes abordagens inovadoras na resolução dos problemas multi-objectivo e no estudo da dinâmica dos algoritmos genéticos. Esta parte é formadas pelos capítulos 5-9.

Para além deste capítulo introdutório (1), o conteúdo de cada um dos restantes nove capítulos em que a tese se encontra dividida é apresentado em seguida de uma forma mais detalhada:

No capítulo 2, *algoritmos evolutivos*, é feito o enquadramento do algoritmos genéticos dentro dos algoritmos de pesquisa. De seguida, são apresentados os conceitos

fundamentais dos algoritmos genéticos. Por último são apresentadas as principais técnicas que promovem a diversidade da população.

No capítulo 3, *algoritmos evolutivos multi-objectivo*, são descritos os aspectos principais dos problemas multi-objectivo. São também apresentados alguns algoritmos genéticos bastante utilizados na resolução de problemas multi-objectivo. Na parte final do capítulo são introduzidas algumas medidas para estudar o desempenho dos métodos propostos.

No capítulo 4, *planeamento de trajectórias e aplicações robóticas usando algoritmos genéticos*, na primeira parte é realizada uma introdução ao planeamento de trajectória para manipuladores robóticos. Na segunda parte do capítulo apresenta-se uma síntese de diversas aplicações robóticas usando algoritmos evolutivos.

No capítulo 5, *optimização de trajectórias em tempo real para um manipulador 2R*, é proposto um algoritmo com vista à geração de trajectórias para um manipulador $2R$, em tempo real. O algoritmo considera dois objectivos principais: a duração da trajectória e a energia gasta pelo manipulador para a efectuar.

No capítulo 6, *optimização de estruturas robóticas*, é desenvolvido um algoritmo genético para gerar a estrutura robótica de um manipulador que melhor se adeqúe à execução de determinadas trajectórias. A optimização da estrutura é realizada seja do ponto de vista mono-objectivo, usando uma função agregada, seja do ponto de vista multi-objectivo.

No capítulo 7, *optimização multi-objectivo de trajectórias robóticas*, é apresentada uma aplicação que optimiza as trajectórias de manipuladores robóticos considerando simultaneamente vários objectivos.

No capítulo 8, *algoritmo de selecção multi-objectivo MaxiMin*, é desenvolvido um método, que pode ser implementado na maioria dos algoritmos genéticos multi-objectivo existentes, e que promove a diversidade das soluções. Neste capítulo são

também propostas técnicas que permitem medir o desempenho dos algoritmos referidos ao nível da diversidade e da aproximação das soluções à frente óptima de Pareto.

No capítulo 9, *dinâmica dos algoritmos genéticos*, é investigada a dinâmica dos algoritmos genéticos. Neste capítulo é realizada a modelação e a identificação dos algoritmos genéticos recorrendo ao cálculo fraccionário. O capítulo é dividido em duas partes: a primeira estuda a dinâmica de um algoritmo genético simples, na segunda é analisada a dinâmica de um planeador genético de trajectórias.

No capítulo 10, *conclusões*, é feita uma breve reflexão relativa ao trabalho desenvolvido na presente dissertação, sendo enumeradas algumas perspectivas para desenvolvimento futuro.

Para além dos capítulos referidos, a tese é ainda composta por três apêndices que contêm conceitos utilizados ao longo do trabalho. Estes apêndices são descritos de seguida.

No apêndice A são descritas sucintamente algumas técnicas de pesquisa.

No apêndice B é descrito o método de Mann-Whitney utilizado na tese.

Finalmente, no apêndice C apresenta-se uma breve introdução ao cálculo fraccionário.

2 Algoritmos evolutivos

2.1 Introdução

A utilização de princípios biológicos como fonte de inspiração levou ao desenvolvimento de algoritmos de aprendizagem, pesquisa e optimização com provas dadas em problemas da inteligência artificial. Entre estes algoritmos constam os algoritmos evolutivos que se baseiam na teoria da selecção natural e no princípio da sobrevivência do mais apto.

Neste capítulo é feita uma abordagem dos principais algoritmos evolutivos e das suas características. Em certos casos, para uma melhor compreensão, é feita uma analogia entre o processo natural e o algoritmo evolutivo "correspondente". Neste contexto são também apresentadas técnicas com vista a melhorar e optimizar os algoritmos genéticos (AGs). Segundo esta ordem de ideias, nas secções 2.2 e 2.3 é feita uma introdução da selecção natural, e é estabelecida a correspondência entre as terminologias da genética natural e da ciência da computação. De seguida, na secção 2.4 são apresentados os algoritmos genéticos. Na secção 2.5, são apresentados os AGs não binários e enumeradas as principais diferenças em relação aos AGs binários. Por último, na secção 2.6, são referenciados alguns mecanismos que favorecem

a diversidade genética da população.

2.2 Introdução à teoria da selecção natural

A teoria da selecção natural, ou seja, a sobrevivência dos seres mais aptos, governa a adaptação evolutiva do mundo biológico, desde o vírus mais pequeno até ao mamífero mais complexo. Assim, a selecção natural opera em todos os organismos de modo a criar descendentes com as melhores características possíveis. Os descendentes resultam, normalmente, do crescimento de uma célula que contém um exemplar de material genético dessa espécie. É através da criação desse material específico que os progenitores influenciam a estrutura e as funções hereditárias dos descendentes. Espécies diferentes criam material genético distinto para os respectivos descendentes, mas, em todas as espécies, os ascendentes providenciam esse material. Em alguns casos, existem dois progenitores, sendo o material genético destes combinado para criar o descendente. Nos casos em que existe só um progenitor, o material genético hereditário sofre mutações (modificações aleatórias do material genético) garantindo desta forma que o descendente não seja igual ao seu progenitor.

2.3 Transposição da teoria de selecção natural para a ciência da computação

A evolução da espécie através da capacidade de reprodução e a criação do plano genético dos descendentes, são temas centrais na adaptação evolutiva de organismos biológicos. Esta perspectiva pode ser aplicada à adaptação evolutiva de estruturas computacionais, resultando a área da computação evolutiva (CE) ou também denominada como algoritmos evolutivos (AEs). Assim, a teoria da computação evolutiva consiste em aplicar os conceitos da selecção natural, baseados na aptidão de uma população de estruturas, num algoritmo de computador. A evolução é então um

processo de optimização que é simulado num computador adicionando-se eventualmente outros métodos comuns nas ciências de engenharia. O interesse destas simulações tem aumentado significativamente nos últimos anos. As aplicações deste novo tipo de algoritmos tem vindo a substituir os métodos clássicos em diversas áreas como, por exemplo, em síntese de circuitos eléctricos [18, 19], no reconhecimento de padrões [20], em identificação de sistemas [21], em sistemas de controlo [22, 23], no planeamento do uso da terra e dos transportes [24], na biologia molecular [25], na epidemiologia genética [26], em sistemas de energia [27] e programas de análise económica e financeira [28].

Os algoritmos evolutivos inspiram-se no processo de evolução natural que conduziu ao aparecimento de estruturas orgânicas complexas e bem organizadas. Por outras palavras, a evolução é o resultado da criação de material genético novo e da sua selecção. Com base num critério de desempenho, um indivíduo de uma população é afectado por outros indivíduos (*e.g.*, a competição por alimentos, a existência de predadores, e o acasalamento), e pelo ambiente (*e.g.*, comida e clima existentes). Assim, a capacidade do indivíduo nestas funções dita a possibilidade de sobreviver e ter descendentes. Desta forma, a informação genética que normalmente passa para as gerações seguintes é a dos indivíduos mais aptos. A natureza aleatória da reprodução conduz à introdução de nova informação genética e, consequentemente, à criação de descendentes com características diferentes.

A computação evolutiva é um tipo de pesquisa meta-heurística inspirada na natureza, ver figura 2.1 (no Apêndice A são descritas as técnicas apresentadas na figura 2.1). Dentro da CE existem três ramos principais de estudo: estratégias de evolução (EEs), programação evolutiva (PE) e algoritmos genéticos (AGs). Dentro dos AGs para além da estrutura comum existem outras variantes como a programação genética (PG) e sistemas de classificação (SCs) que desenvolveram as suas próprias direcções de pesquisa e de aplicação. A computação evolutiva ou os algoritmos evolutivos (AEs) são os termos utilizados para denominar este tipo de técnicas. A

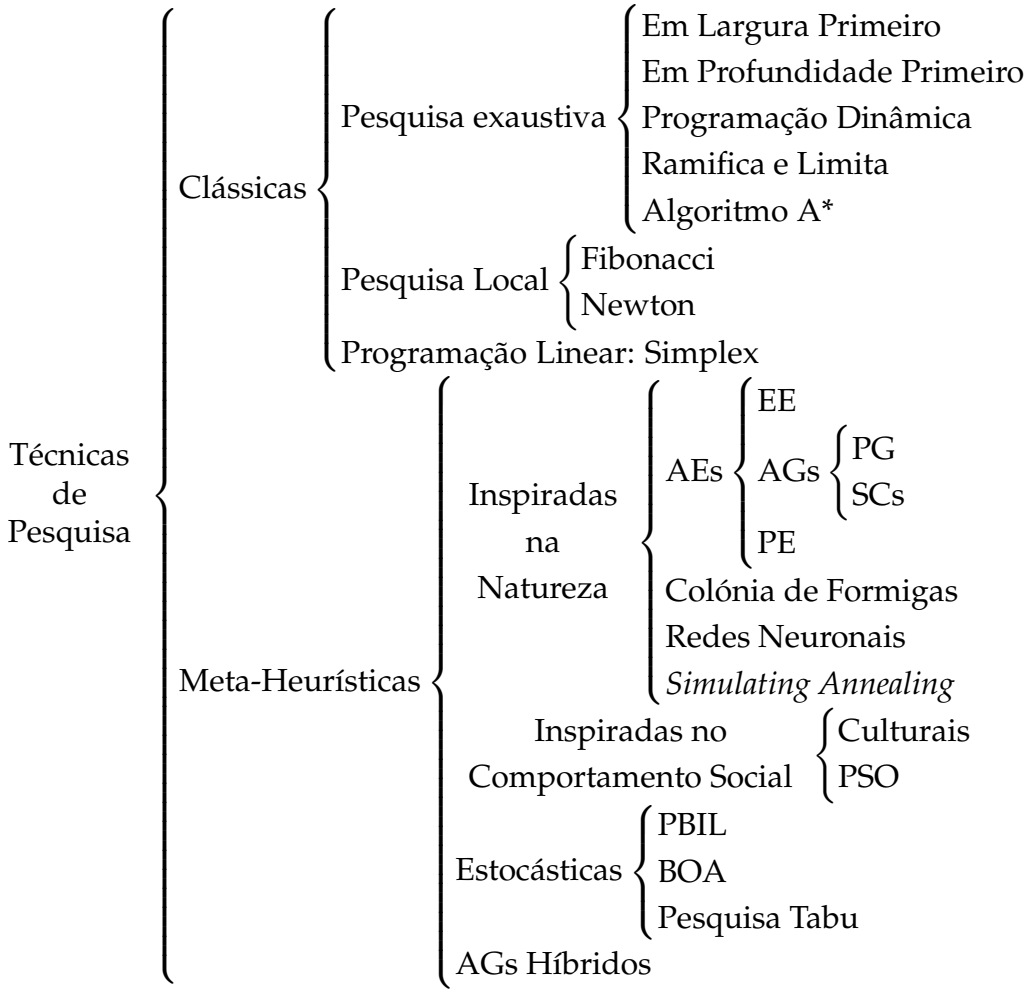


Figura 2.1. Técnicas de pesquisa

diferença entre os algoritmos referidos consiste no tipo de alterações que são introduzidas nas soluções com vista a criar os respectivos descendentes, no método de selecção dos novos progenitores e na estrutura de dados que representa das soluções.

A principal vantagem na utilização da pesquisa evolutiva tem a ver com a adaptabilidade a diferentes tarefas e com a robustez de desempenho (que obviamente depende do tipo de problema) *versus* a capacidade de execução de uma pesquisa global. Assim, os algoritmos evolutivos conduzem frequentemente a resultados excelentes quando aplicados a problemas complexos, enquanto que outros métodos

Tabela 2.1. Terminologia natural/computacional

Termo natural	Termo utilizado nos AEs
Cromossoma, indivíduo	*String*, vector, solução
Gene	Carácter, parâmetro, característica
Alelo	Valor
Lugar (*locus*)	Posição no vector
Genótipo	Estrutura de codificação
Fenótipo	Conjunto de parâmetros, soluções alternativas, estrutura de descodificação
Epistasis	Não-linearidade
Geração	Iteração

não são aplicáveis ou são pouco satisfatórios.

A grande adaptabilidade dos AEs deve-se a estes fazerem poucas suposições sobre o domínio do problema. Contudo, quando aplicados a problemas de grande dimensão, as soluções geradas através da explosão combinatória vão aumentar o custo do método. Para minimizar este fenómeno, pode ser introduzido no AE conhecimento inerente ao problema, de modo a lidar melhor com as características do mesmo. Consequentemente este AE particular não poderá ser utilizado directamente noutro tipo de problema.

As designações usadas nos AEs estão de acordo com a genética natural e com a ciência da computação. A tabela 2.1 apresenta a correspondência entre os termos mais utilizados nas duas áreas.

2.4 Algoritmos genéticos

2.4.1 Introdução

Os algoritmos genéticos, desenvolvidos por Holland [1] em 1975 na Universidade de Michigan, só nas últimas décadas têm sido utilizados para optimizações estruturadas. Os AGs são algoritmos de pesquisa baseados em mecanismos da selecção natural e da genética natural. Com estes mecanismos os indivíduos com melhores

capacidades são normalmente os vencedores num ambiente altamente competitivo. Assim, os AGs combinam a sobrevivência dos indivíduos mais aptos com vectores de estruturas. A informação criada aleatoriamente nestas estruturas é trocada para formar um algoritmo de pesquisa. Em cada geração é criada uma nova população artificial de vectores desenvolvida a partir de *bits* e/ou conjunto de *bits* dos vectores mais aptos, substituindo a população anterior.

Um tema central da investigação nos AGs tem sido a robustez, bem como o balanço entre a eficiência e a eficácia necessárias para a sobrevivência em diferentes ambientes. Os AGs funcionam bem num espectro largo do problema, enquanto que a maior parte dos algoritmos tradicionais funciona eficientemente apenas numa gama estreita de valores. Deve-se salientar que os AGs, tal como outros métodos heurísticos, não garantem que o valor óptimo seja encontrado. No entanto, encontram sempre uma solução razoável de acordo com a complexidade do problema.

Na secção seguinte é feita a comparação entre os AGs e os métodos de optimização clássicos. De seguida, na secção 2.4.3 são indicados os passos necessários para criar um AG. Na secção 2.4.4 é descrita a forma de representação e de codificação dos AGs. Posteriormente, na secção 2.4.5 é referido o espaço de pesquisa. Na secção 2.4.6 são indicados os tipos de implementações das restrições. Nas secções 2.4.7 e 2.4.8 é assinalada a importância da função de aptidão e são descritos os operadores genéticos, respectivamente. Na secção 2.4.9 são indicadas estratégias de formação da geração seguinte. Na secção 2.4.10 são referidos os modos de finalização do AG. Por último, na secção 2.4.11 é enunciada a convergência de um AGs.

2.4.2 Comparação entre os algoritmos genéticos e os métodos clássicos de optimização

Os AGs diferem dos algoritmos clássicos de optimização e pesquisa nos seguintes pontos [29]:

- Os AGs trabalham com os parâmetros do problema codificados em vez do uso directo dos parâmetros;
- Os AGs pesquisam um conjunto de pontos em simultâneo, e não apenas um ponto;
- Os AGs usam informação de troca (funções objectivo), ao contrário de cálculos analíticos (baseados nos operadores integro-diferenciais) e outros conhecimentos auxiliares;
- Os AGs usam regras de transição estocásticas, e não regras deterministas.

As vantagens dos AGs são:

- As restrições do problema são facilmente tratadas, inserindo-as no vector de codificação, ou através da função de aptidão;
- O AG é uma técnica que pode ser utilizada em problemas com vários óptimos locais, não diferenciáveis, não contínuos, não polinomiais, multi-dimensionais e com múltiplos objectivos;
- Os AGs estruturados são uma ferramenta que podem ser facilmente implementados em sistemas paralelos.
- Os AGs são uma técnica de simples compreensão, requerendo cálculos matemáticos poucos elaborados;
- A "interface" entre as simulações e os modelos existentes é simples;
- Os AGs executam uma pesquisa multi-direccional, mantendo uma população de potenciais soluções, encorajando a construção e a troca de informação entre essas soluções.

Numa segunda fase foram introduzidas nos AGs várias modificações com a finalidade de aumentar o desempenho e o espaço de pesquisa para um problema em particular, nomeadamente, vectores com comprimento variável, vectores que incluem regras do tipo *se então senão* [30–32], e estruturas mais ricas do que vectores binários (*e.g.*, matrizes, árvores, pilhas, listas [33] e alterações nos operadores genéticos).

2.4.3 Projecto de um algoritmo genético

Para desenvolver um AG é necessário especificar os requisitos seguintes:

- Escolher o esquema de representação considerando:
  1. A dimensão do alfabeto;
  2. O tipo de selecção e a projecção entre os vectores e os pontos do espaço.
- Definir a função de aptidão;
- Determinar os parâmetros e as variáveis necessários para controlar o algoritmo entrando em linha de conta com:
  1. O número de vectores da população (pop_{dim});
  2. O número de gerações (T_{t});
  3. A probabilidade de reprodução (p_{r});
  4. A probabilidade de cruzamento (p_{c});
  5. A probabilidade de mutação (p_{m});
  6. A percentagem de população que vai ser substituída.
- Método adoptado para criar a população inicial;
- O mecanismo de selecção;
- O tipo dos operadores de variação:

1. Cruzamento;
2. Mutação.

- Determinar o modo de preservar o(s) vector(es) que representa(m) o(s) melhor(es) resultado(s).
- Método de implementação das restrições do problema;
- Definir o critério para terminar o processo evolutivo.

Após a escolha dos requisitos necessários para a implementação do AG, a estrutura de um AG simples é fornecida pelo algoritmo 2.1, onde P(T) é a população na geração T. As funções do algoritmo serão explicadas nas subsecções seguintes.

```
1  início
2      T = 0;
3      inicializar aleatoriamente P(T);
4      avaliação P(T);
5      repetir
6          cruzamento P(T);
7          mutação P(T);
8          avaliação P(T);
9          obter P(T + 1) a partir de P(T);
10         T = T + 1;
11     até condição de conclusão verificada;
12 fim
```

Algoritmo 2.1. Algoritmo genético simples

2.4.4 Representação e codificação dos algoritmos genéticos

Os AGs canónicos utilizam uma representação vectorial binária de dimensão fixa podendo cada carácter do vector ter o valor '0' ou '1'. Assim, os AGs utilizam frequentemente funções de codificação e descodificação, de modo a facilitar a projecção entre os vectores binários e as soluções do problema. O sucesso dos AGs é muitas

vezes dependente da função de codificação. No caso de problemas de optimização com parâmetros contínuos, os algoritmos genéticos representam normalmente o conjunto de parâmetros reais por um vector, sendo este dividido em vários segmentos (um por cada parâmetro (gene)) com o mesmo comprimento. Cada segmento contém um valor inteiro, V_c, que é projectado linearmente no intervalo que o parâmetro real correspondente pode tomar. Este valor, v_r, pode ser calculado através de l_p caracteres binários com uma resolução π_r de acordo com a seguinte fórmula:

$$v_r = \pi_r V_c + U_{\min} \tag{2.1a}$$

$$\pi = \frac{U_{\max} - U_{\min}}{2^{l_p} - 1} \tag{2.1b}$$

onde: $U_{\max}$ e $U_{\min}$ são os valores máximo e mínimo do parâmetro a codificar, respectivamente.

Deve notar-se que os parâmetros reais podem não ser a única discretização necessária. Muitos problemas de optimização têm também parâmetros e funções de controlo que são discretizadas.

O argumento fundamental que justifica o uso do alfabeto binário nos algoritmos genéticos é devido ao facto do número de *arranjos* ser máximo para um conjunto de pontos de pesquisa num algoritmo binário [29]. Consequentemente, o teorema do *esquema* favorece a representação binária das soluções. Além da sua simplicidade o uso do alfabeto binário facilita a análise teórica e permite operadores genéticos estruturalmente elegantes. Todavia a representação binária tem a desvantagem da função de codificação poder introduzir picos adicionais, fazendo com que a função binária de optimização se torne mais complexa do que a função real inicial. Outro problema da codificação binária é o efeito dos picos de Hamming. No entanto este pode ser eliminado, a precisão e convergência melhorada usando a codificação Gray [34].

2.4.5 Espaço de pesquisa

Na maioria das aplicações o espaço de pesquisa é definido por um conjunto de objectos (*e.g.*, unidades de processamento, bombas hidráulicas, fornos e arrefecedores para uma unidade industrial) cada um com parâmetros diferentes (tais como, consumo de energia e capacidade). Estes parâmetros, que são sujeitos à optimização, constituem o chamado espaço fenótipo. Por outro lado, os operadores genéticos trabalham frequentemente com objectos abstractos, matemáticos (tais como vectores binários) que constituem o espaço genótipo. A correspondência entre estes dois espaços é feita através da função de codificação.

Existem dois processos que são normalmente utilizados para representar o problema. No primeiro processo escolhe-se um algoritmo e uma função de codificação de acordo com os parâmetros requeridos pelo algoritmo. No segundo processo utiliza-se uma representação o mais próxima possível das características do problema real (espaço fenótipo). Deste modo, evita-se frequentemente a necessidade de uma função de codificação.

Existem muito resultados empíricos e teóricos que se encontram disponíveis para os algoritmos evolutivos normalizados [35–39]. Este conhecimento constitui uma vantagem numa primeira aproximação e permite reutilizar os parâmetros e os operadores destes algoritmos. Por outro lado, funções de codificação complexas podem introduzir não linearidades e outras dificuldades matemáticas que podem prejudicar substancialmente o processo de pesquisa. *A priori* não se sabe qual das soluções é a melhor para um determinado processo mas, diversas aplicações têm mostrado que os algoritmos modificados, de acordo com o problema a resolver, têm um desempenho superior aos algoritmos normalizados.

2.4.6 Restrições dos problemas

Uma consequência dos AGs serem independentes do domínio é a sua capacidade de resolver problemas com restrições não triviais. Estas restrições podem ser implementadas introduzindo penalidades nos indivíduos que não satisfazem essas restrições (ao nível da função de aptidão) ou através de funções de descodificação que permitam representar apenas os vectores que verifiquem essas restrições. No caso das restrições serem resolvidas ao nível da função de aptidão podem surgir alguns inconvenientes. De facto, se for introduzida uma penalidade muito grande na função de aptidão, e se as restrições do domínio forem violadas pela representação dos indivíduos, então o algoritmo pode criar um grande número de vectores ilegais. Por um lado, quando é encontrado um vector que satisfaz essas restrições, pode acontecer que, este conduza os restantes na sua direcção sem se encontrarem as melhores soluções. Por outro lado, se a penalidade introduzida é pequena, o sistema pode apresentar soluções que não verifiquem as restrições com um desempenho superior ao de outros vectores que satisfazem as mesmas restrições. Isto deve-se ao facto da parte da solução que satisfaz o domínio de um vector apresentar um desempenho superior ao dos outros vectores. No segundo caso, a representação tendo em vista que os indivíduos ilegais não seja viável, é bastante intensiva computacionalmente e, por vezes, difícil ou mesmo impossível de implementar.

2.4.7 Função de aptidão

Os termos função de *aptidão* e *objectivo* indicam a preferência dos parâmetros a optimizar. Normalmente, estes termos são utilizados para referenciar a função final que quantifica a relação entre os objectivos e não são diferenciados. No entanto, estes termos têm um significado diferente que será considerado nesta secção. Assim, a função objectivo, f_o, é a função que quantifica o desempenho dos objectivos, a função de aptidão, f, é a função objectivo modificada com o intuito de controlar

certas características do algoritmo[1]. Consequentemente a função de aptidão constitui uma ligação importante entre os AGs e o sistema a tratar. A função de aptidão deve ser definida positiva no intervalo $[0; 1]$. Contudo, maximizar um problema não garante que a função objectivo apresente valores positivos para todos os vectores. Adicionalmente, existem problemas em que interessa minimizar. De modo a resolver os problema referidos, é frequente projectar a função objectivo numa função de aptidão.

Para transformar um problema de minimização num problema de maximização pode utilizar-se a função (2.2). Sendo a função f_o a função objectivo inicial; a função f a nova função objectivo (aptidão); e o valor C_{max} é o valor máximo da função f_o observado na geração i.

$$f = \begin{cases} C_{max} - f_o, & f_o < C_{max}; \\ 0, & \text{Outros casos.} \end{cases} \tag{2.2}$$

No entanto, se o problema é de maximização pode usar-se a equação 2.3, onde: o valor $-C_{min}$ é o valor negativo mínimo da função f_o observado na geração i.

$$f = \begin{cases} C_{min} + f_o, & f_o + C_{min} > 0; \\ 0, & \text{Outros casos.} \end{cases} \tag{2.3}$$

A função de aptidão varia de acordo com o problema a resolver, nomeadamente com os parâmetros que se pretendem optimizar e com as restrições do problema. Para manter a uniformidade entre os diversos problemas, usa-se uma projecção após a função objectivo. As projecções utilizadas mais frequentemente são: escalonamento linear, escalonamento por potência e truncamento sigma. As suas características são:

- Escalonamento linear (*linear scaling*): o valor de aptidão ($f(v_j)$) do vector j tem a seguinte relação linear com o valor objectivo:

[1] Em certos métodos de selecção, como por exemplo o método proporcional.

$$f(v_j) = a\, f_{\mathrm{o}}(v_j) + b \tag{2.4}$$

onde os coeficientes a e b podem ser escolhidos de diversos modos. Um exemplo consiste em assegurar a igualdade entre a média da função objectivo ($\bar{f}_{\mathrm{o}}$) e a média dos valores de aptidão. Assim, com o uso do procedimento de selecção, espera-se que cada membro médio da população contribua com um descendente para a geração seguinte. Se n é a relação entre o valor de aptidão máximo e o valor médio de aptidão então n é o número de descendentes esperados pelo melhor vector da população. Deste modo, o valor de aptidão máximo pode ser escolhido como um múltiplo da média da aptidão de forma a controlar o número esperado de descendentes do melhor vector da população. Quando se utiliza esta projecção deve-se assegurar que os valores de aptidão não apresentem valores negativos. Se os parâmetros a e b forem constantes ao longo das gerações este escalonamento é independente da evolução do problema;

- Escalonamento por potência *(power law)*: O valor de aptidão é calculado a partir da potência de ordem k do valor objectivo:

$$f(v_j) = f_{\mathrm{o}}(v_j)^k \tag{2.5}$$

onde k varia de acordo com o problema ou, eventualmente, durante a sua execução. A gama de valores típica de k é próxima de 1 (*e.g.*, $k = 1,005$);

- Truncamento sigma: O valor de aptidão ($f(v_j)$), do vector j, é calculado de acordo com a seguinte expressão:

$$f(v_j) = f_{\mathrm{o}}(v_j) + (\bar{f}_{\mathrm{o}} - c\sigma) \tag{2.6}$$

onde c é um valor inteiro pequeno (normalmente $1 \leq c \leq 5$), $\bar{f}_{\mathrm{o}}$ é a média dos valores objectivos e σ é o desvio padrão da população.

Para prevenir valores negativos da função $f(v_j)$, qualquer resultado negativo (*i.e.*, $f(v_j) < 0$) é colocado arbitrariamente a zero. Este mecanismo foi introduzido para melhorar o método escalonamento linear.

As projecções das funções de aptidão permitem controlar a convergência dos problemas. Por exemplo, considere-se as funções de aptidão $f_1(v)$ e $f_2(v)$: $f_1(v) = f_2(v) + c$ onde c é uma constante. Se $c \gg \bar{f}_1(x)$ ($\bar{f}_1$ é a média da função de aptidão f_1) então a função $f_2(v)$ vai ter uma convergência muito mais lenta que a função $f_1(v)$[2]. No caso extremo a função $f_2(v)$ terá uma pesquisa aleatória enquanto que a função $f_1(v)$ terá uma convergência prematura. Este fenómeno deve-se à incapacidade do operador de selecção conseguir distinguir "pequenas diferenças" em valores muito grandes.

Muitos problemas contêm restrições que devem ser satisfeitas e que podem ser resolvidas ao nível da função de aptidão. Um método que satisfaz as restrições do problema é o seguinte:

1. O modelo é executado;
2. As funções objectivo são avaliadas;
3. Verifica-se se existe alguma restrição que não é satisfeita;
4. Se existir alguma restrição que não é verificada, então a solução não é admissível e, consequentemente, a sua função de aptidão tem o valor zero (considera-se que o problema é de maximização).

Os pontos fracos deste método aparecem quando o problema tem muitas restrições pois, nestes casos, encontrar uma solução admissível é tão difícil como encontrar a melhor solução.

[2]Considera-se que o operador de selecção é o proporcional

Noutro método, o das penalidades, um problema com restrições é transformado noutro sem restrições. Esta transformação é feita associando um custo, ou penalidade, a todas as restrições que não são satisfeitas. Este custo é incluído na função objectivo. Assim, o seguinte problema com restrições:

$$\begin{array}{ll} \text{Minimizar:} & g(x) \\ \text{Sujeito a:} & b_i \geq 0 \quad i = 1, 2, \ldots, n \end{array} \tag{2.7}$$

é transformado no seguinte problema sem restrições:

$$\min \{g(x)\} + r \sum_{i=1}^{n} \phi[b_i(x)] \tag{2.8}$$

onde: x é um vector de dimensão m; $g(x)$ é a função a minimizar; b_i é a restrição i do problema; ϕ é a função de penalidade e r é o coeficiente de penalidade.

A função de penalidades pode ter muitas representações como, por exemplo a equação (2.9).

$$\phi[b_i(x)] = b_i^2(x) \tag{2.9}$$

2.4.8 Operadores genéticos

Introdução

Existem três tipos principais de operadores nos algoritmos genéticos: selecção, recombinação (cruzamento) e mutação. Dentro destes tipos principais existe ainda uma grande variedade de operadores mas todos devem obedecer a certas propriedades como, por exemplo, o teorema do *esquema*. Nos AGs canónicos são utilizados apenas os operadores de cruzamento binário e de mutação binária. Nesta secção são apresentados e estudados os principais operadores.

Mecanismo de selecção

Introdução

Para gerar descendentes com boas características é necessário um bom mecanismo de selecção de progenitores proficientes. Este processo é adoptado para determinar o número de acasalamentos, utilizados na reprodução, para um indivíduo em particular. A probabilidade de escolher um vector como progenitor deve ser directamente proporcional ao número de descendentes produzidos. A ideia principal por detrás do operador é que soluções com melhor desempenho (aptidão) devem ter uma probabilidade superior de serem seleccionadas para participar no processo de acasalamento. Os operadores de selecção diferem no modo como diferenciam as melhores soluções e como as seleccionam. O operador de selecção é baseado essencialmente[3] nos valores de aptidão dos indivíduos. Nos AGs a selecção é normalmente implementada baseada num operador estocástico, utilizando a aptidão relativa

Nesta secção são referidos os principais operadores de selecção. São ainda apresentadas medidas de desempenho e de classificação do mecanismo de selecção.

Selecção proporcional

Na selecção proporcional (selecção da roleta) o número de cópias esperado de um determinado vector, v_i, está relacionado proporcionalmente com o seu valor de aptidão. A probabilidade de escolher uma solução de uma população de pop_{dim} elementos é dada pela formula (2.10). Os valores de aptidão $f(v_k)$ devem ser previamente projectados de modo a serem todos positivos.

$$p(v_i) = \frac{f(v_i)}{\sum_{k=1}^{pop_{\text{dim}}} f(v_k)} \tag{2.10}$$

[3]Existem algoritmos de selecção que restringem o número de acasalamentos de um indivíduo.

Existem algumas variações do modelo nomeadamente:

- Modelo do valor esperado. Este método reduz os erros estocásticos da rotina de selecção. O algoritmo introduz um contador para cada vector da população (que é inicializado com o seu valor de aptidão sobre o valor de aptidão médio da população) diminuído de 0,5 ou 1 quando o cromossoma é seleccionado para reprodução com o cruzamento ou para mutação, respectivamente. Quando o contador desce abaixo de zero, o vector deixa de estar disponível para ser seleccionado;

- Modelo elitista do valor esperado. Este modelo é idêntico ao anterior excepto que o melhor vector é sempre copiado integralmente para a geração seguinte, sem participar no processo de cruzamento;

- Modelo de amostragem estocástica com substituição (*remainder stochastic sampling with replacement*). O algoritmo selecciona os vectores de acordo com a parte inteira do valor esperado do número de ocorrências destes. A população é completada pelo modelo proporcional utilizando a parte decimal como aptidão dos vectores.

Selecção por posto

Este operador utiliza os índices dos indivíduos, quando ordenados de acordo com os valores de aptidão, para calcular a probabilidade de selecção correspondente. Existem dois tipos de projecções: lineares (2.11a) e não lineares (2.11b).

$$p(posto) = q - (posto - 1)r \tag{2.11a}$$

$$p(posto) = q(1-q)^{posto-1} \tag{2.11b}$$

$$\sum_{i=1}^{pop_{\dim}} p_i = 1 \tag{2.11c}$$

$$q = 0,5r(pop_{\dim} - 1) + {pop_{\dim}}^{-1} \tag{2.11d}$$

Ambas as funções determinam a probabilidade de um indivíduo ser escolhido numa amostra. Considera-se que os indivíduos se encontram cotados numa escala de níveis ($posto = 1$ ($posto = pop_{\text{dim}}$) quando o indivíduo em questão é o melhor (pior) da população). O parâmetro r, escolhido pelo utilizador, controla a pressão do método de selecção:

- para $r = 0$ não existe pressão no método, isto é, todos os pop_{dim} indivíduos têm a mesma probabilidade de serem seleccionados;
- contrariamente para $r = 2/({pop_{\text{dim}}}^2 - pop_{\text{dim}})$ a pressão de selecção é máxima).

Tanto a equação (2.11a) como a equação (2.11b) devem satisfazer as equações (2.11c) e (2.11d). Além da sua simplicidade, este método evita que uma solução com um sobre-desempenho, *i.e.*, com um desempenho muito grande relativamente às outras soluções domine o processo de selecção, prevenindo assim a convergência prematura. Adicionalmente, este tipo de selecção adequa-se a problemas onde é difícil quantificar a preferência entre soluções.

Este método apresenta um bom comportamento para certos tipos de AGs mas, em contrapartida, tem as seguintes desvantagens:

- A responsabilidade de escolher os parâmetros é deixada ao cargo do utilizador;
- O algoritmo não verifica o teorema do *esquema*.

Selecção por torneio

Este tipo de selecção retira uma amostra de q ($q > 1$) vectores da população, segundo uma lei aleatória uniforme. O melhor vector, da amostra, é dado como vencedor e a operação é repetida até que o número de vectores seleccionados preencha o

número de progenitores desejado. Este método é bastante popular devido à sua fácil implementação, do ponto de vista da eficiência computacional, e permite controlar o peso da selecção pelo aumento ou diminuição da dimensão do torneio (q).

Selecção de Boltzmann

Na selecção de Boltzmann, um valor de aptidão é afectado a um vector de acordo com a distribuição probabilística de Boltzmann (2.12), onde o parâmetro T_{k}, na distribuição da equação (2.12), é análogo com a temperatura no processo de arrefecimento de metais. Este parâmetro deve ser reduzido, de modo pré-definido, ao longo das gerações. Com o valor de T_{k} grande todos os vectores têm praticamente a mesma probabilidade de serem seleccionados, mas à medida que a T_{k} diminui a selecção das boas soluções aumenta em detrimento das restantes.

$$\frac{1}{1+\mathrm{e}^{\frac{f(v_i)}{T_{\mathrm{k}}}}} \tag{2.12}$$

Desempenho do mecanismo de selecção

De acordo com Grefenstette [39] existem três medidas de desempenho no mecanismo de selecção: *direcção* (*bias*), *extensão* (*spread*), e *eficiência*. A medida *direcção* indica a diferença absoluta entre as probabilidades esperadas e o número de vectores seleccionados posteriormente. A medida *extensão* é o intervalo entre zero e o número possível de acasalamentos que um indivíduo pode realizar. A *eficiência* está relacionado com o tempo de execução do algoritmo de selecção.

O método de selecção proporcional tem uma direcção próxima de zero, mas tem uma extensão ilimitada. O método proporcional pode ser implementado com um tempo na ordem de $pop_{\mathrm{dim}} \log(pop_{\mathrm{dim}})$, onde pop_{dim} é o número de vectores da população. Outro algoritmo de amostragem de fase simples consiste na amostragem estocástica universal (*stochastic universal sampling*) com direcção nula, extensão mínima e um tempo de execução do algoritmo na ordem de pop_{dim}.

Classificação do mecanismo de selecção

Existem dois temas importantes na pesquisa genética: a diversidade da população e a pressão da selecção (medida proporcional correspondente à diferença dos valores de aptidão que os vectores podem apresentar). Se a função de selecção é muito apurada (*i.e.*, selectiva) a diversidade da população tende a diminuir. Por outro lado, se o método é pouco selectivo então a diversidade da população tende a aumentar. Assim, se o método de selecção é muito apurado o método tende a convergir prematuramente.

Os métodos de selecção podem ser classificados nos seguintes tipos:

- Métodos *estáticos versus* métodos *dinâmicos*. Os métodos *estáticos* requerem que as probabilidades de selecção se mantenham constantes ao longo das gerações, enquanto que nos métodos *dinâmicos* as probabilidades de selecção podem variar de geração para geração;

- Métodos *comedidos versus* métodos *extintivos*. Os métodos *comedidos* requerem que a probabilidade de selecção de um vector seja não nula. Os métodos *extintivos* podem ainda subdividir-se em selecção *esquerda* e *direita*. No método *extintivo esquerdo* os melhores vectores são proibidos de se reproduzirem, evitando eventuais convergências prematuras. Na selecção pelo método *extintivo direito* o melhor vector tem uma reprodução normal;

- Uma selecção diz-se *pura* se o tempo de vida dos vectores é de uma geração.

- Métodos *geracional versus* métodos *substituição imediata*. Quando são fixados os progenitores, até que a nova geração seja completada, a selecção designa-se por *geracional*. Contrariamente à selecção de *substituição imediata* onde os descendentes são inseridos na população após a sua criação;

- Os modelos *elitistas* permitem que alguns progenitores passem para a geração seguinte, em função do seu valor de aptidão.

Ponto de cruzamento	$pc = 3$	
Vectores Progenitores	10010000	**01100111**
Número do *bit*	*123*|*45678*	*123*|*45678*
Vectores Resultantes	100**00111**	**011**10000

Figura 2.2. Cruzamento simples

Pontos de cruzamentos	$pc = \{3, 6\}$	
Vectores Progenitores	10010000	**01100111**
Número do *bit*	*123*|*456*|*78*	*123*|*456*|*78*
Vectores Resultantes	100**001**00	**011**100**11**

Figura 2.3. Cruzamento de ponto duplo

Operador de cruzamento

O algoritmo canónico utiliza o cruzamento (recombinação) de ponto simples. Para esse efeito são seleccionados dois indivíduos da população. De seguida é escolhido, aleatoriamente, um ponto nos vectores, que consiste no ponto de cruzamento, *pc*, (figura 2.2). Desta forma, um descendente é criado a partir dos *bits* da parte esquerda de um progenitor com os *bits* da parte direita do segundo progenitor. As partes restantes serão utilizadas para formar um segundo descendente. Este operador permite várias extensões tais como o número de pontos de cruzamento ser superior a um (cruzamento de ponto duplo ver figura 2.3) e o cruzamento ser uniforme. Este operador é aplicado à população com uma probabilidade p_c (normalmente $p_c \approx 0{,}6$ para populações grandes $pop_{\text{dim}} \geq 100$ e $p_c \approx 0{,}9$ para populações pequenas $pop_{\text{dim}} \leq 30$). O aumento da probabilidade de cruzamento provoca o aumento da recombinação na construção de blocos.

No cruzamento uniforme cada *bit* do descendente é escolhido aleatoriamente do *bit* da posição correspondente de um dos progenitores. Para esse fim é utilizada uma máscara gerada aleatoriamente. Um exemplo de um cruzamento deste tipo é exemplificada na figura 2.4. O número de pontos de cruzamento não é fixo mas tem, em média, $l/2$ pontos de cruzamento (onde l é o comprimento do vector).

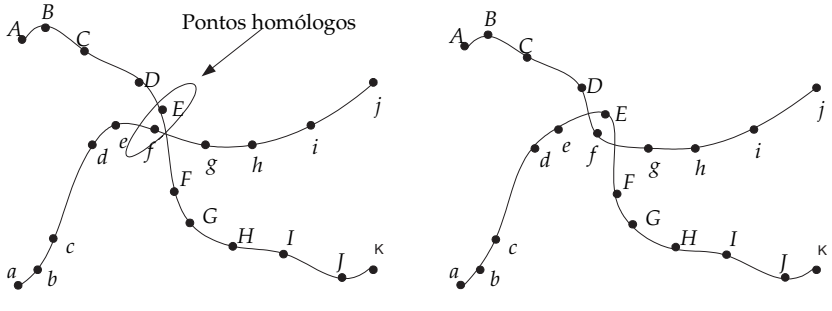
A recombinação com vários progenitores consiste em criar um descendente a partir

Máscara	01001110	
Vectores Progenitores	10100001	**01000111**
Número do bit	*12345678*	*12345678*
Vectores Resultantes	11100111	00000001

Figura 2.4. Cruzamento uniforme

de vários vectores (*i.e.*, mais do que dois vectores).

Em contraste com o cruzamento tradicional, onde o ponto de cruzamento é escolhido de acordo com a posição do vector, no cruzamento análogo [40] o local de cruzamento é baseado em semelhanças na características do fenótipo. Este operador usa uma função fenotípica de parâmetros como critério do ponto de cruzamento. O cruzamento análogo não só preserva a ordem da característica do vector como também tem uma justificação biológica (*i.e.*, órgãos análogos são órgãos ou partes de corpos adaptados para servir o mesmo propósito). O cruzamento de parâmetros de acordo com o seu carácter genotípico é superior ao cruzamento de acordo com a sua posição genotípica, para vectores onde o número, o tamanho e a posição dos parâmetros não tem uma estrutura rígida. Neste tipo de estruturas é importante que o cruzamento ocorra entre locais que controlam a mesma função ou uma função similar no espaço fenotípico. A figura 2.5 exemplifica o cruzamento análogo. A distância menor entre os dois vectores, neste caso, identifica os pontos homólogos. Sendo cada descendente criado com a parte direita, de um progenitor, um ponto homólogo e a parte direita do outro progenitor.



(a) Vectores progenitores (b) Vectores descendentes

Figura 2.5. Cruzamento análogo

Vectores inicial	`1011`**`1`**`011`
Número do *bit*	*`12345678`*
Vector resultante	`1011`**`0`**`011`

Figura 2.6. Mutação no *bit* 5

Na natureza o local de cruzamento pode ocorrer entre genes ou até dentro de genes. O cruzamento entre genes (*i.e.*, o cruzamento que não divide genes) é chamado cruzamento de segregação [41]. Quando os parâmetros com vários símbolos têm significado natural, o cruzamento dentro destes grupos pode ter um efeito de ruptura no genótipo e, consequentemente, no fenótipo.

Operador de mutação

Inicialmente, o operador de mutação foi introduzido como um operador secundário e de menor importância. O operador canónico de mutação inverte um *bit* com uma probabilidade p_{m} bastante baixa. Normalmente $p_{\mathrm{m}} \approx 0{,}001$ para grandes populações $pop_{\mathrm{dim}} \geq 100$ indivíduos e $p_{\mathrm{m}} \approx 0{,}01$ para populações pequenas, $pop_{\mathrm{dim}} \leq 30$ indivíduos, $p_{\mathrm{m}} \in [0{,}005; 0{,}01]$ ou $p_{\mathrm{m}} = 1/l$ (onde l é o comprimento do vector)[42]. A mutação é responsável pela perda de alelos, prevenindo assim possíveis convergências para óptimos locais, e introduzir uma pequena pesquisa aleatória. Probabilidades de mutação elevadas aumentam o tempo de convergência do algoritmo, mas ajudam na prevenção da convergência prematura. De facto, o aumento da probabilidade de mutação transforma o AG num algoritmo de pesquisa aleatório e permite a reintrodução de material genético perdido. Este operador é aplicado a todos os descendentes depois da recombinação. Cada *bit* pode ser alterado com uma probabilidade p_{m}. O *bit* que sofre a mutação é modificado para o valor complementar. Deste modo garante-se que todos os pontos de pesquisa têm probabilidade não nula de serem examinados. Na figura 2.6 está ilustrado o efeito da mutação no *bit* 5 de um vector. Inicialmente o *bit* tem o valor '1', ao ocorrer a mutação o valor passa a ter o seu valor complementar '0'.

Pontos de inversão	$pi = \{2, 6\}$
Vectores Progenitores	10**0111**00
Número do *bit*	*12|3456|78*
Vector resultante	10**1110**00

Figura 2.7. Operador de inversão

Uma variante do operador de mutação são os operadores de reordenação e de inversão. Estes operadores têm como finalidade encontrar ordenações dos genes que tenham um potencial evolutivo superior ao normal. O principal mecanismo que deu origem à reordenação foi o operador de inversão. A inversão consiste em:

1. Escolher dois pontos aleatórios do vector;
2. Os *bits* entre esses dois pontos são retirados do vector;
3. Os *bits* são recolocados por ordem inversa.

Pode-se ver o funcionamento do operador de inversão na figura 2.7 onde os valores dos *bits* 3, 4, 5 e 6 são invertidos.

Na natureza os genes são independentes do lugar que ocupam. Para ter uma certa flexibilidade na representação utilizam-se nomes para os genes. Com esta representação o valor dos símbolos é independente do seu lugar. Os operadores de reordenação não têm qualquer influência na função de descodificação e na função de aptidão.

Nas populações onde os vectores contêm fracas ordenações, os alelos com epistase elevada ou com interacções não-lineares estão espaçados através de grandes distâncias. Consequentemente, o cruzamento tem uma grande probabilidade de destruir importantes *arranjos* de alelos. Contudo, se o operador de reordenação modificar a posição dos alelos, então existe uma certa probabilidade de se conseguirem boas ordenações de alelos que permitam a construção de blocos, com uma, eficiência superior à inicial.

Os seguintes operadores de reordenação combinam características de reordenação e cruzamento: cruzamento parcialmente semelhante (*Partially matched crossover*, PMX), cruzamento ordenado (*order crossover*, OX) e cruzamento cíclico (*cycle crossover*, CX). Este tipo de operadores são bastante utilizados no problema do caixeiro viajante [43] e em problemas de escalonamento em que a ordem dos parâmetros a optimizar seja crucial [44].

O operador de translocação (*translocation*) garante que o cromossoma esteja organizado de uma determinada forma de modo a que o operador de segregação possa explorar essa organização. Assim, é necessário referenciar os alelos, através do nome de gene, para identificar a sua função quando mudam no cromossoma, para outras posições relativas, através do operador de translocação.

O operador de duplicação duplicada um determinado gene do cromossoma e coloca-o, juntamente com o seu progenitor, dentro do cromossoma. O operador de remoção retira um gene de um cromossoma.

2.4.9 Mecanismo de reinserção

Depois de gerada a subpopulação de descendentes, existem várias estratégias de substituição da população antiga. Existe o caso onde a população antiga é totalmente substituída pela nova geração. Nesta situação é gerado um número de descendentes igual ao número de indivíduos existentes na população anterior. Esta reposição tem um aspecto negativo uma vez que o melhor vector pode não ser seleccionado para gerar novos descendentes para a geração seguinte. Assim, esta estratégia é normalmente combinada com uma estratégia elitista, de modo a que seja introduzida o melhor vector, ou um conjunto dos melhores vectores, da geração presente na geração futura. A estratégia elitista pode levar ao domínio da população por um vector mas, em contrapartida, aumenta o desempenho do algoritmo. Nos processos onde é utilizado um número pequeno de vectores é substituído somente

uma parte da população pela nova geração de vectores. Em certos casos, o pior vector é também inserido na nova população juntamente com os restantes vectores. A substituição directa dos progenitores pelos descendentes é também utilizada neste caso. Desta forma evita-se a convergência prematura do algoritmo.

2.4.10 Condição de finalização do algoritmo

Um algoritmo termina quando excede o número máximo de gerações estabelecidas (depende do problema em questão) ou quando são satisfeitas certas características de pesquisa. Existem duas características de pesquisa: uma baseada nas estruturas dos vectores (genótipo) e outra baseada no significado de um vector em particular (fenótipo). A condição de paragem do algoritmo, de acordo com a estrutura, mede a convergência da população através da verificação do número de alelos que convergiram (ver secção 2.4.11). A condição de finalização, de acordo com o significado de um vector, mede o progresso feito pelo algoritmo num número predefinido de gerações.

2.4.11 Convergência do algoritmo

Um algoritmo converge quando as funções de aptidão do melhor indivíduo e da média da população aumentam ao longo das sucessivas gerações no sentido do óptimo global. Diz-se que um gene convergiu quando 95% da população têm o mesmo valor nesse gene. Diz-se que uma população convergiu, quando todos os genes convergiram. Diz-se que houve uma convergência prematura quando a população convergiu para um óptimo local.

A analogia da convergência prematura nos AGs manifesta-se na natureza como o princípio de preempção de nicho (*niche preemption*). De acordo com este princípio, um nicho biológico na natureza tende a ficar dominados por uma espécie. Uma forma de minimizar os efeitos de: preempção de nicho, convergência prematura,

sensibilidade a condições iniciais e outros eventos aleatórios, quando são utilizados AGs é realizar várias experiências independentes para o mesmo problema. Outros métodos são discutidos na secção 2.6.

A convergência do algoritmo depende de muitos factores como da função de aptidão, o tamanho da população, as probabilidades da recombinação, o método de selecção, *etc*..

2.5 Algoritmos genéticos não-binários

2.5.1 Introdução

Esta secção aborda os AGs com representação não-binária. Os AGs não-binários diferem dos AGs canónicos basicamente na sua representação e nos operadores de variação. Assim, nesta secção é descrita a sua representação e o funcionamento dos operadores de cruzamento e de mutação.

2.5.2 Representação

Um vector é uma sequência de símbolos que, nas representações não-binárias, podem ser representados por n caracteres diferentes. Neste tipo de vectores os símbolos utilizados representam valores inteiros ou reais, dependendo do tipo de parâmetros a codificar. Este tipo de representação evita a descodificação dos parâmetros, e os operadores são mais fáceis de especificar para cada tipo de problema. Esta representação apresenta melhor desempenho (a convergência prematura é mais baixa, tem maior precisão e não necessita de funções de codificação) para problemas onde os parâmetros a codificar são reais ou inteiros. Neste caso, os operadores são bastante diferentes dos adoptados para os AGs clássicos.

2.5.3 Operador de cruzamento

O código binário, quando usado em problemas com espaço de pesquisa linear, tem alguns problemas inerentes. Uma das dificuldades são os picos de *Hamming* associados em certos vectores (como 01111 e 10000) da qual uma transição para a solução vizinha (no espaço binário) requer a transição de muitos *bits*. O efeito dos picos de *Hamming* presente no código binário provoca um impedimento artificial para uma pesquisa gradual no espaço contínuo. Outra dificuldade é a incapacidade para obter uma precisão arbitrária na solução óptima. O comprimento do vector deve ser escolhido *a priori* de modo a permitir uma determinada precisão da solução do AG. Quanto maior é a precisão maior deve ser o comprimento do vector. Para vectores de comprimento grande a população também deve ser grande. Os intervalos das variáveis devem ser escolhidos de modo a conter o óptimo. Contudo, em muitos problemas esta informação não é conhecida *a priori*.

Existe um número de implementações de AGs com parâmetros reais, onde os operadores de cruzamento e de mutação são aplicados directamente nos parâmetros de valor real. Uma vez que os parâmetros reais são usados directamente, sem qualquer vector de codificação, resolver problemas de optimização com parâmetros reais é um processo fácil quando comparado com os AGs de codificação binária. Ao contrário do AG de codificação binária, as variáveis de decisão podem ser usadas directamente no cálculo do valor de aptidão. Uma vez que o operador de selecção trabalha com o valor de aptidão, qualquer operador de selecção usado no AG binário pode ser usado no AG real. No AG real, o maior desafio consiste em saber como utilizar um par de vectores com parâmetros reais para gerar novos vectores, ou saber como perturbar o vector para um vector mutado de forma consistente. Em muitos casos o cruzamento não tem grande desempenho e pode ser visto como um operador secundário.

O cruzamento linear introduzido por Wright [45] onde dois progenitores $X^{(1,T)}$ e $X^{(2,T)}$ na geração T dão origem a três descendentes (figura 2.8):

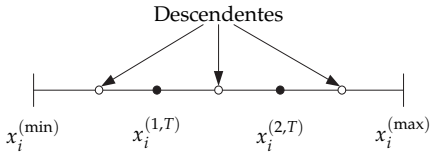


Figura 2.8. Cruzamento de Wright

1. $D^{(1,T+1)} = 0{,}5(X^{(1,T)} + X^{(2,T)})$
2. $D^{(2,T+1)} = 1{,}5X^{(1,T)} - 0{,}5X^{(2,T)}$
3. $D^{(3,T+1)} = -0{,}5X^{(1,T)} + 1{,}5X^{(2,T)}$

onde são escolhidas as duas melhores soluções.

O cruzamento de ponto simples é similar ao cruzamento utilizado no AG binário. Os pontos de cruzamento estão compreendidos entre os limites das variáveis $[1; n-1]$. Por exemplo, sendo o ponto de cruzamento $pc = 1$, *i.e.* entre as posições 1 e 2, obtêm-se os vectores descendentes D_1 e D_2 a partir dos progenitores P_1 e P_2:

$$\begin{aligned} P^{(1,T)} &= (x_1^{(1,T)}, x_2^{(1,T)}, \ldots, x_n^{(1,T)}) \\ P^{(2,T)} &= (x_1^{(2,T)}, x_2^{(2,T)}, \ldots, x_n^{(2,T)}) \\ D^{(1,T)} &= (x_1^{(1,T)}, x_2^{(2,T)}, \ldots, x_n^{(2,T)}) \\ D^{(2,T)} &= (x_1^{(2,T)}, x_2^{(1,T)}, \ldots, x_n^{(1,T)}) \end{aligned}$$

Tal como no operador de ponto simples, o operador de 2 pontos, de $n > 2$ pontos ou uniforme pode ser usado de maneira similar. Estes tipos de operadores não têm qualquer poder de pesquisa nas variáveis de decisão. Consequentemente, a pesquisa do algoritmo é efectuada essencialmente através do operador de mutação [45].

O operador de cruzamento aritmético é definido como uma combinação linear de dois vectores. Assim, sendo $P^{(1,T)}$ e $P^{(2,T)}$ os vectores progenitores, os vectores resultantes do cruzamento são:

$$\begin{aligned} D^{(1,T)} &= aP^{(1,T)} + (1-a)P^{(2,T)} \\ D^{(2,T)} &= aP^{(2,T)} + (1-a)P^{(1,T)} \end{aligned}$$

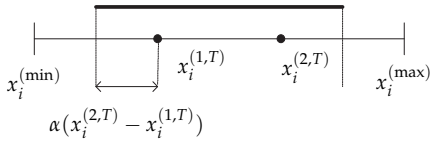


Figura 2.9. Cruzamento BLX$-\alpha$

Se o valor da variável a é constante então o cruzamento é denominado aritmético uniforme. Se a variável a varia com a idade da população, o cruzamento é dito aritmético não-uniforme.

Eshelmamn e Schaffer [45] propuseram o cruzamento combinado (*blend crossover*), BLX$-\alpha$, para a codificação real. Assim, considerando dois progenitores onde $x_i^{(1,T)} < x_i^{(2,T)}$, para os elementos i, o operador gera aleatoriamente o elemento i do descendente no intervalo:

$$[x_i^{(1,T)} - \alpha(x_i^{(2,T)} - x_i^{(1,T)}), x_i^{(2,T)} + \alpha(x_i^{(2,T)} - x_i^{(1,T)})].$$

A figura 2.9 ilustra este cruzamento. Sendo μ um número aleatório do intervalo $[0;1]$ o descendente será obtido através da equação (2.13).

$$x_i^{(1,T+1)} = (1 - \gamma_i)x_i^{(1,T)} + \gamma_i x_i^{(2,T)} \tag{2.13a}$$

$$\gamma_i = (1 + 2\alpha)\mu_i - \alpha \tag{2.13b}$$

Se $\alpha = 0$ o operador cria uma solução aleatória no intervalo $(x_i^{(1,T)}, x_i^{(2,T)})$. Os autores indicam que o valor de $\alpha = 5$ é aquele que obtém melhores resultados. Notar que γ_i tem uma distribuição uniforme para um valor fixo de α. No entanto, o intervalo dos descendentes depende das soluções progenitoras (2.14).

$$(x_i^{(1,T+1)} - x_i^{(1,T)}) = \gamma_i(x_i^{(2,T)} - x_i^{(1,T)}) \tag{2.14}$$

Se a diferença entre os progenitores for pequena então a diferença entre os descendentes é, também, pequena. Esta propriedade do operador de pesquisa permite formar uma pesquisa adaptativa. Se a diversidade da população progenitora é grande,

é esperado um descendente com uma diversidade significativa e vice-versa. Assim, com um operador deste tipo permite a pesquisa de todo o espaço (quando a população é inicializada em todo o espaço de pesquisa) e permite uma pesquisa focalizada quando a população tende a convergir numa região.

Deb *et. al.* [45] desenvolveram o operador de cruzamento binário simulado (*Simulated Binary Crossover*), SBX, que cria dois descendentes a partir de dois progenitores. O operador SBX segue o princípio de funcionamento do operador de cruzamento de ponto simples para a codificação binária. Ou seja, respeita o processamento de intervalos dos esquemas, no mesmo sentido de que os intervalos dos esquemas entre os progenitores e os descendentes sejam preservados. Para esse fim, usa o factor β_{q_i} (2.15) e a função de probabilidades (2.16) de modo a obter um poder de pesquisa idêntico ao encontrado no operador de cruzamento binário de ponto simples. Nesta equação, o parâmetro n_c é um número real não negativo. Assim, um valor grande de n_c permite criar descendentes na vizinhança dos seus progenitores com uma grande probabilidade. Por outro lado, um valor pequeno de n_c conduz a soluções distantes dos seus progenitores.

$$\beta_{q_i} = \left| \frac{x_i^{(2,T+1)} - x_i^{(1,T+1)}}{x_i^{(2,T)} - x_i^{(1,T)}} \right| \tag{2.15}$$

$$P(\beta_{q_i}) = \begin{cases} 0{,}5(n_c+1)\beta_i^{n_c}, & \beta_i \leq 1 \\ 0{,}5(n_c+1)\frac{1}{\beta_i^{n_c+2}}, & \text{Outros casos.} \end{cases} \tag{2.16}$$

O descendentes são calculados através da fórmula (2.17). O parâmetro μ_i é um valor aleatório compreendido no intervalo $[0;1]$, o valor de β_{q_i} pode ser calculado igualando a área da curva de probabilidades (2.18).

$$\begin{aligned} x_i^{(1,T+1)} &= 0{,}5(1+\beta_{q_i})x_i^{(1,T)} + 0{,}5(1-\beta_{q_i})x_i^{(2,T)} \\ x_i^{(2,T+1)} &= 0{,}5(1-\beta_{q_i})x_i^{(1,T)} + 0{,}5(1+\beta_{q_i})x_i^{(2,T)} \end{aligned} \tag{2.17}$$

$$\beta_{q_i} = \begin{cases} (2\mu_i)^{\frac{1}{n_c+1}}, & \mu_i \leq 0{,}5; \\ \left(\frac{1}{2\,(1-\mu_i)}\right)^{\frac{1}{n_c+1}}, & \text{Outros casos.} \end{cases} \tag{2.18}$$

Com este método os descendentes são simétricos em relação aos seus progenitores de modo a evitar um cruzamento polarizado. Inicialmente, quando a população se encontra dispersa, o método permite criar soluções longe dos seus progenitores permitindo desta forma explorar o espaço de pesquisa. No entanto, quando a população tende a convergir, no decorrer do algoritmo, são apenas criadas soluções na vizinhança dos seus progenitores sendo a pesquisa focalizada numa região estreita.

Para além destes operadores há a salientar os seguintes:

- Média: o descendente é criado a partir da média aritmética dos progenitores (cruzamento aritmético para $\alpha = 0,5$);
- Geométrico: o descendente é criado a partir da média geométrica (*i.e.*, da raiz quadrada do produto) dos progenitores;
- Extensão: o descendente é criado através da diferença dos valores dos dois progenitores. Esta diferença é posteriormente adicionada ao valor mais alto ou subtraída ao valor mais baixo.

2.5.4 Operador de mutação

O esquema de mutação simples (aleatório ou uniforme) consiste em criar uma solução dentro de todo o espaço de pesquisa (2.19). O parâmetro r_i é um número aleatório no intervalo $[0;1]$. Este operador é independente da solução progenitora e é equivalente à inicialização aleatória.

$$y_i^{(1,T+1)} = r_i\ (x_i^{(\max)} - x_i^{(\min)}) \tag{2.19}$$

Uma variante deste operador consiste em criar a solução numa vizinhança do seu progenitor (2.20). O parâmetro Δ_i é a perturbação i máxima definida pelo utilizador. Deve ter-se especial atenção para que o valor $y_i^{(1,T+1)}$ esteja dentro do intervalo.

$$y_i^{(1,T+1)} = x_i^{(1,T)} + (r_i - 0,5)\ \Delta_i \tag{2.20}$$

O uso da mutação não-uniforme (2.21), ao contrário da mutação uniforme, tem como principal objectivo criar soluções próximas das soluções progenitoras. Ou seja, a probabilidade da nova solução ser criada perto da solução inicial é maior do que a probabilidade da nova solução ser criada afastada da solução progenitora.

$$y_i^{(1,T+1)} = x_i^{(1,T+1)} + \tau\ (x_i^{(\max)} - x_i^{(\min)})\ (1 - r_i^{(1-\frac{T+1}{T_t})^b}) \tag{2.21}$$

Nesta expressão o parâmetro τ pode tomar os valores $\{-1;1\}$ com a mesma probabilidade. O parâmetro T_t representa o número máximo de gerações enquanto que o parâmetro b é definido pelo utilizador. Assim, a mutação das gerações iniciais têm uma distribuição uniforme, enquanto que a mutação das gerações finais actua como um impulso de Dirac, permitindo assim uma pesquisa localizada.

O método com distribuição normal (2.22) usa a função densidade de probabilidade gaussiana de média zero e desvio padrão σ_i. O parâmetros σ_i, constante, tem um papel importante e deve ser escolhido correctamente de acordo com o problema em questão. O parâmetro também pode ser adaptativo, variando ao longo das gerações ou através de uma regra pré-definida. Este operador é idêntico ao operador de mutação usado nas EEs. Contudo, este último tem um operador de mutação auto-adaptativo, onde o poder da mutação é associado com as variáveis de decisão.

$$y_i^{(1,T+1)} = x_i^{(1,T+1)} + N(0;\sigma_i) \tag{2.22}$$

De forma semelhante ao operador de cruzamento SBX, a função de distribuição de

probabilidade pode ser polinomial (mutação polinomial) (2.23).

$$y_i^{(1,T+1)} = x_i^{(1,T+1)} + (x_i^{(\max)} - x_i^{(\min)})\ \overline{\delta}_i \tag{2.23}$$

Onde o parâmetro $\overline{\delta}_i$ é calculado através da distribuição de probabilidade polinomial (2.24):

$$P(\delta) = 0,5\ (\eta_m + 1)(1 - |\delta|)^{\eta_m} \tag{2.24a}$$

$$\bar{\delta}_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m+1}} - 1, & \text{se } r_i < 0,5; \\ 1 - (2\,(1 - r_i))^{\frac{1}{\eta_m+1}}, & \text{se } r_i \geq 0,5. \end{cases} \tag{2.24b}$$

Esta distribuição é semelhante ao operador de mutação não uniforme, contudo é sugerido um valor fixo do parâmetro η_m. A diferença é que no operador polinomial a forma da densidade de probabilidade é controlada directamente por um parâmetro externo η_m e a distribuição não é modificada dinamicamente com as gerações.

Para os dois últimos operadores, o valor aleatório r_i é gerado por uma função com distribuição de probabilidade que pode ser do tipo uniforme, exponencial, Gaussiana, binomial, ou outra.

Para problemas onde a representação é numérica, a representação dos AGs, em virgula flutuante, tem uma velocidade superior, tem mais consistência, e os resultados são mais precisos do que para o caso da representação binária.

2.6 Mecanismos para preservar a diversidade da população

2.6.1 Introdução

Uma característica inerente dos AGs consiste na perda de diversidade ao longo da sua evolução. Este fenómeno tem como principal problema a convergência prematura o que poderá conduzir o algoritmo para um óptimo local. Quando a convergência do AG ocorre muito rapidamente leva à perda de grande parte da informação genética. Esta perda pode ser retardada ou eliminada através de mecanismos que mantenham a população o mais heterogénea possível. Por outro lado, manter a diversidade na população permite que o AG possa ser transportado para domínios onde seja requerida a identificação e manutenção de múltiplas soluções (*e.g.*, problemas com vários óptimos e problemas com múltiplos objectivos).

Existem vários mecanismos que encorajam a diversidade da população penalizando soluções pertencentes à mesma área de pesquisa. Assim, estes mecanismos podem ser divididos em:

- Aproximações explicitas:
  - Separação geográfica;
  - Especiação.
- Aproximações implícitas (métodos de nicho):
  - Indivíduos similares competem pelos mesmos recursos (método de partilha);
  - Indivíduos similares competem entre si para sobreviver (esquema de agrupamento).

Nesta secção são apresentados os principais métodos para manter a diversidade da população ou para retardar a convergência dos AGs.

2.6.2 Separação geográfica

Para algumas populações, especialmente para populações grandes, não é razoável considerar que um indivíduo pode acasalar com qualquer outro indivíduo à sua escolha. Mesmo o conceito de elitismo é posto em causa, tanto do ponto de vista prático como do ponto de vista biológico. Em populações relativamente pequenas, *i.e.* $pop_{\text{dim}} \leq 750$, a informação necessária para manter o controlo centralizado torna-se pesado. Além disso, populações grandes requerem tempos de execução excessivamente longos, mesmo para um número reduzido de gerações. Assim, esses casos são normalmente implementados em máquinas paralelas, onde o controlo centralizado é evitado tanto quanto possível. Para evitar uma centralização excessiva, é usada uma notação de isolamento por distância. Duas técnicas bastante utilizadas actualmente são:

- O modelo difuso;
- O modelo das ilhas.

O modelo difuso coloca os indivíduos numa grelha (figura 2.10) e só permite acasalamentos entre os vizinhos. Desta forma partes diferentes da grelha podem pesquisar áreas diferentes do espaço de pesquisa. O funcionamento do método para uma geração encontra-se no algoritmo 2.2.

O modelo das ilhas divide a população em vários demes conforme representado na figura 2.11. Cada uma evolui com a sua própria velocidade e direcção existindo uma certa migração entre as ilhas. Esta migração é normalmente efectuada periodicamente ao fim de algumas gerações. Este Algoritmo pode ser implementado

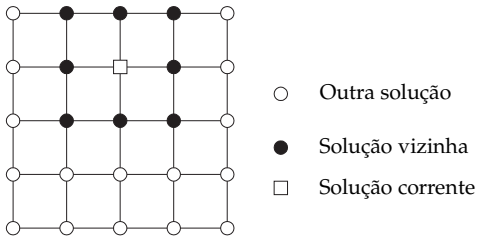


Figura 2.10. Modelo difuso

```
1 repetir de i = 1 até pop_dim
2     Selecciona elemento j;
3     Selecciona vizinho de j;
4     Cria descendente através de cruzamento e mutação;
5     Substitui elemento j;
6 fim repetir
```

Algoritmo 2.2. Algoritmo difuso

usando uma rede de computadores onde a cada computador é afectada uma população ou num computador onde cada população é executada por um processo.

2.6.3 Especiação

Os métodos de especiação restringem os acasalamentos de modo a ocorrerem apenas entre soluções semelhantes (no genótipo ou no fenótipo). Inicialmente, são identificadas as soluções que pertencem aos diferentes picos. De seguida, estas são divididas em subpopulações de acordo com os picos encontrados. Posteriormente, as

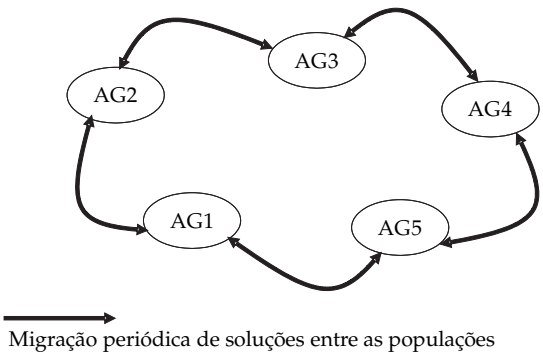


Figura 2.11. Modelo das ilhas

Vector	<Padrão>:<dados>
1	#10#:1010
2	#01#:1100
3	#00#:0000

Figura 2.12. Vectores com acasalamento restrito

subpopulações evoluirão de forma independente criando-se desta forma uma pesquisa paralela.

Dentro deste métodos há a salientar a restrição no acasalamento que será desenvolvida na secção seguinte.

Restrição no acasalamento

Existem vários técnicas que identificam os parceiros com quem um indivíduo pode acasalar [29]:

- Técnica reprodução de linhagem (*linebreeding*): Onde o indivíduo mais apto é o único que pode acasalar com os restantes. Este método é adequado para funções com apenas um pico;
- Técnica reprodução interna com cruzamento intermitente (*inbreeding with intermittent crossbreeding*): somente os indivíduos bastantes chegados podem acasalar entre si. Esta técnica é usada em funções com vários picos;
- Técnica do acasalamento por padrão (*mating templates*): Apenas é permitido o acasalamento entre certos progenitores. Esta técnica é utilizada nas estratégias de evolução.

Para ilustrar como o acasalamento por padrão funciona, conside-se os seguintes vectores:

Os vectores são constituídos por duas partes: o rótulo e a parte de dados. A parte do rótulo (padrão) contém os *bits* de controlo que permite identificar os vectores

com que uma solução pode acasalar. Nos vectores da figura 2.12 o padrão de acasalamento é construído pelo alfabeto {0, 1, #}. Onde, '0' acasala com um '0', '1' acasala com um '1', e '#' acasala com '0' e '1'. Para determinar quais os vectores que podem acasalar, os padrões são comparados com a parte de dados. Podem ser implementadas várias regras de casamento: "casamento bidireccional", "casamento unidireccional" e "melhor casamento parcial".

No casamento bidireccional só é permitido o acasalamento entre dois vectores se o padrão de um vector encaixar na parte de dados do outro e vice-versa. Assim, no exemplo indicado só podem acasalar os vectores 1 (#01#:1100) e 2 (#10#:1010). No casamento unidireccional, basta apenas que um padrão de uma solução verifique a parte funcional do outro vector. No melhor casamento parcial, o vector escolhido para acasalar é o que tem a maior semelhança entre a sua parte de dados e o padrão do primeiro vector.

2.6.4 Métodos de nicho

As técnicas de nicho têm como principal alvo distribuir a população pelos diversos picos do espaço de pesquisa. Desta forma, as técnicas de nicho são incapazes de focalizar a pesquisa dentro de cada pico e terão mais dificuldade em encontrar a solução óptima correspondente a esses picos. Isto deve-se ao esforço gasto na recombinação de soluções de diferentes picos que pode produzir soluções que não pertençam a nenhum dos picos. Nestas técnicas podem-se salientar o método de partilha e o esquema de agrupamento que serão desenvolvidos nas secções seguintes.

Método de partilha

O método de partilha (*sharing*) baseia-se no princípio de que um ambiente contém recursos limitados e que indivíduos fenotipicamente similares devem partilhar esses

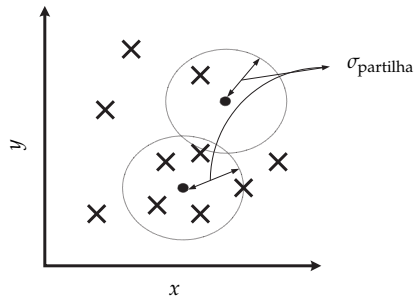


Figura 2.13. Exemplo do método de partilha da aptidão para um espaço bidimensional

recursos (uma ideia razoável do ponto de vista biológico). Desta forma, indivíduos similares sofrem penalidades na sua função de aptidão dependendo da sua similaridade (proximidade) com outros indivíduos da população, figura 2.13. Ou seja, a aptidão partilhada f' de um determinado vector é igual ao seu valor de aptidão dividido pelo seu contador de ninho, n_{c_i}, (2.25a). O contador de ninho determina quantos vectores partilham o mesmo espaço (2.25b). A função de partilha (2.25c) dá uma indicação da proximidade de dois elementos. O parâmetro α regula a forma da função de partilha.

$$f'^{(i)} = \frac{f^{(i)}}{n_{c_i}} \tag{2.25a}$$

$$n_{c_i} = \sum_{j=1}^{pop_{\text{dim}}} d_{\text{part}}(d(i,j)) \tag{2.25b}$$

$$d_{\text{part}}(d_{i,j}) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\sigma_{\text{partilha}}}\right)^{\alpha}, & \text{se } d_{i,j} < \sigma_{\text{partilha}}; \\ 0, & \text{se } d_{i,j} \geq \sigma_{\text{partilha}}. \end{cases} \tag{2.25c}$$

$$d(i,j) = \sum_{k=1}^{l} x_k^{(i)} \oplus x_k^{(j)} \tag{2.25d}$$

$$d(i,j) = \sqrt{\sum_{k=1}^{l} \left(\frac{x_k^{(i)} - x_k^{(j)}}{x_k^{(\max)} - x_k^{(\min)}}\right)^2} \tag{2.25e}$$

$$d(i,j) = \sqrt{\sum_{k=1}^{n_{\text{obj}}} \left(\frac{f_k(\vec{x}_i) - f_k(\vec{x}_j)}{f_k^{\max} - f_k^{\min}}\right)^2} \tag{2.25f}$$

Como se pode ver o método de partilha altera apenas o estágio da atribuição da aptidão. Este tipo de mecanismo pode ser aplicado em qualquer AG.

Contrariamente ao que se passa na natureza, em que a partilha de recursos é aplicada apenas no fenótipo, nos AGs a função $d(.,.)$ pode ser aplicada tanto no genótipo (espaço dos parâmetros) como no fenótipo (espaço dos objectivos). No genótipo a função $d(.,.)$ mede o número de *bits* diferentes (2.25d) ou a distância euclidiana (2.25e), respectivamente para a codificação binária ou real. No caso da função ser aplicada no espaço dos objectivos esta mede a distância euclidiana (2.25f). O parâmetro típico de α é 1 e o valor da distância de partilha, σ_{partilha}, é dependente do problema mas deve ter um valor suficientemente pequeno de modo a poder diferenciar soluções de picos diferentes. Esta técnica é bastante utilizada com o método da selecção proporcional. Contudo, na selecção por torneio deve ter-se especial atenção de modo a promover a estabilidade [46]. Uma desvantagem deste método deve-se ao facto de ser necessário especificar o valor da distância de partilha σ_{partilha}.

Esquema de agrupamento

O esquema de agrupamento (*crowding*) é usado entre gerações, onde os novos indivíduos vão substituir aqueles que genotipicamente são similares com estes (em alternativa à substituição dos indivíduos menos aptos). Para prevenir um número impraticável de comparações que seriam necessárias quando um indivíduo é criado, indivíduos recém-nascidos vão ser comparados apenas a um certo número (factor de agrupamento, FA) de indivíduos.

Para determinar a similaridade entre as soluções é usada a função distância referida na secção anterior. Os métodos de agrupamento tendem a dispersar os vectores ao longo dos picos mais salientes.

2.6.5 Prevenção de clones

A prevenção de clones ajuda a evitar a convergência prematura na evolução artificial. Eshelman [31] sugeriu a prevenção de incesto, nomeadamente onde apenas progenitores geneticamente diferente podem procriar, a fim de prevenir a formação de clones. No entanto, existe o problema de saber até que ponto dois progenitores são diferentes. A estratégia de Eshelman foi a de calcular a distância de Hamming entre os progenitores escolhidos. Se a distância for superior a um dado valor fixo então os progenitores podem acasalar. Todavia, à medida que o algoritmo converge os progenitores ficam mais semelhantes entre si e o número de acasalamentos rejeitados aumenta. Para contrariar este processo a distância de Hamming pode ser relaxada ligeiramente ao longo da evolução de modo a evitar que a taxa de acasalamentos rejeitados aumente. Este método reduz significativamente a criação de clones com baixo custo.

2.7 Resumo

Neste capítulo foram revistos alguns conceitos dos AEs que serão usados nos capítulos seguintes. Assim, foram apresentados os conceitos básicos da selecção natural, foram descritos os aspectos fundamentais dos AGs e os mecanismos para preservar a diversidade da população.

3 Algoritmos evolutivos multi-objectivo

3.1 Introdução

A optimização multi-objectivo ou multi-critério surge em problemas onde se pretende optimizar mais do que um objectivo. Na maioria dos problemas reais é necessário tomar decisões que envolvem frequentemente múltiplas medidas de desempenho, ou vários objectivos, que devem ser optimizados simultaneamente. Na prática, raramente é conseguido, pois alguns dos objectivos são conflituosos. Além do mais, as escalas dos múltiplos objectivos dificilmente se encontram harmonizadas. Assim, a qualidade de uma solução é expressa melhor, não por um escalar, mas por um vector de valores, onde cada elemento do vector representa o desempenho de um objectivo.

Existe um certo número de diferenças fundamentais entre problemas uni-objectivo e multi-objectivo. Em problemas de optimização com apenas um objectivo a tarefa baseia-se em determinar a solução que optimiza esse objectivo. Extendendo a ideia a problemas multi-objectivo, é erróneo considerar que a tarefa de optimizar um problema multi-objectivo consiste em encontrar a solução correspondente ao óptimo de todos os objectivos.

Devido à falta de metodologias adequadas, a optimização de problemas multi-objectivo (OPMO), tem sido frequentemente adaptada e resolvida como problemas de optimização de um único objectivo. Neste caso o problema multi-objectivo é transformado em uni-objectivo através de uma função agregada (3.1). Para este efeito é executado o algoritmo diversas vezes variando os pesos β_i para se obter um conjunto de soluções não-dominadas. Este processo tem o inconveniente de obter apenas uma solução por cada execução e, ao usar um espaçamento uniforme dos pesos β_i, raramente obtém uma distribuição uniforme das soluções ao longo do espaço dos n_{obj} objectivos.

$$z = f_{MO} = \sum_{k=1}^{n_{\text{obj}}} \beta_k f_k(x), \quad x \in \mathbb{R}^N \tag{3.1}$$

Uma diferença fundamental entre os métodos de optimização clássicos[1] e os algoritmos evolutivos multi-objectivo (AEMO) tem a ver com o facto de nestes ser usada uma população de soluções em cada iteração. Notar que um dos principais objectivos dos algoritmos multi-objectivos é encontrar tantas soluções não-dominadas quanto possível. Desta forma, pode tirar-se partido da população de soluções dos AEs e introduzindo-lhes pequenas alterações com o intuito de obter um conjunto de soluções não-dominadas. Consequentemente, é eliminada a necessidade de repetir o método de optimização usando uma combinação de objectivos, como medida de desempenho, por cada execução do algoritmo. Assim, este processo elimina a necessidade de usar certos parâmetros como os pesos dos vectores, β_i e, consequentemente, simplifica a tarefa de procura das soluções não-dominadas. Por outro lado, a população do AE pode ser explorada de modo a manter as soluções não-dominadas e preservar, simultaneamente, um conjunto diverso das mesmas usando operadores que promovam e preservam nichos genéticos (ver secção 2.6.4). Após algumas gerações, e dependendo do problema, este processo pode levar a população a convergir para a frente óptima de Pareto, onde cada solução da população representa

[1] *e.g.* equivalente paramétrico, equivalente por restrições, Simplex multi-objectivo.

uma solução não-dominada.

A eficiência de um AE multi-objectivo é medida pelas características seguintes [47]:

- A distância entre a frente não-dominada encontrada pelo algoritmo e a frente óptima de Pareto: deve ser minimizada.

- A distribuição das soluções não-dominadas ao longo da frente: deve ser uniforme.

- A extensão da frente não-dominada: deve ser maximizada (*i.e.*, cada objectivo deve ser coberto por um vasto número de soluções não dominadas).

Mesmo nos casos onde se pretende obter apenas uma solução óptima, a frente óptima de Pareto permite auxiliar a escolher a solução mais adequada ao problema. O agente de decisão pode não ter a certeza da relação exacta dos pesos que deve atribuir a cada objectivo de modo a obter a solução pretendida. Neste cenário, de um ponto de vista prático, é melhor encontrar primeiro a frente de Pareto e depois escolher uma solução do conjunto final pelo uso de informação ou considerações de nível superior.

Este capítulo está organizado em 6 secções. Assim, na secção 3.2 discute-se o princípio da optimização multi-objectivo e analisam-se as condições necessárias para uma solução ser óptima de Pareto quando em presença de múltiplos objectivos. De seguida são apresentados os AEMO. Na secção 3.4 são descritos os AEMO que preservam a elite. Posteriormente, na secção 3.5 são formulados alguns índices com o fim de avaliar o desempenho dos métodos multi-objectivo. Por último, na secção 3.6 incluí-se um resumo do capítulo.

3.2 Problemas multi-objectivo

Um Problema multi-objectivo (PMO), como o próprio nome indica, tem mais do que uma função objectivo para optimizar. De forma semelhante aos problemas com apenas um objectivo, os problemas PMO também têm um conjunto de restrições que qualquer solução admissível deve satisfazer. As seguintes equações [47] especificam de modo geral um PMO.

$$\begin{array}{lll} \text{Maximizar/Minimizar:} & f_m(x), & m = 1, \ldots, n_{\text{obj}}; \\ \text{Sujeito a:} & g_j(x) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(x) = 0, & k = 1, 2, \ldots, K; \\ & x_n^{(I)} \leq x_n \leq x_n^{(S)}, & n = 1, 2, \ldots, N. \end{array}$$

A solução x é um vector de N variáveis de decisão: $x = (x_1, x_2, ..., x_N)^T$. A última inequação representa o conjunto de restrições que são denominadas por variáveis fronteira, restringindo cada variável de decisão a tomar um valor entre o valor inferior $x_n^{(I)}$ e o valor superior $x_n^{(S)}$ do intervalo. Estes limites constituem o espaço das variáveis de decisão D ou simplesmente o espaço de decisão. Ao longo da tese os termos: *string*, vector, solução e indivíduo são usados para indicar o vector x. Associado ao problema, existem J inequações e K igualdades. Os termos $g_j(x)$ e $h_k(x)$ são denominados por funções de restrição. As restrições menores ou iguais a zero podem ser transformadas em restrições maiores ou iguais a zero multiplicando as inequações por -1. Uma solução, x, que não satisfaça todas as restrições é denominada de não-admissível. Por outro lado, se a solução, x, satisfaz todas as restrições e as variáveis limite é chamada de admissível. Assim, na presença de restrições o espaço de decisão pode não ser todo ele admissível. Ao espaço admissível denomina-se por região admissível, S. Existem n_{obj} funções objectivo $f(x) = (f_1(x), f_2(x), ..., f_{n_{\text{obj}}}(x))^T$ consideradas na formulação anterior. Cada função objectivo pode ser maximizada ou minimizada.

De seguida são expostas algumas definições [48, 49] que são úteis na descrição dos algoritmos apresentados neste capítulo. As definições consideram que os problemas

são de minimização.

Definição 3.1 (Relação de dominância)**.** *Sejam $x, y \in \mathbb{R}^{n_{obj}}$ duas soluções. Diz-se que x domina y, ou seja, $x \succ y$, se e só se:*

1. $\forall_i \in \{1, \ldots, n_{obj}\} : f_i(x) \leq f_i(y)$
2. $\exists_j \in \{1, \ldots, n_{obj}\} : f_j(x) < f_j(y)$

Definição 3.2 (Solução óptima de Pareto ou não-dominada)**.** *Uma solução $x \in \mathbb{R}^{n_{obj}}$ diz-se óptima de Pareto se e só se:*

$$\nexists y \in \mathbb{R}^{n_{obj}} : y \succ x$$

Definição 3.3 (Conjunto de Pareto)**.** *Seja $F \subseteq \mathbb{R}^{n_{obj}}$ um conjunto de vectores. Então o conjunto de Pareto F^* do conjunto F é definido como: F^* contém todos os vectores $x \in F$ que não são dominados por qualquer vector $y \in F$, i.e.,*

$$F^* = \{x \in F | \nexists y \in F : y \succ x\}$$

Os vectores de F^* são denominados de vectores óptimos de Pareto. O conjunto de todos os vectores óptimos de Pareto é definido como $P^*(F)$. Da definição pode ser deduzido que qualquer vector $y \in F \backslash F^*$, i.e.,

$$\forall y \in F \backslash F^* : \exists x \in F^* : x \succ y$$

Para um dado conjunto F, o conjunto F^* é único. Consequentemente $P^*(F) = \{F^*\}$

Definição 3.4 (Dominância-ϵ)**.** *Sejam $x, y \in \mathbb{R}^{n_{obj}}$. x domina y com margem ϵ, para qualquer $\epsilon > 0$, denotado por $x \succ_\epsilon y$, se e só se para todos os $i \in \{1, \ldots, n_{obj}\}$*

$$(1 - \epsilon) f_i(x) \leq f_i(y)$$

Definição 3.5 (Conjunto de Pareto aproximado-ϵ)**.** *Seja $x \in \mathbb{R}^{+n_{obj}}$ um conjunto de vectores e $\epsilon > 0$. Então o conjunto F_ϵ é denominado por conjunto de Pareto aproximado-ϵ, se qualquer vector $y \in F$ é dominado por pelo menos um vector de $x \in F_\epsilon$, i.e.,*

$$\forall y \in F : \exists x \in F_\epsilon : x \succ_\epsilon y$$

O conjunto de todos os conjuntos de Pareto aproximado-ϵ é definido por $P_\epsilon(F)$.

Definição 3.6 (Conjunto de Pareto-ϵ)**.** *Seja $F \in \mathbb{R}^{+n_{obj}}$ um conjunto de vectores e $\epsilon > 0$. Então o conjunto $F^*_\epsilon \subseteq F$ é denominado por conjunto de Pareto-ϵ de F se e só se:*

1. *F^*_ϵ é um conjunto de Pareto aproximado-ϵ, i.e., $F^*_\epsilon \in P_\epsilon(F)$, e*
2. *F^*_ϵ contém apenas pontos de Pareto de F, i.e., $F^*_\epsilon \subseteq F^*$.*

O conjunto de todos os conjuntos de Pareto-ϵ de F é designado por $P^*_\epsilon(F)$ e $P^*_\epsilon(F) \subseteq P_\epsilon(F)$

3.3 Algoritmos evolutivos multi-objectivo

Nesta secção apresenta-se uma introdução aos AEMO mais importantes, quer por razões históricas quer pela popularidade que ganharam nos últimos anos. Notar que nesta secção só se apresentam algoritmos sem elitismo.

Schaffer [50] implementou o primeiro AGMO para determinar um conjunto de soluções não-dominadas. O algoritmo usa como medida de desempenho um vector, em vez de usar um escalar, onde cada elemento do vector representa o desempenho relativo a um objectivo. O algoritmo VEGA (*Vector Evaluated Genetic Algorithm*) é simplesmente uma extensão do AG clássico para problemas de optimização multi-objectivo. De facto, uma vez que são considerados n_{obj} objectivos, Schaffer divide aleatoriamente, em cada geração, a população em n_{obj} subpopulações. De seguida, a cada elemento de uma subpopulação é atribuída a aptidão baseada apenas na função objectivo correspondente. A selecção, o cruzamento e a mutação são processados de forma independente de acordo com a subpopulação a que pertencem. Posteriormente as soluções vencedoras de cada subpopulação são agrupadas e o algoritmo prossegue para a nova iteração. O Algoritmo VEGA tem a tendência para encontrar soluções próximas dos óptimos de cada objectivo. Contudo, não é conseguida uma frente de Pareto uniforme, ou seja, as soluções não se encontram uniformemente distribuídas ao longo da frente óptima de Pareto [47]. Mesmo quando se cruzam indivíduos próximos dos diferentes objectivos constata-se que o AG é incapaz de manter a diversidade, convergindo apenas para os óptimos de cada objectivo.

Hajela *et al.* [51] propuseram um AG baseado em pesos (WBGA: *Weight-Based Genetic Algorithm* também conhecido pelas iniciais dos autores: HLGA), onde a cada indivíduo é atribuído um vector de pesos. Desta forma, a população do AG mantém simultaneamente múltiplos vectores pesos e, consequentemente, permite encontrar várias soluções óptimas de Pareto em apenas uma execução. Assim, para que o

algoritmo tenha um bom desempenho deve manter uma boa diversidade entre os vectores pesos dos indivíduos da população. No algoritmo WBGA, a diversidade pode ser mantida de duas formas:

1. É usado um método de nicho apenas no vector que representa os pesos;
2. São definidas várias subpopulações de acordo com os vectores peso pré-definidos.

Qualquer optimização baseada em pesos, em princípio não consegue encontrar os óptimos de Pareto em regiões não convexas [45]. O algoritmo WBGA usa a selecção proporcional nos valores de aptidão partilhados. Assim, as funções objectivo devem ser transformadas de modo a tornar os objectivos em funções a maximizar. Por outro lado, se se pretender uma distribuição uniforme de soluções não-dominadas torna-se difícil escolher o valor dos pesos, pois uma distribuição uniforme de pesos não leva necessariamente a um conjunto de soluções uniformemente distribuídas. Embora o método WBGA seja mais simples em termos de complexidade do que os métodos que recorrem à função de partilha, pois não é requerida a métrica de distância, é necessário a definição, *a priori*, do vector de pesos. Além do mais, é necessária memória adicional para guardar os vectores peso. Como os operadores do AG são aplicados de acordo com cada subpopulação é necessário um certo número mínimo de elementos para encontrar o óptimo de cada subpopulação. Consequentemente, o número de soluções da população terá um tamanho considerável.

Goldberg [29] sugeriu como deve ser elaborado um AGMO para obter um bom desempenho. Assim, sugeriu o uso do conceito de dominância de modo a atribuir um número maior de cópias a indivíduos não-dominados da população. Adicionalmente, Goldberg propôs também o uso da estratégia de nicho (*niching*) sobre as populações não-dominadas, uma vez que a diversidade da população é crucial para a eficácia do algoritmo. No início, foi proposto um vasto número de AEMO dos quais se destacaram os seguintes: *Multi-Objective GA* (MOGA), *Niched Pareto GA*

(NPGA) e *Non-dominated Sorting Genetic Algorithm* (NSGA). Os algoritmos diferem essencialmente no modo como o mérito é atribuído a cada indivíduo.

Fonseca e Flemming [52] propuseram um algoritmo multi-objectivo que denominaram por algoritmo genético multi-objectivo (*Multi-Objective Genetic Algorithm* – MOGA). Este algoritmo foi o primeiro a usar conceitos de população não-dominada, a realçar explicitamente as soluções não-dominadas, promovendo simultaneamente a diversidade entre as soluções não-dominadas. O algoritmo MOGA difere do AG clássico no modo como calcula a aptidão. O resto do algoritmo é idêntico (selecção estocástica universal, cruzamento de ponto simples e mutação por lugar (*bitwise*)).

O algoritmo que atribui o valor de aptidão é o seguinte:

1. Atribuir o posto às soluções, considerando o número de soluções que dominam a solução $+1$ (ver figura 3.1).

2. Ordenar as soluções pelo posto, atribuindo valores inteiros consecutivos de 1 a pop_{dim} para as posições da ordenação descendente.

3. O valor de aptidão é calculado pela média das posições de todas as soluções que pertencem a esse posto.

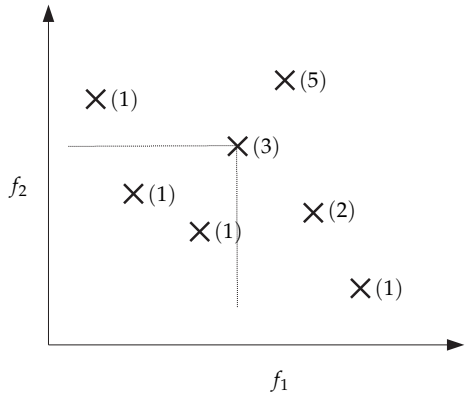


Figura 3.1. Atribuição do valor de aptidão de acordo com o número de soluções dominantes (considera-se que o problema é de minimização), com dois objectivos: f_1 e f_2

O valor de aptidão também pode ser calculado através da fórmula (3.2) onde $\mu(r_i)$ representa o número de soluções do posto r_i.

$$f_i = pop_{\text{dim}} - \sum_{k=1}^{r_i-1} \mu(k) - 0.5(\mu(r_i) - 1) \tag{3.2}$$

Para manter a diversidade das soluções não-dominadas Fonseca e Flemming introduziram uma técnica de nicho entre as soluções de cada posto. Assim, a função de partilha (2.25a) usa $\alpha = 1$ e a distância normalizada entre duas soluções do mesmo posto é representado pela equação (3.3) onde $f_k^{\max}$ e $f_k^{\min}$ são, respectivamente os valores máximos e mínimos da função objectivo k. Para a solução i é calculada a distância d_{ij} para qualquer solução j do posto a que a solução pertence.

$$d_{ij} = \sqrt{\sum_{k=1}^{n_{\text{obj}}} \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{\max} - f_k^{\min}} \right)^2} \tag{3.3}$$

Sendo o valor do contador de nicho obtido pela fórmula (3.4), onde $\mu(r_i)$ é o número de soluções do posto r_i.

$$n_{c_i} = \sum_{j=1}^{\mu(r_i)} d_{\text{part}}(d_{ij}) \tag{3.4}$$

Por fim, o valor de aptidão partilhado (f_i') é calculado pela equação (3.5).

$$f_i' = f_i / n_{ci} \tag{3.5}$$

Desta forma, dentro do mesmo posto, às soluções mais dispersas no espaço dos objectivos são-lhes atribuídos valores de aptidão superiores em relação às soluções mais concentradas. Por outras palavras, as soluções menos representadas num determinado posto são beneficiadas. Uma vez que a técnica de nicho é aplicada no espaço dos objectivos, o algoritmo MOGA pode ser facilmente aplicado em qualquer

problema de optimização, tais como em problemas de optimização combinatória. Apesar do conceito de dominância, usado para calcular o desempenho, os valores de aptidão das soluções dentro do mesmo posto podem não ter valores iguais, o que pode originar uma pesquisa direccional numa determinada região de pesquisa. O procedimento do cálculo da função de partilha não garante que uma solução de posto inferior apresente um valor superior a uma outra que tenha posto superior.

A ideia proposta por Goldberg [29], de usar o conceito de ordenamento não-dominado, foi implementada mais fidedignamente por Srinivas e Deb [53], no algoritmo genético de ordenação não dominada (*Non-dominated Sorting Genetic Algorithm* – NSGA). Os objectivos são mantidos usando um processo de optimização que favorece soluções não-dominadas e pelo uso da técnica de nicho que preserva a diversidade entre as soluções da mesma frente. O primeiro passo consiste em ordenar as soluções de acordo com o posto baseado em frentes não-dominadas. Inicialmente, retiram-se todas as soluções não-dominadas da população e atribui-se-lhes o posto 1. De seguida, repete-se o processo e atribui-se-lhes o posto 2. O processo termina quando forem retiradas todas as soluções da população (ver figura 3.2).

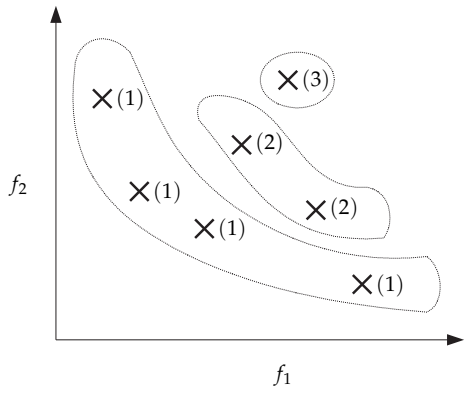


Figura 3.2. Atribuição do valor de aptidão de acordo com a frente não-dominada (considera-se que o problema é de minimização)

Após concluir esta fase, cada solução do posto não-dominado toma o valor de aptidão de $f_{\mu(1)} = pop_{\dim}$, sendo depois penalizada de acordo com a concentração de soluções, próximas, pertencentes ao posto em questão. O valor de aptidão do

posto seguinte difere ϵ (normalmente $\epsilon = 0{,}22$) unidades do pior valor de aptidão encontrado até ao momento ($f_{\mu(i+1)} = \min\{f_{\mu(i)}\} - \epsilon$). Determinado o valor deste novo posto, são consideradas as respectivas soluções e o processo repete-se. O algoritmo termina quando forem atribuídos os valores a todas as soluções. Desta forma é garantido que o valor de aptidão entre duas soluções de postos consecutivos difira pelo menos de um valor ϵ. Assim, consegue-se direccionar a pesquisa na direcção da frente de Pareto. Contrariamente ao algoritmo MOGA, a distância d_{ij} é calculada no espaço dos atributos em vez do espaço dos objectivos, utilizando a expressão (3.6), com $\alpha = 2$ e $f_i' = f_i / n_{ci}$. Adicionalmente, a distância d_{ij} considera apenas as soluções da frente, P_n, a que a solução i pertence. Por outro lado, a técnica de nicho aplicada ao espaço dos parâmetros é favorável à emergência de soluções fenotipicamente diferentes.

$$d_{ij} = \sqrt{\sum_{k=1}^{\#P_n} \left(\frac{x_k^{(i)} - x_k^{(j)}}{x_k^{\max} - x_k^{\min}} \right)^2} \tag{3.6}$$

Horn *et. al.* [54] propuseram um método multi-objectivo (*Niched Pareto Genetic Algorithm* – NPGA) baseado no conceito de dominância. A diferença deste algoritmo em relação aos anteriores reside no operador de selecção que usa o torneio binário ao contrário da selecção proporcional.

No algoritmo NPGA a estratégia de nicho é actualizada dinamicamente. Durante a selecção, duas soluções e uma subpopulação são escolhidas aleatoriamente. De seguida, as soluções i e j são comparadas com todas as soluções da subpopulação para testar a dominância. Se uma das soluções domina todas as soluções da subpopulação e a outra solução, então esta é escolhida como a vencedora. Se ambas as soluções i e j são dominadas ou não-dominadas, então o contador de nicho, n_{c_i}, é calculado para ambos os vectores em relação aos elementos da subpopulação corrente, de modo a determinar a concentração de soluções na sua vizinhança. A solução com o menor contador, n_{c_i}, ganha o torneio. Inicialmente, quando a população se

encontra vazia é escolhida uma solução do torneio aleatoriamente. A distância d_{ik} é calculada no domínio dos objectivos. O algoritmo NPGA não necessita de explicitar a atribuição da aptidão. Do ponto de vista computacional este algoritmo não depende do número de objectivos, uma vez que só calcula a função de aptidão de uma subpopulação. O parâmetro de partilha, σ_{partilha}, desempenha um papel importante no algoritmo NPGA, pois o valor do contador de nicho n_{c_i} da solução i é calculado como o número de vectores a uma distância σ_{partilha} da solução i. O desempenho do algoritmo NPGA depende do número de soluções da subpopulação.

Laumanns *et. al.* [55] desenvolveram uma estratégia evolutiva multi-objectiva (Estratégia Evolutiva Predador-Presa) que é totalmente diferente dos métodos anteriores. O cálculo do valor de aptidão não depende da dominância da solução. No entanto, é usado o conceito de predador-presa em sua substituição. As presas representam um conjunto de soluções $(x^{(i)}, i = 1, 2, ..., pop_{\text{dim}})$ que são colocadas nos vértices de um grafo não direccional. Inicialmente, cada predador é colocado aleatoriamente num dos vértices do grafo. A cada predador é-lhe associado uma função objectivo. Durante a evolução do algoritmo, os predadores olham em sua volta (*i.e.* nos vértices vizinhos) por presas. A presa vizinha, $x^{(i)}$, com o menor valor objectivo que o predador representa é morta e removida do vértice. Sendo esta substituída pela mutação (e recombinação) de uma presa aleatória da vizinhança de $x^{(i)}$. Por fim, o predador desloca-se aleatoriamente para um dos seus vértices vizinhos. Este processo ocorre em paralelo ou sequencialmente com todos os predadores até prefacer as iterações necessárias. O algoritmo é bastante simples de implementar, não utiliza conceitos de dominância nem usa operadores que promovam a diversidade no conjunto das soluções não-dominadas.

3.4 Algoritmos evolutivos com operadores que preservam a elite

Os Algoritmos descritos na secção anterior ilustram formas diferentes de implementar AEMO garantindo um certo êxito, especialmente na atribuição do valor de aptidão de acordo com a sua dominância e na preservação da diversidade da população no decorrer da evolução. Contudo, para assegurar a convergência das soluções para a verdadeira frente de Pareto devem ser implementados operadores que preservem as melhores soluções ou elite [48].

Assim, nos AEs, os operadores que preservam a elite favorecem os melhores elementos da população na passagem para a geração seguinte. Na maioria dos AEs mono-objectivo, as melhores α soluções da população são consideradas como elites. Desta forma, a escolha de α é bastante importante no sucesso do algoritmo. Este parâmetro está directamente relacionado com a pressão de selecção associado às elites, uma vez que estas passam para a geração seguinte e participam nas operações genéticas, pelo que tendem a influenciar a população a crescer à volta delas. Se é usado um valor de α elevado, a influência da elite é maior e a população perde diversidade. Por outro lado, se é usado um valor muito baixo, não se tira vantagem adequada do elitismo. Os valores típicos de α encontram-se no intervalo $[1; 0{,}1pop_{\text{dim}}]$.

Nos algoritmos mono-objectivos a elite é identificada facilmente. Contudo nos algoritmos multi-objectivo esta identificação não é directa. Consequentemente, recorre-se a técnicas de ordenamento das soluções de acordo com a sua dominância (*non-domination ranking*). Neste caso, o parâmetro α, não indica o número de vectores que transitam para a próxima iteração, mas o posto das soluções que passam para a geração seguinte (*e.g.*, se $\alpha = 2$ os vectores com posto 1 e 2 passam para a geração seguinte). Usando este processo, o número de elementos transportados entre os ciclos é indeterminável.

No contexto da optimização mono-objectivo, tem-se constatado que a presença de

operadores de preservação da elite favorece o AG a ter uma melhor convergência. Nesta secção será apresentado um conjunto de AGMO, com diversos tipos de operadores que preservam a elite com vista a melhorar a convergência das soluções óptimas de Pareto. A quantidade de algoritmos apresentados nesta secção mostra formas diferentes de implementar o elitismo nos AGMO.

Rudolph [56] propôs um algoritmo evolutivo multi-objectivo que usa elitismo. O algoritmo considera λ descendentes, na população Q_T, e μ progenitores na população P_T, seguindo os seguintes pontos:

- Os elementos não-dominados da população Q_T passam para a população Q^*.
- Cada solução q de Q^* é comparada com a população P_T. Todas as soluções dominadas pela solução q serão removidas da população P_T. Se a solução q eliminar pelo menos uma solução de P_T é colocada na população P'. Caso contrário, e se a solução q não for dominada por qualquer solução de P_T, é colocada na população Q' (a solução está ao mesmo nível que as soluções de P_T).
- A população final será constituída pelas populações P_T e P'. Contudo, se o número de elementos da população final for inferior à dimensão desejada, esta é preenchida por elementos das populações Q', Q^* e Q_T seguindo esta ordem.

Este algoritmo converge mas pode não manter a diversidade entre as soluções óptimas de Pareto. Sempre que todas as μ soluções progenitoras forem óptimas de Pareto, não é possível acomodar mais nenhuma solução na nova população. O algoritmo pode ser melhorado pelo uso de um esquema de nicho para seleccionar os elementos da população Q', Q^* e Q_T a fim de preencher a população com um conjunto diverso de soluções destes conjuntos.

Deb *et. al.* [47] desenvolveram um algoritmo (NSGA–II: *Non-dominated Sorting GA*) que usa um mecanismo de preservação de elitismo. De forma análoga ao algoritmo

de Rudolph, a população de descendentes Q_T é gerada através da população progenitora P_T. Uma terceira população, R_T, é criada a partir da reunião das populações $Q_T \cup P_T$ e ordenada segundo o algoritmo NSGA. Uma nova população, P'_T é preenchida de acordo com o posto das soluções que é baseado em frentes não-dominadas. Desta forma, ao preencher a população P'_T, as soluções da frente 1, 2, *etc.*, têm preferência segundo esta ordem. Quando a próxima frente, a ser inserida na nova população, exceder o número de lugares vagos na população, em vez de se descartar os elementos excedentes é usada uma estratégia de nicho pombalina (*crowding algorithm*) para escolher os membros desta última frente que favoreça as soluções mais dispersas (ver figura 3.3). Esta técnica é especialmente relevante na parte final da execução, quando a maior parte das soluções são não-dominadas. Desta forma, o algoritmo garante a diversidade das soluções nesta etapa. O peso computacional do algoritmo é de $O(pop_{\text{dim}} \log pop_{\text{dim}})$.

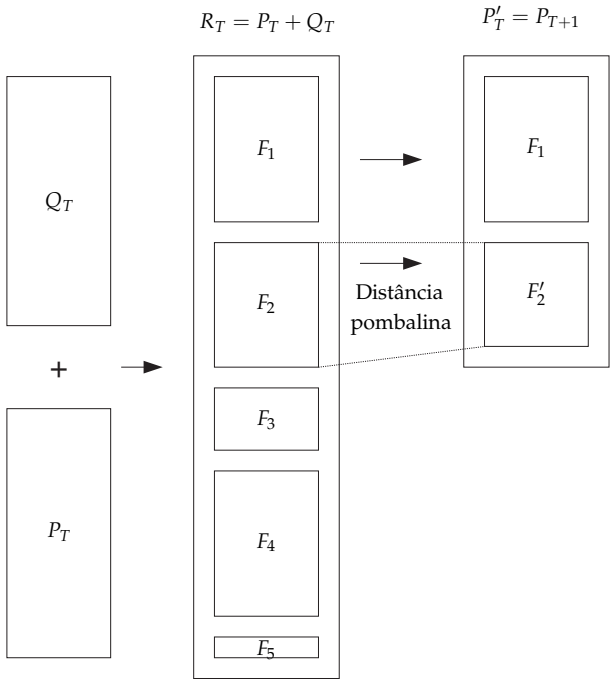


Figura 3.3. Processo de selecção do algoritmo NSGA–II

De seguida é descrita a selecção por torneio pombalino (*crowded tournament*) e a distância pombalina (*crowding distance*), usada no algoritmo NSGA-II, para apurar as soluções que preencherão os restantes lugares da população. A selecção por torneio

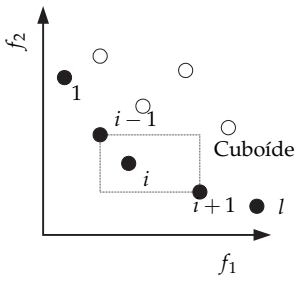


Figura 3.4. Cálculo da distância pombalina

pombalino é efectuada entre duas soluções i e j. A solução i ganha o torneio se o posto da solução i é menor do que o posto da solução j ($r_i < r_j$) ou se têm o mesmo posto e se a solução i tem uma distância pombalina superior à solução j ($r_i = r_j$ & $d_i > d_j$).

A distância pombalina é usada para ter uma estimativa da densidade das soluções na vizinhança da solução i. Este valor é calculado através da distância pombalina normalizada entre as soluções mais próximas, uma de cada lado, à solução i em todos os objectivos. O valor d_i serve como estimativa do perímetro formado pelo cubóide (figura 3.4) usando os vizinhos, um de cada lado, mais próximos como vertices.

Procedimento para calcular a distância pombalina:

1. l = número de soluções do posto f, $l = \#F$ para cada elemento no conjunto $d_i = 0$.

2. Para cada função objectivo ($m = 1, 2, \ldots, n_{\text{obj}}$) ordenar o conjunto por ordem descendente de f_m, encontrar os índices ordenados dos vectores: $I^m = ordenar(f_m, >)$.

3. Para $m = \{1, \ldots, n_{\text{obj}}\}$ atribuir uma distância elevada às soluções periféricas (limites) $dI_1^m = dI_l^m = \infty$ e para as soluções restantes usar a equação (3.7), onde I_k^m é a $k = 1 \ldots l^{\text{éssima}}$ solução ordenada de acordo com o objectivo n_{obj}.

$$dI_j^m = dI_j^m + \frac{f_m^{\left(I_{j+1}^m\right)} - f_m^{\left(I_{j-1}^m\right)}}{f_m^{(\max)} - f_m^{(\min)}} \tag{3.7}$$

Uma vez que as soluções competem através da sua distância pombalina (uma medida que dá uma indicação da densidade de soluções na vizinhança) não é necessário especificar qualquer outro parâmetro, tal como o parâmetro σ_{partilha}. A distância pombalina é uma estimativa da distância euclidiana quando a dimensão do espaço dos objectivos é pequena. No entanto, à medida que a dimensão aumenta a distância pombalina perde precisão e, consequentemente, o algoritmo tem dificuldade em encontrar soluções uniformemente dispersas.

Osyczka e Kundu [57] sugeriram um AG elitista (DPGA: *Distance-based Pareto Genetic Algorithm*), baseado na distância euclidiana, que pretende realçar o progresso na direcção da frente óptima de Pareto e a diversidade das soluções obtidas usando um determinado critério. O algoritmo lida com duas populações: a população do AG, P_T, e uma outra população, denominada por arquivo, E_T, onde são mantidas as melhores soluções encontradas até ao momento. Inicialmente, uma população, P_0, de pop_{dim} indivíduos é gerada aleatoriamente. Ao primeiro elemento é-lhe atribuído um valor de aptidão aleatório, F_1, e é automaticamente adicionado ao arquivo E_0. De seguida, o valor de aptidão das soluções é calculado baseado na distância ao conjunto da elite, $E_T = \{e^{(k)} : k = 1, 2, \ldots, K\}$, onde K é o número de soluções da população E_T. Cada solução do arquivo, $e^{(k)}$, tem n_{obj} valores ou $e^{(k)} = \left(e_1^{(k)}, e_2^{(k)}, \ldots, e_{n_{\text{obj}}}^{(k)}\right)$. A distância euclidiana da solução x a um elemento da elite é calculado pela seguinte expressão:

$$d^{(k)}(x) = \sqrt{\sum_{m=1}^{n_{\text{obj}}} \left(\frac{e_m^{(k)} - f_m(x)}{e_m^{(k)}}\right)^2} \tag{3.8}$$

Por fim, para a solução x, a distância ao conjunto da elite é dado pelo valor mínimo $d^{(k)}(x)$ de todos os $k = 1, 2, \ldots, K$, que é encontrado através da expressão:

$$d^{\min}(x) = \min_{k=1}^{K} d^{(k)}(x) \tag{3.9}$$

O parâmetro k^* identifica a solução do conjunto da elite cuja distância à solução é menor. Se a solução x é não-dominada em relação ao conjunto da elite, então é aceite neste conjunto. Sendo o valor de aptidão de x calculado pela adição do valor de aptidão do membro da elite com distância menor a x e a distância entre as duas soluções ($F(x) = F(e^{(k^*)}) + d^{\min}(x)$). O arquivo é então actualizado eliminando os elementos dominados por esta solução. Por outro lado, se a solução x é dominada por uma solução do arquivo, a solução é calculada pela fórmula $F(x) = \max[0, F(e^{(k^*)}) - d^{\min}(x)]$. Deste modo, à medida que os membros da população são avaliados o conjunto da elite é actualizado. No fim de cada geração é atribuído a todas as soluções do arquivo E_T o valor de aptidão máximo encontrado nessa população. Após este processo estar concluído é criada uma nova população através dos operadores de selecção, cruzamento e mutação. É interessante notar que as soluções não-dominadas, com uma distância grande ao arquivo E_T, obtêm uma aptidão melhor. Consequentemente, no caso da solução dominar alguns membros do conjunto da elite, o valor de aptidão que lhe é atribuído ajuda a realçar soluções perto do conjunto óptimo de Pareto. Uma solução com uma grande distância significa uma solução distante do conjunto da elite existente mas perto da frente óptima de Pareto. Atribuindo um valor de aptidão elevado a estas soluções ajuda-se o deslocamento das soluções no sentido da frente óptima de Pareto. Por outro lado, se a nova solução incide na mesma frente, não-dominada, que as soluções do conjunto da elite, então a atribuição da aptidão ajuda a manter a diversidade entre elas. Neste caso a distância da solução indica o grau de isolamento. Neste sentido, atribuindo um valor de aptidão grande a uma solução "isolada" promove a diversidade entre as soluções não-dominadas. Neste método, tanto as metas de progressão no sentido do óptimo da frente de Pareto, como a manutenção da diversidade das soluções, são desempenhadas sem auxílio a qualquer método de nicho. Todavia, como o número

de elementos do arquivo E_T não é constante, a população pode crescer indefinidamente. Assim, esse número pode ser especificado e uma solução só é inserida na população se ainda existir um lugar vago.

Zitzler e Thiele [58] propuseram um algoritmo evolutivo que chamaram de *Strenght Pareto Evolutionary Algorithm* – SPEA. Este algoritmo introduz o elitismo mantendo explicitamente uma população externa, E (arquivo). Esta população externa guarda todas as soluções não-dominadas que foram encontradas até ao momento. Por outras palavras, em cada geração, as novas soluções não-dominadas da população $P(T)$ são introduzidas no arquivo. De seguida, as soluções dominadas da população E são eliminadas. Ou seja, o algoritmo limita-se simplesmente a preservar as elites. O algoritmo usa também estas soluções, juntamente com as soluções da população corrente, para participarem nas operações genéticas, com vista a influenciar a direcção da população no sentido das melhores regiões do espaço de pesquisa. Para evitar que o arquivo fique E fique superlotado, a população é limitada a $arq_{\dim}$ elementos e são guardadas apenas as soluções mais dispersas usando o método de agrupamento (*clustering*), descrito no paragrafo seguinte. Para calcular a aptidão, o algoritmo inicialmente atribui aos elementos da população externa E a aptidão S_i. Este valor é proporcional ao número de membros existentes na população corrente, $P(T)$, que a solução externa i domina, ou seja $S_i = \frac{n_i}{pop_{\dim}+1}$. Para os elementos da população corrente, a aptidão toma em consideração o número de elementos do arquivo E_T que dominam fracamente a solução j: $F_j = 1 + \sum_{i \in E_T \wedge i \preccurlyeq j} S_i$. Consequentemente, quanto menor é o valor de aptidão melhor é a solução. Este método garante que as soluções do arquivo obtenham valores de aptidão mais pequenos do que as soluções da população. Assim, os membros da população do AG dominados por soluções do arquivo apresentam valores de aptidão superiores. Com estes valores de aptidão é aplicada uma selecção por torneio binário à população $(E_T \cup P_T)$ para escolher as soluções com os menores valores de aptidão. O peso computacional do algoritmo é da ordem de $O((pop_{\dim})^3)$.

O algoritmo de agrupamento reduz a população E_T com arq'_{dim} elementos para arq_{dim}. Inicialmente, cada solução da população E_T é considerada residir num grupo *cluster* separado. Assim, existem arq'_{dim} grupos. Inicialmente, é calculada a distância entre todos os pares. Em geral, a distância é determinada usando a equação:

$$d_{1,2} = \frac{1}{\#c_1 \#c_2} \sum_{i \in c_1, j \in c_2} d(i,j) \tag{3.10}$$

A distância $d(i,j)$ pode ser calculada no espaço das variáveis de decisão ou no espaço dos objectivos, sendo $\#c_i$ o número de soluções do grupo i. De seguida, os dois grupos mais perto um do outro são fundidos (na figura 3.5 encontra-se ilustrada a fusão dos dois grupos representados a tracejado que originam o grupo 3). O processo repete-se até restarem arq_{dim} grupos. Finalmente, dentro de cada grupo, a solução com a distância média menor em relação às restantes soluções dentro do próprio grupo é eleita e as restantes soluções são descartadas. Como o método não tem parâmetros é bastante atractivo de usar.

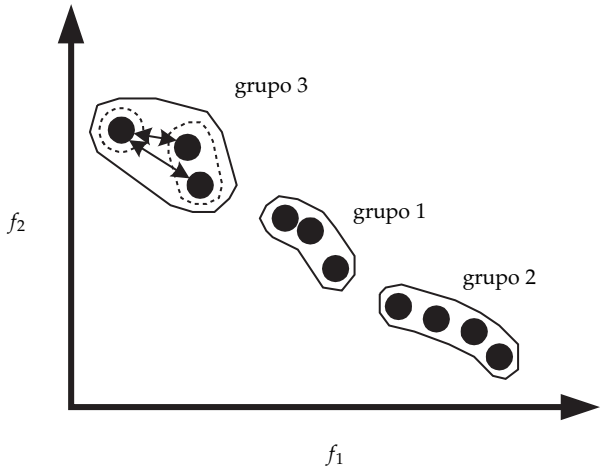


Figura 3.5. Método de agrupamento usado no algoritmo SPEA

Knowles e Corne [59] desenvolveram um AEMO baseado na estratégia evolutiva (*Pareto-Archived Evolution Strategy* – PAES) usando a estratégia $(1+1)$. Depois de uma solução ser escolhida aleatoriamente, sofre uma mutação de acordo com uma

função de distribuição normal com média zero. A melhor das duas soluções é transportada para a geração seguinte. A característica principal do algoritmo PAES incide no modo como o vencedor é escolhido. Na geração T, em adição ao progenitor p_T e ao descendente c_T, o algoritmo guarda num arquivo as melhores soluções encontradas até ao momento. Quando as soluções c_T e p_T são comparadas resulta um dos dois cenários possíveis:

1. Se apenas uma das solução é não-dominada então esta é escolhida como vencedora.

2. Se ambas as soluções forem não-dominadas então o descendente é comparado com o arquivo corrente, da seguinte forma:

   - O descendente é dominado por uma solução do arquivo. Então não é incluído no arquivo e o progenitor é mutado novamente.
   - O descendente domina algumas soluções do arquivo. Essas soluções são eliminadas e o descendente é arquivado. O descendente torna-se o progenitor da geração seguinte.
   - O descendente não é dominado pelo arquivo nem as soluções do arquivo dominam o descendente. O descendente é adicionado ao arquivo se couber. Para determinar o progenitor da geração seguinte (entre o progenitor actual ou o seu descendente) é determinada a densidade das soluções vizinhas e é escolhida a que tiver menor densidade. Outro processo alternativo consiste em dividir o espaço em quadrados, de lado d. O quadrado que contiver menos soluções determina o progenitor da geração seguinte. Para inserir uma solução remove-se a solução que pertencer ao hipercubo mais povoado (caso o descendente não pertencer a este). O parâmetro d é bastante importante no algoritmo pois controla directamente o comprimento do hipercubo.

Deb *et al.* [60] desenvolveram o algoritmo C-NSGA-II (*Clustered NSGA-II*), que é

semelhante ao algoritmo NSGA-II, diferindo apenas no mecanismo que promove a diversidade. O mecanismo usado é o método de agrupamento, adoptado no algoritmo SPEA, em substituição do método pombalino. Assim, as soluções da última frente a considerar (*i.e.* as soluções que vão preencher por completo a população seguinte) são determinadas pelo algoritmo de agrupamento.

Laummanns *et al.* [48] propõem uma estratégia de arquivo e de selecção que garante a convergência em direcção da frente óptima de Pareto, com a capacidade de obter uma representação significativa de toda a frente não-dominada. O algoritmo mantém um arquivo onde guarda soluções óptimas de Pareto$-\epsilon$, com dimensão finita, actualizado de acordo com o conceito de dominância ϵ. O parâmetro ϵ é escolhido tendo em conta a resolução do arquivo que se pretenda obter. Os autores apenas focam a actualização do arquivo considerando uma nova solução em cada geração.

Deb *et al.* [60] desenvolveram um algoritmo baseado no conceito de dominância-ϵ (ϵ-MOEA: *ϵ-Multi-Objective Evolutionary Algorithm*). O algoritmo divide o espaço de pesquisa numa grelha (em células ou hipercubos) e mantém a diversidade assegurando que em cada célula é apenas incluída uma solução. O algoritmo é constituído por duas populações co-evolutivas. A população do AG, $P(T)$, e um arquivo, $E(T)$. A população $P(T)$ é inicializada normalmente, sendo posteriormente inicializado o arquivo com as soluções não-dominadas-ϵ de $P(T)$. Em cada geração, é seleccionada um elemento de cada população para procriar. Para esse efeito, a solução de $P(T)$ é seleccionada da seguinte forma:

- Escolhem-se aleatoriamente dois elementos da população $P(T)$;
- Se um deles é dominate em relação ao outro então é escolhido como solução progenitora, p, que representará $P(T)$,
- Caso contrário é seleccionado aleatoriamente um dos dois elementos.

Para seleccionar a solução 'e' que representará $E(T)$ podem ser seguidas várias estratégias. A mais simples consiste em seleccionar aleatoriamente a solução 'e'. No acasalamento entre as soluções 'e' e 'p' são criadas λ (normalmente $\lambda = 1$) descendentes. Posteriormente estas soluções são comparadas com o arquivo e a população $P(T)$ para uma possível admissão.

Para comparar os descendentes com os elementos do arquivo de acordo com a dominância-ϵ recorre-se a um vector de identificação. Assim, cada elemento do arquivo e descendente é afectado com um vector de identificação $B = (b_1, b_2, \ldots, b_{n_{\text{obj}}})^{\text{T}}$ onde b_j é dado pela equação (3.11) onde $f_j^{\min}$ é o valor mais baixo do objectivo j e ϵ_j é o valor máximo para o qual o agente de decisão não vê diferença significativa no objectivo j ($\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_{n_{\text{obj}}})^{\text{T}}$). Desta forma, o vector de identificação divide o espaço em hipercubos, cada um com lado ϵ_j no objectivo j (ver figura 3.6).

$$b_j = \begin{cases} \frac{\lfloor (f_j - f_j^{\min}) \rfloor}{\epsilon_j}, & f_j \text{ é uma função para minimizar} \\ \frac{\lceil (f_j - f_j^{\min}) \rceil}{\epsilon_j}, & f_j \text{ é uma função para maximizar} \end{cases} \tag{3.11}$$

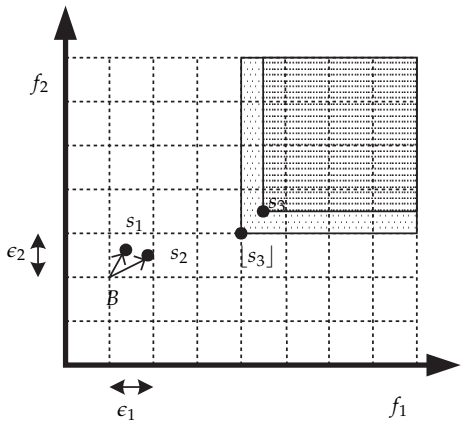


Figura 3.6. Conceito de dominância$-\epsilon$ (considerando as funções f_1 e f_2 a minimizar)

Com os vectores B calculados é adoptado o seguinte procedimento para determinar se um descendente é aceite no arquivo:

- Se o vector de identificação de um elemento do arquivo dominar o vector descendente (*i.e.*, o vector descendente é dominado-ϵ) então este não é aceite no arquivo.
- Se o vector de identificação B do descendente dominar um elemento do arquivo, então o descendente substitui esse elemento.
- Se nenhuma das situações se verificar, então o descendente é não-dominado-ϵ juntamente com as soluções do arquivo. Neste caso podem ocorrer duas situações:
  1. Se o descendente partilhar o mesmo vector B com um membro do arquivo (*i.e.* pertencem ao mesmo hipercubo) então são comparados da forma tradicional. O descendente substitui a solução do arquivo se dominar a solução do arquivo ou se forem ambas não-dominadas e o descendente se encontrar mais próximo do vector B (ver figura 3.6, soluções s_1 e s_2).
  2. No caso de um descendente não partilhar o mesmo vector B que as soluções do arquivo então é guardado.

Desta forma é salvaguardado que apenas existe num hipercubo uma solução com um vector distinto, garantindo assim que as soluções do arquivo são distintas e a dimensão final do arquivo é limitada.

Em relação à solução descendente substituir um elemento da população $P(T)$ pode seguir-se várias estratégias:

- Se a solução domina algum elemento de $P(T)$, então aquela substitui aleatoriamente um desses elementos;
- Se alguma solução da população domina o descendente, então este não é aceite;
- Se as duas condições anteriores não se verificarem então o descendente substitui aleatoriamente um elemento da população.

3.5 Índices de desempenho

Desde o aparecimento do primeiro AEMO têm sido desenvolvidos vários métodos e variações de AEMO [47]. De modo a estudar esses AEs, tornou-se necessário criar mecanismos capazes de medir o desempenho dos métodos. Como a frente óptima de Pareto pode ser convexa, côncava, descontinua ou contínua em intervalos, surgiram vários mecanismos diferentes com o fim de medir a diversidade num conjunto finito de soluções e a sua distância à frente óptima de Pareto. Além do mais, um bom resultado obtido por um mecanismo não significa necessariamente uma boa distribuição das soluções ou uma boa proximidade à frente óptima de Pareto. Assim, este conjunto de mecanismos permite um estudo alternativo e complementar do estudo da características das soluções. Este tipo de mecanismos são denominados por índices de desempenho. Assim, nesta secção são descritos alguns índices usados para medir e comparar a uniformidade da distribuição das soluções, a extensão da frente não-dominada e a convergência dos diversos AEs.

3.5.1 Índices de distribuição

Este tipo de índices têm como objectivo avaliar a uniformidade da distribuição das soluções não-dominadas encontradas pelos algoritmos.

Índices baseados na distância euclidiana

Schott [61] propôs um índice baseado na distância mínima, *spacing index*, (SP). Este índice é calculado através do desvio padrão das distâncias mínimas entre uma solução não-dominada e as restantes, utilizando as expressões (3.12), nas quais S representa o conjunto das soluções não-dominadas. Contudo, o método pode usar a mesma distância mais do que uma vez e, consequentemente, pode não utilizar qualquer distância entre dois grupos de soluções.

$$SP(S) = \sqrt{\frac{1}{\#S - 1} \sum_{i=1}^{\#S} (d_i - \overline{d})^2} \tag{3.12a}$$

$$d_i = \min_{s_k \in S \wedge s_k \neq s_i} \sum_{m=1}^{n_{\text{obj}}} |f_m(s_i) - f_m(s_k)| \tag{3.12b}$$

Outro método alternativo proposto por Deb em [62] consiste no índice de distribuição baseado na distância da frente (Δ'). O índice fundamenta-se na distância, d_i, que representa a média entre duas soluções consecutivas (3.13). O método pode apenas ser utilizado em problemas bi-objectivo.

$$\Delta'(S) = \sum_{i=1}^{\#S-1} \frac{|d_i - \overline{d}|}{\#S - 1} \tag{3.13}$$

Índices baseados em nichos

Os índices desta categoria são baseados no conceito de nicho. O número de soluções dentro de cada nicho é calculado e usado para determinar o índice de desempenho.

Zitler *et al.* [63] propuseram um índice que considera tanto a distribuição como o número de soluções não-dominadas no espaço dos objectivos (3.14), onde ρ é a distância radial do nicho e $\#S$ é o número de soluções do conjunto S. Se se considerar o desempenho no espaço dos parâmetros o índice é denominado por $M_2^*(S)$.

$$M_2^*(S) = \frac{1}{\#S - 1} \sum_{s_1 \in S} \#\{s_2 \in S | \parallel s_1 - s_2 \parallel > \rho\} \tag{3.14}$$

Tan [64] sugeriu um índice, *uniform distribution* – UD, para medir a distribuição das soluções não-dominadas (3.15), onde $nc(s_i)$ representa o contador do nicho da solução i em S (3.16) e σ é a distância radial do nicho.

$$UD(s) = \frac{1}{1 + D_{nc}} \tag{3.15a}$$

$$D_{nc} = \sqrt{\frac{\sum_{i=1}^{\#S} \left(nc(s_i) - \frac{1}{\#S} \sum_{i=1}^{S} nc(s_i)\#S \right)^2}{\#S - 1}} \tag{3.15b}$$

$$nc(s_i) = \#\{s_j \in S :\| s_i - s_j \| < \sigma\} - 1 \tag{3.16}$$

Tanto o índice M_2^* como o UD podem fornecer um bom resultado apesar das soluções não apresentarem uma boa distribuição [65].

3.5.2 Índices de desempenho de extensão da frente

Este tipo de índices medem a extensão da frente constituída pelas soluções não-dominadas.

Zitler *et al.* em [63] propuseram o índice de extensão máxima (M_3^*) que adopta a distância máxima entre as soluções extremas (3.17). Para o espaço dos parâmetros usam (M_3).

$$M_3^*(S) = \sqrt{\sum_{i=1}^{n_{\text{obj}}} \max\{\| s_i - s_j \| : s_i, s_j \in S\}} \tag{3.17}$$

Wu e Azarm [66] desenvolveram o índice (OS) para calcular a extensão da frente não-dominada (3.18), onde $OS_k(S, P_G, P_B)$ representa a extensão do objectivo k. P_G e P_B representam os hipervolumes.

$$OS(S, P_G, P_B) = \prod_{i=1}^{n_{\text{obj}}} OS_k(S, P_G, P_B) \tag{3.18a}$$

$$OS_k(S, P_G, P_B) = \frac{|\max_{s\in S} f_k(s) - \min_{s\in S} f_k(s)|}{|f_k(P_B) - f_k(P_G)|} \tag{3.18b}$$

3.5.3 Índices híbridos

Para além dos índices apresentados nas secções 3.5.1 e 3.5.2 existem outros tipos de índices [66] que utilizam informação relativa à extensão da frente e da distribuição das soluções ao longo desta.

3.5.4 Índices de convergência

Este tipo de índices medem a distância das soluções não-dominadas, S, encontradas pelo algoritmo e a frente de Pareto ou um conjunto de soluções não-dominadas que servem de referência, P.

Veldhuizen [67] usa a distância média do conjunto S ao P (3.19). Zitzler *et al.* [63] substituem d_i pela distância mínima à frente de Pareto (3.20).

$$GD(S, P) = \frac{1}{\#S}\left(\sum_{i=1}^{\#S} d_i^q\right)^{\frac{1}{q}} \tag{3.19a}$$

$$d_i = \min_{p\in P}\left\{\sqrt{\sum_{k=1}^{n_{\text{obj}}}(f_k(s_i) - f_k(p))^2}\right\} \tag{3.19b}$$

$$M_1^* = \frac{1}{\#S}\sum_{i=1}^{\#S}\min\{\parallel s_i - p_j \parallel; p_j \in P\} \tag{3.20}$$

Uma alternativa [65] aos índices referidos, consiste em calcular a distância entre a frente de Pareto, ou de referência, ao conjunto se soluções não-dominadas encontrada pelo algoritmo .

Existem outros índices baseados no volume, *i.e.* a área que é dominada pelo conjunto de soluções S. Ou seja, quanto maior for a área que as soluções dominam tanto melhor. Zitzler *et al.* [63] propuseram o índice (H^2) indicado na equação (3.21).

$$H^2 = \prod_{i=1}^{n_{\text{obj}}} (f_i^{\max} - f_i^{\min}) \tag{3.21}$$

Outro índice de desempenho é o hipervolume também conhecido por métrica-S ou medida de Lebesgue. De seguida descreve-se este índice de desempenho. O hipervolume de um conjunto de soluções fornece uma grandeza escalar de acordo com o espaço objectivo dominado por essas soluções. Com apenas um valor este índice fornece uma indicação da proximidade das soluções encontradas com a frente óptima de Pareto e, até certo ponto, dá uma indicação da distribuição das soluções no espaço dos objectivos. Apesar de Fleischer [68] reclamar que o seu algoritmo tem uma complexidade computacional de $O((pop_{\text{dim}})^3 \left(n_{\text{obj}}\right)^2)$, While [69] provou que o algoritmo é polinomial em função do número de soluções e é exponencial em função do número de objectivos.

Dado um conjunto de soluções S, o hipervolume é calculado relativamente a um ponto (solução) de referência que corresponde ao ponto com o pior valor objectivo encontrado em todas as soluções. Assim, quando se comparam dois conjuntos S e S', o conjunto que ocupa um hipervolume maior é o que apresenta o melhor conjunto de soluções para o problema. Por outra palavras, o algoritmo que obteve este conjunto de soluções é aquele que apresenta melhor desempenho.

O algoritmo para obter o hipervolume é descrito em seguida. Inicialmente as soluções não-dominadas são colocadas numa lista. De seguida é processada uma solução de cada vez, *i.e.*, retira-se uma solução da lista e calcula-se o hipervolume relativo que esta domina. Cada vez que uma solução é processada, são geradas n_{obj} soluções que dominam o hipervolume restante não considerado pela contribuição da solução geradora. Os pontos gerados não-dominados são inseridos na lista.

Por exemplo considere-se a lista de soluções não-dominadas: $A = \{6, 9, 4\}$, $B = \{9, 7, 5\}$, $C = \{1, 12, 3\}$ e $D = \{4, 2, 9\}$ representada na figura 3.7. Quando se retira a solução A são geradas as seguintes soluções: $A^1 = \{4, 9, 4\}$, $A^2 = \{4, 7, 4\}$ e $A^3 = \{4, 9, 3\}$ ($A^i = \{a_{i-1}, k, a_{i+1}$, com $k = \max(j_i)$, $k < a_i$, $j_i = a_i..d_i$, $i = 1..3$). Como a solução A^2 é dominada pela solução B não é inserida na lista. O hipervolume relativo resultante da remoção da solução A é $(6-4)(9-7)(4-3) = 6$.

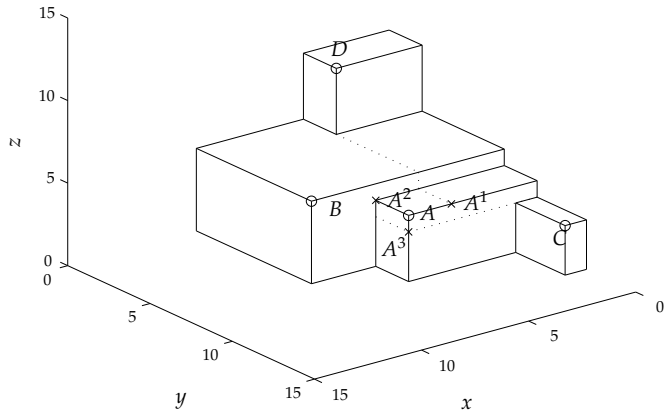


Figura 3.7. Hipervolume das soluções $A - D$

O hipervolume é também utilizado como critério de selecção nos AEMO, onde o valor usado é o hipervolume relativo que cada solução que domina [70].

3.6 Resumo

Neste capítulo apresentaram-se os conceitos básicos dos PMO. De seguida foram apresentados alguns AEMO bastante conhecidos, alguns por razões históricas, outros pela sua popularidade. Foram também descritos AEMO que preservam a elite. Por último, foram referidos alguns índices de desempenho que permitem analisar os algoritmos multi-objectivo.

4 Planeamento de trajectórias e aplicações robóticas usando AGs

4.1 Introdução

Este capítulo encontra-se organizado em quatro secções principais. Assim, nas secções 4.2 e 4.3 é descrito, respectivamente o planeamento de trajectórias para manipuladores robóticos e o tipo de problemas existentes no planeamento de trajectórias. De seguida, na secção 4.4 são analisadas as principais representações dos obstáculos. Por último, são apresentadas algumas aplicações robóticas usando AGs.

4.2 Planeamento de trajectórias

O planeamento de trajectórias para manipuladores robóticos consiste no processo de criar trajectórias livres de colisões, para os manipuladores, que lhe permitam desempenhar as tarefas pretendidas. Os planeadores de trajectórias têm como função substituir o operador de especificar explicitamente as trajectórias. Assim, o operador fica livre para se concentrar na tarefa em vez de se preocupar com o movimento do robô. O operador necessita apenas de especificar os pontos inicial e final da trajectória, ficando o planeador encarregue de gerar uma trajectória adequada para o

manipulador percorrer.

As trajectórias são compostas por sucessivos deslocamentos do braço robótico. Assim, uma trajectória pode ser vista como uma sequência de pontos nos quais o órgão terminal deve passar. Como resultado da movimentação do braço, ao longo dos pontos discretos, obtém-se uma trajectória continua. Consequentemente, a optimização da trajectória de um robô implica a identificação da combinação óptima de pontos e do número de posições intermédias. A optimização de um manipulador robótico é bastante difícil devido às não-linearidades correlacionadas da dinâmica e à dimensão do espaço de pesquisa das trajectórias.

O planeamento de trajectórias pode ser implementado usando a cinemática directa ou a inversa. Quando é usada a cinemática directa, o problema é resolvido directamente no espaço das juntas. Quando o problema é resolvido através da cinemática inversa, é inicialmente determinada, no espaço operacional, a trajectória do órgão terminal. De seguida, usando a cinemática inversa, são calculados os valores das juntas. Os valores das juntas obtidos sucessivamente são posteriormente usados pelo planeador para formar a trajectória. No entanto, quando a cinemática inversa é usada, devido à existência de pontos singulares, podem surgir alguns problemas [71], tais como:

- A mobilidade do manipulador é reduzida, *i.e.* não é possível impor certos movimentos ao órgão terminal;
- A eventual ocorrência de múltiplas soluções para o problema da cinemática inversa;
- Na vizinhança das singularidades, baixas velocidades no espaço operacional podem requerer altas velocidades no espaço das juntas.

De seguida é analisada a cinemática de um manipulador robótico. Assim, para o manipulador da figura 4.1 as coordenadas $\{x, y\}$ do órgão terminal são dadas pela

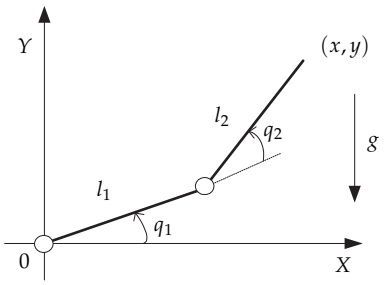


Figura 4.1. Manipulador planar com dois elos e duas juntas rotacionais ($n = 2$)

equação (4.1). A cinemática inversa para o robô de dois elos é dada pela equação (4.2).

$$x = \sum_{i=1}^{n} l_i \cos(\theta_i) \tag{4.1a}$$

$$y = \sum_{i=1}^{n} l_i \sin(\theta_i) \tag{4.1b}$$

$$\theta_k = \sum_{i=1}^{k} q_i \tag{4.1c}$$

$$q_2 = \pm \arccos \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{4.2a}$$

$$q_1 = \arctan \frac{y}{x} - \arctan \frac{l_2 \sin(q_2)}{l_1 + l_2 \cos(q_2)} \tag{4.2b}$$

No cálculo da trajectória o planeador deve ter em atenção o esforço a realizar pelos motores do manipulador durante a trajectória de modo a que esta não exceda o binário máximo que os motores possam suportar. Para esse fim, o modelo dinâmico tem um papel fundamental no movimento da trajectória. Este modelo pode ser determinado a partir do formalismo de Lagrange.

As equações da dinâmica para um manipulador robótico planar encontram-se na equação (4.3). Sendo $\boldsymbol{T}$ o vector $n \times 1$ relativos aos binários dos actuadores; $\boldsymbol{J}$ a matriz das inércias $n \times n$; $\boldsymbol{C}$ a matriz $n \times 1$ dos binários coriolis/centripetos; $\boldsymbol{G}$ o

vector $n \times 1$ dos binários gravitacionais; $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ e $\ddot{\boldsymbol{q}}$ são os vectores $n \times 1$ da posição, velocidade e aceleração das juntas, respectivamente.

$$\boldsymbol{T} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) \tag{4.3}$$

A partir da equação de Lagrange (4.4a) e das energias cinética (4.4b) e potencial (4.4c) pode determinar-se a equação do binário τ_j relativo a junta $j = \{1, \ldots, n\}$ (4.4d) obtendo-se a expressão geral (4.5) para o binário do motor j para um manipulador com n juntas [71]. Onde m_i e r_i são, respectivamente, a massa e o centro de massa do elo i, g é a aceleração da gravidade, γ_i são as variáveis lógicas dadas pela equação (4.6) e os valores de θ_p são fornecidos pela equação (4.1c).

$$L = E_c - E_p \tag{4.4a}$$

$$E_c = \frac{1}{2}mv^2 \tag{4.4b}$$

$$E_p = mgy \tag{4.4c}$$

$$\tau_j = \frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} \tag{4.4d}$$

$$\begin{aligned}
\tau_j = \sum_{i=j}^{n} m_i \Bigg\langle & \sum_{p=1}^{i-1} \left(l_p^2 \sum_{u=1}^{p} \ddot{q}_u \gamma_1 \right) + r_i^2 \sum_{u=1}^{i} \ddot{q}_u \\
& + \sum_{p=2}^{i} \Bigg\{ \sum_{k=1}^{p-1} \Bigg[l_k (l_p \gamma_2 + r_p \gamma_3) \Bigg(\Bigg(\cos(q_k + \theta_p) \left(\sum_{u=1}^{p} \ddot{q}_u + \sum_{u=1}^{k} \ddot{q}_u + \right) \right. \\
& \left. - \sin(q_k + \theta_p) \left(\left(\sum_{u=k+1}^{p} \dot{q}_u \right)^2 + 2 \sum_{u=1}^{k} \dot{q}_u \sum_{u=k+1}^{p} \dot{q}_u \right) \right) \gamma_4 \\
& + \left(\sin(k + \theta_p) \left(\sum_{u=1}^{k} \dot{q}_u \right)^2 + \cos(k + \theta_p) \sum_{u=1}^{k} \ddot{q}_u \right) \gamma_5 \Bigg) \Bigg] \Bigg\} \\
& + g \sum_{p=1}^{i-1} l_p \cos(\theta_p) \gamma_1 + r_i \cos(\theta_i) \Bigg\rangle
\end{aligned} \tag{4.5}$$

$$\gamma_1 = \begin{cases} 0, & \text{se } j > p; \\ 1, & \text{se } j \leq p. \end{cases} \tag{4.6a}$$

$$\gamma_2 = \begin{cases} 0, & \text{se } p > i-1; \\ 1, & \text{se } p \leq i-1. \end{cases} \tag{4.6b}$$

$$\gamma_3 = \begin{cases} 0, & \text{se } p \neq i; \\ 1, & \text{se } p = i. \end{cases} \tag{4.6c}$$

$$\gamma_4 = \begin{cases} 0, & \text{se } j > k; \\ 1, & \text{se } j \leq k. \end{cases} \tag{4.6d}$$

$$\gamma_5 = \begin{cases} 0, & \text{se } (j < k+1) \vee (j > p); \\ 1, & \text{se } k+1 \leq j \leq p. \end{cases} \tag{4.6e}$$

4.3 Classificação dos problemas no planeamento de trajectórias

Existem várias categorias [72] nas quais os problemas de planeamento de movimento podem ser classificados. Estas categorias são identificadas de acordo com a construção física do robô, do ambiente e da informação que o robô dispõe do espaço de trabalho:

Tipo de ambiente
- Estacionário;
- Variante no tempo;
- Mutante (*Comformable*);
- Objectos móveis.

Num ambiente estacionário todos os objectos estão imóveis. Contudo, quando os objectos se podem deslocar o problema torna-se variante no tempo. Se os objectos podem modificar a sua morfologia o ambiente é mutante. O ambiente é dito de objectos móveis quando o robô os pode deslocar. Se o planeador conhece *a priori* a informação relativa ao ambiente o problema designa-se por estático. Caso contrário é designado por dinâmico. Neste caso, a informação relativa ao ambiente é

guardada e deve ser actualizada sempre que se detecta qualquer alteração.

O robô pode operar sozinho, possuir algumas restrições ou pode ter necessidade de cooperar com outros manipuladores existentes no ambiente de trabalho:

Tipo de robô $\begin{cases} \text{Manipulador único;} \\ \text{Com restrições;} \\ \text{Cooperante.} \end{cases}$

4.4 Representação dos obstáculos

Cada objecto no ambiente de trabalho deve ser modelado e guardado numa estrutura de dados adequada. O conhecimento dos obstáculos deve ser determinado antes da trajectória ser calculada. Nos planeadores interactivos, os obstáculos são detectados pelos sensores à medida que o manipulador se desloca.

De seguida são descritos os principais métodos utilizados na representação de obstáculos:

- No método da matriz de células o ambiente de trabalho é dividido em células, todas com idêntica dimensão. Se a célula for ocupada por uma fracção do objecto é-lhe atribuído o valor '1' (o manipulador não pode passar por esta célula), caso contrário é colocada a '0'. A precisão do método depende da resolução da matriz e requer uma quantidade de memória considerável. Contudo é adequado para objectos com morfologias irregulares.

- No método de árvore de células o ambiente é, inicialmente, dividido em quartis. Se alguma das quatro células contém partes de um objecto, esta é novamente dividida. Esta divisão termina quando for alcançada a resolução pretendida.

- No método poligonal os objectos são aproximados por polígonos ou pela união

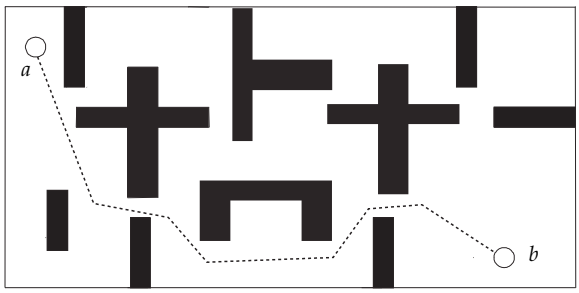


Figura 4.2. Trajectória de um robô móvel ($n = 2$)

de vários polígonos. Na representação CSG (*Constructive solid geometry*), os objectos são representados pela união, intersecção de conjunto de figuras básicas (*e.g.*, quadriláteros, círculos).

4.5 Aplicação de algoritmos genéticos na robótica

Nos últimos trinta anos os AGs têm sido aplicados em diversos campos da robótica, nomeadamente: na geração de trajectórias para robôs móveis, na geração de trajectórias para manipuladores robóticos, na selecção ou síntese de manipuladores, na locomoção de robôs, no controlo de garras mecânicas, na calibração de manipuladores, em sistemas que requerem aprendizagem entre outros. Nesta secção é apresentado uma revisão de algumas aplicações robóticas referidas usando AGs.

4.5.1 Geração de trajectórias para robôs móveis

O planeamento de trajectórias para robôs móveis consiste em determinar uma trajectória, livre de colisões, entre os dois locais específicos (figura 4.2). Neste tipo de problema é desejável que o robô móvel encontre o ponto de destino sem colidir com os obstáculos e garantido uma margem de segurança com os mesmos.

Um navegador/planeador evolutivo flexível e adaptativo foi desenvolvido por Xiao

et al. [73]. O método realiza o planeamento em tempo real e/ou recorre a trajectórias previamente calculadas. O algoritmo permite detectar alterações do ambiente de trabalho e ajusta as probabilidades do AG enquanto o robô se desloca. O objectivo consiste em percorrer o caminho mais curto garantindo sempre as margens de segurança com os obstáculos.

Han *et al.* [74] propõem outro algoritmo semelhante mas que incorpora um sistema de visão no sistema (que tanto pode estar incluído no robô como em qualquer parte do ambiente) e de um algoritmo que planeia o percurso. No sistema dinâmico o algoritmo deve identificar em tempo real o movimento dos obstáculos e ajustar a trajectória em tempo real.

Ramírez *et al.* [75] resolveram o problema através de uma implementação baseada num controlador preditivo. O controlador deve evitar os obstáculos do ambiente. Na resolução do problema são consideradas restrições ao nível da velocidade máxima que os robôs podem atingir. O AG é usado na estratégia de controlo preditivo, na navegação de um robô móvel num ambiente parcialmente estruturado.

Geimeinder e Gerke [76] apresentam um algoritmo para um ambiente, dividido em quadrados, com pavimento desigual tendo como objectivo minimizar o consumo de energia durante o movimento.

Existem também aplicações que disponibilizam mais que uma solução. Nomeadamente, Dozier *et al.* [77] usam um algoritmo de selecção para um sistema evolutivo com o fim de determinar caminhos admissíveis. Este algoritmo combina regras difusas com o mecanismo de selecção numa perspectiva com vários objectivos. O algoritmo disponibiliza ao operador caminhos alternativos de acordo com os objectivos especificados. Outra aplicação usando regras difusas para obter um bom comportamento no deslocamento foi desenvolvida por Gacôgne [78]. As regras permitem ao robô determinar a sua direcção e velocidade sem seguir restrições precisas. Para responder a um conjunto de soluções que se adeqúem ao problema em diversas situações usam um algoritmo genético multi-objectivo. Por outro lado, Hocaoğlu e

Sanderson [79–81] propõem um algoritmo para determinar o caminho de aplicações robóticas, tais como: robôs móveis, robôs articulados e para objectos de diferentes geometrias. O algoritmo também pode fornecer caminhos alternativos para essas aplicações. Para este efeito usam técnicas de nicho para gerar e preservar trajectórias diferentes na população ao longo da evolução.

Os AGs são também usados em sistemas cooperantes. De facto, Cai e Peng [82] propõem um AG cooperativo, co-evolutivo adaptativo para sistemas multi-robóticos. O sistema é constituído por dois robôs móveis e tem como objectivo planear os caminhos dos robôs de modo a efectuar uma tarefa. Existem uma população para cada robô, que evolui internamente de forma independente; contudo, a coordenação e a cooperação entre as populações é reflectida na função de aptidão de cada população.

Existem também trabalhos de manipuladores móveis, *i.e.*, manipuladores robóticos montados em cima de veículos móveis. Chen e Zalzala [83] apresentam um algoritmo que tem como finalidade optimizar o percurso da base móvel, a posição e a configuração do manipulador, sem ocorrer qualquer colisão. O cálculo do caminho é baseado no campo potencial discreto e tomando em consideração: a distância aos obstáculos da base, o binário aplicado, a capacidade de manipulação e a distribuição óptima do binário em relação ao manipulador. Outra aplicação com um robô idêntico é apresentada por Chen e Zalzala [84] de modo a que o manipulador siga um conjunto de pontos predefinidos. O problema é resolvido usando a cinemática inversa e um método polinomial. O algoritmo procura minimizar o binário necessário e maximizar a manipulabilidade do manipulador, sem que a base colida com os obstáculos. Wu *et al.* [85] resolvem o problema usando um controlador genético difuso onde o robô conhece parcialmente o ambiente.

Na tabela 4.1 encontram-se as principais características dos algoritmos referidos. Onde a segunda coluna com o título "tempo real" indica que o algoritmo permite efectuar ou recalcular a trajectória em tempo real. A terceira coluna menciona o

objectivo principal da função objectivo. A coluna "Multi-objectivo" assinala os algoritmos que fornecem caminhos alternativos. A quinta e sexta coluna informam se o ambiente inclui obstáculos e o tipo de ambiente. A coluna seguinte assinala se o algoritmo usa regras difusas. Por fim é indicado se o robô tem incorporado um manipulador e se existe mais que um robô no ambiente de trabalho.

Autores	Tempo real	Objectivo	Multi-Objectivo	Obstáculos	Ambiente	Regras difusas	Manipulador	Robôs cooperantes
Xiao *et al.*	x	Distância	-	x	Dinâmico	-	-	-
Han *et al.*	x	Distância	-	x	Dinâmico	-	-	-
Ramírez *et al.*	x	Controlo de velocidade	-	x	Dinâmico	-	-	-
Geimeinder e Gerke	-	Consumo de Energia	-	x	Discreto / Irregular	-	-	-
Dozier *et al.*	-	-	x	x	-	x	-	-
Gacôgne	-	-	x	x	-	x	-	-
Hocaoğlu e Sanderson	-	-	x	x	-	-	-	-
Cai e Peng	-	Distância	-	x	-	-	-	x
Chen e Zalzala [83]	-	Binário, manipulablidade	-	x	Discreto	-	x	-
Chen e Zalzala [84]					-			
Wu *et al.*	x	Distância	-	x	Dinâmico	x	x	-

Tabela 4.1. Aplicações de AGs na robótica móvel

4.5.2 Geração de trajectórias para manipuladores robóticos

Como foi referido anteriormente as trajectórias são compostas por uma sequência de deslocamentos do braço robótico. Uma trajectória pode ser vista como uma sequência de pontos nos quais o ponto terminal deve passar. Como resultado da movimentação do braço, ao longo dos pontos discretos obtém-se uma trajectória continua. Assim, a optimização da trajectória de um manipulador implica determinar a combinação óptima dos pontos e das configurações intermédias (figura 4.3).

Nesta secção são apresentadas algumas aplicações para gerar trajectórias de manipuladores robóticos.

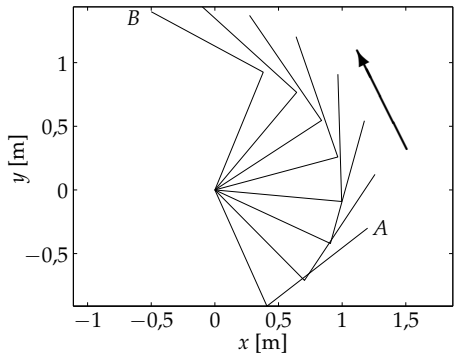


Figura 4.3. Trajectórias de um manipulador

Davidor [86] foi o primeiro a aplicar os AGs ao planeamento de trajectórias. Para esse fim, usa um vector binário com comprimento variável para representar a trajectória $3R$[1] e mede o seu desempenho através do erro entre a trajectória pretendida e a realizada. Para determinar as configurações do manipulador usa a cinemática inversa. Outra aplicação semelhante para um robô $7R$ é proposta por Nearchour e Aspragathos [87]. Neste problema é considerado o movimento ponto-a-ponto do manipulador robótico num ambiente com obstáculos. O problema consiste em ir determinando a configuração que o manipulador deve adoptar para posicionar e orientar o órgão terminal. Também Lavoie e Boudreau [88] resolvem um problema semelhante para manipuladores com 7 ou 10 gdl. Na resolução do problema é usada a cinemática inversa e o manipulador deve passar pelos pontos intermédios pré-definidos. O algoritmo tem como objectivo minimizar o deslocamento angular do manipulador sem colidir com os obstáculos do ambiente de trabalho. Por outro lado, Doyle e Jones [89] propõem um AG para determinar as configurações sucessivas de um manipulador. O algoritmo pesquisa o espaço das configurações de modo a encontrar o melhor caminho. Lee *et al.* [90] apresentam duas aplicações usando AEs, para um robô $2R$. Uma aplicação é baseada em AGs, usando o método das

[1]Manipulador robótico constituído por três juntas rotacionais

penalidades, e a outra adopta EEs incorporando técnicas heurísticas, usando escalonamento temporal para lidar com as restrições (relativas ao binário) nos actuadores das juntas. Estes autores fazem, também, uma comparação com a programação não-linear.

Para além dos AGs também são usados outros algoritmos inspirados na natureza. Kubota *et al.* [91, 92] usam um AE hierárquico para determinar a trajectória de um manipulador redundante livre de colisões. O algoritmo é baseado na co-evolução entre uma população de vírus e uma de hospedeiros e tem como objectivo minimizar a distância e a potência requerida para efectuar a trajectória. Por outro lado, Luo e Wei [93] resolvem o problema para um manipulador $3R$ baseando-se num sistema de imunidade biológico. O problema é reduzido a um objectivo (agregando os vários objectivos) e a um parâmetro (tornando os valores das juntas dependentes deste parâmetro).

Na geração de trajectórias o AG pode ser usado para determinar os pontos intermédios de polinómios temporais que as representam. Neste tipo de aplicações consideram-se nulas as velocidades e as acelerações nos extremos. Nesta perspectiva, Wei-Min e Yu-Geng [94] desenvolveram um método para manipuladores $2R$ e $3R$ incorporando restrições cinemáticas, dinâmicas e de controlo. O método usa polinómios de grau 4 e 5 e tem como objectivo determinar a trajectória que demora menos tempo. Por outro lado, Wang e Zalzala [95] desenvolveram um algoritmo para um robô industrial também com o objectivo de fornecer uma trajectória num intervalo de tempo reduzido. Para esse fim, dividem o espaço das juntas numa grelha, usam um polinómio de grau 6 e um AG binário com técnicas de pesquisa heurísticas na sua resolução. Outra aplicação foi proposta por Tian e Collins [96] que usam um AG binário e um polinómio cúbico na geração da trajectória de um manipulador $2R$ para um ambiente com obstáculos pontuais.

Os AGs também são aplicados a sistemas com manipuladores cooperantes, ou seja,

sistemas onde existe mais que um manipulador a operar. Rana e Zazala [97] propõem uma técnica evolutiva com dois manipuladores. Este algoritmo minimiza a duração das trajectórias de modo a evitar colisões entre os manipuladores. O planeamento é realizado no espaço das juntas do manipulador, sendo a trajectória representada por vectores que guardam os pontos intermédios de *splines* polinomiais cúbicas. Outro método é descrito por Ridao *et al.* [98] baseando-se num AE híbrido. O AG implementado tem como finalidade determinar a sequência de pontos de sincronização de modo a minimizar a duração da trajectória. Para este efeito o algoritmo usa caminhos dos manipuladores previamente determinados (por outros algoritmos como a pesquisa local). Ali *et al.* [99] propuseram um método para dois manipuladores $2R$ tridimensionais tendo como objectivo minimizar a distância percorrida sem colidir com os obstáculos. Uma outra aplicação, tendo como objectivo minimizar a energia requerida, foi proposta por Garg e Kumar usando, simultaneamente, AGs e SA [100, 101].

No planeamento de trajectórias podem também ser aplicados AGMO. Assim, Ortmann [102] apresenta um problema multi-critério para determinar a trajectória de um manipulador num ambiente com obstáculos. O algoritmo multi-objectivo é usado para identificar múltiplas posições das juntas para uma trajectória.

Na tabela 4.2 encontram-se resumidas as principais diferenças entre os algoritmos descritos. Assim, a segunda coluna "cinemática" indica como são utilizadas as equações da cinemática (D–directa, I–inversa ou não é referida). Na coluna seguinte é assinalado o tipo de manipulador utilizado e se o ambiente inclui obstáculos. A coluna "objectivo" indica a principal componente da função de aptidão. De seguida é referida a dimensão do espaço de trabalho e se o AG usa técnicas de co-evolução. Na oitava coluna "cooperantes" indica se existe mais que um manipulador no espaço de trabalho e o seu número. As colunas "dinâmica" e "interpolação polinomial" indicam se são adoptadas as equações da dinâmica e funções polinomiais ou *splines*. Por fim é dito se o algoritmos se baseiam em sistemas imunológicos e se usam AGs

híbridos.

Autores	Cinemática	Manipulador	Obstáculos	Objectivo	Ambiente	Co-Evolutivo	Cooperativos	Dinâmica	Interpolação polinomial	Sistemas imunológicos	AGs Híbridos
Davidor	I	3*R*	-	Distância cartesiana	2*D*	-	-	-	-	-	-
Nearchour e Aspragathos	I	7*R*	S	Distância cartesiana	2*D*	-	-	-	-	-	-
Lavoie e Boudreau	I	7;10*R*	S	Distância angular	2*D*	-	-	-	-	-	-
Doyle e Jones		3*R*	S	Distância angular	2*D*	-	-	-	-	-	-
Lee *et al.*		2*R*	-	Duração da trajectória	2*D*	-	-	S	-	-	-
Kubota *et al.*	D	7 gdl	S	Distância e Potência	3*D*	S	-	S	-	-	-
Luo e Wei	I	3*R*	-	Distância cartesiana	2*D*	-	-	-	-	S	-
Wei-Min e Yu-Geng	I	2;3*R*	-	Duração da trajectória	2*D*	-	-	S	S	-	-
Wang e Zalzala	D	6*R*	-	Duração da trajectória	3*D*	-	-	-	S	-	-
Tian e Collins	I	2*R*	S	Distância angular	2*D*	-	-	-	S	-	-
Rana e Zazala	I	2;3*R*	-	Duração da trajectória	2;3*D*	-	2	S	S	-	-
Ridao *et al.*		2;3*R*	S	Duração da trajectória	2;3*D*	-	2	-	-	-	S
Ali *et al.*	D	2*R*	S	Distância cartesiana	3*D*	S	2	S	S	-	S
Garg e Kumar	D	2*R*	-	Energia	2*D*	-	2	S	S	-	S
Ortmann	D	3*R*	-	Oscilação residual	3*D*	-	-	-	S	-	S

Tabela 4.2. Aplicações de AGs no planeamento de trajectórias para manipuladores robóticos

4.5.3 Selecção e síntese de manipuladores

A escolha de um manipulador robótico deve ter em atenção a tarefa a ser executada e os obstáculos que se encontram no ambiente de trabalho. Um exemplo deste tipo de aplicação é apresentado por Chedmail e Ramstein [103] que permite seleccionar, dentro de um conjunto predefinido de robôs, qual o que se adequa melhor a uma determinada tarefa.

Um outro problema consiste na síntese de robôs. Assim, trata-se de projectar um manipulador robótico que satisfaça determinadas restrições ou que permita executar diferentes tarefas com o melhor desempenho possível. Uma aplicação para um robô $3R$ foi apresentado por Kim e Khosla [104]. O projecto consiste em determinar a trajectória e o comprimento dos elos usando um AG. Outra aplicação foi desenvolvida por Gallant e Bourdeau [105] para um manipulador RPR paralelo com 3 gdl, tendo como objectivo obter um espaço de trabalho o mais parecido com o especificado, maximizar a sua destreza, e colocando os pontos singulares fora do ambiente de trabalho. Também, Sobh *et al.* [106] apresentam uma aplicação *web* que permite projectar um manipulador com 3 gdl tendo em atenção os parâmetros cinemáticos e dinâmicos. O manipulador óptimo é aquele que apresenta maior manipulablidade, menor erro de posição e de velocidade. Por outro lado, o trabalho de Kosinska *et al.* [107] tem como objectivo determinar os parâmetros que conduzam a um espaço de trabalho máximo ou com o melhor desempenho cinemático e dinâmico. O projecto procura determinar três parâmetros geométricos essenciais para um manipulador paralelo.

A síntese de um manipulador pode ser realizada a partir de um conjunto de módulos tais como elos e juntas pré-definidas. Neste tipo de aplicações encontra-se a desenvolvida por Chocron e Bidaud [108] que consiste em determinar o local da base e os módulos que constituirão o braço usando a cinemática e a geometria como medida de desempenho. Outra aplicação semelhante é a de Han *et al.* [109] que usa as equações da cinemática para determinar a configuração do robô e, numa segunda

fase, adopta um AG para obter a trajectória óptima. Por outro lado, Bi e Zhang [110] usam como índices de desempenho: a manipulabilidade, os erros e os binários máximos nos pontos de trabalho e a distância percorrida.

Na síntese e planeamento de robôs flexíveis Zhu *et al.* [111] propuseram um AG para determinar a estrutura e os parâmetros de controlo de um manipulador com 2 elos que melhor se adeque para uma determinada trajectória. Os índices de controlo usados são a velocidade, a precisão do órgão terminal, a deflexão máxima permitida, os binários máximos requeridos entre outros.

Na tabela 4.3 encontram-se realçadas as principais diferenças dos algoritmos descritos. A tabela indica o tipo de manipulador adoptado, o uso de módulos pré-definidos, o recurso às equações da dinâmica para a construção do manipulador, os principais objectivos do projecto e, por fim, assinala-se quando o manipulador tem uma estrutura flexível.

Autores	Manipulador	Módulos discretos	Dinâmica	Objectivo	Flexível
Kim e Khosla	3*R*	-	-	Tarefa	
Gallant e Bourdeau	*RPR*	-	-	Tarefa	
Sobh *et al.*		-	S	Espaço de trabalho	
Kosinska *et al.*	Paralelo	-	S	Espaço de trabalho	
Chocron e Bidaud		S	-	Geometria e cinemática	
Han *et al.*		S	-	Tarefa	
Bi e Zhang		S	S	Binário e manipulabilidade	
Zhu *et al.*	2*R*	-	S	Deflexão, binário e precissão	S

Tabela 4.3. Aplicações de AGs na síntese de manipuladores robóticos

4.5.4 Locomoção de robôs

Os AGs são bastante utilizados no estudo da locomoção através de robôs com pernas (figura 4.4) onde, normalmente são adoptados para determinar um ciclo da locomoção do robô. A trajectória de locomoção será depois composta pela parte inicial da locomoção, pela repetição do ciclo e por uma parte que diz respeito à finalização do movimento. Nesta secção são apresentadas algumas aplicações de AGs em sistemas robóticos de locomoção.

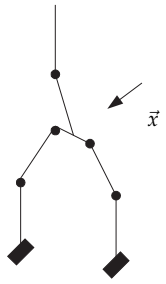


Figura 4.4. Locomoção de um robô de dois membros inferiores

Os AGs podem ser usados como ferramentas principais na geração de trajectórias como nos exemplos seguintes. Parker *et al.* [112] desenvolveram um AG para um robô de seis pernas. O algoritmo é responsável pelas três fases do deslocamento do robô tendo como objectivo determinar um movimento suave. Outro algoritmo para um robô de dois membros foi desenvolvido por Cabodevila e Abba [113] de modo a obter um movimento autónomo com o menor dispêndio de energia possível. Outra aplicação, recorrendo aos AGs e a PE, foi proposta por Arakawa e Fukuda [114]. O método hierárquico proposto é dividido em duas camadas: a camada PE, que gere a configuração interpolada da locomoção bípede do robô, e a camada dos AGs, que selecciona a configuração interpolada efectiva da locomoção bípede do robô, com o objectivo de minimização da energia total dos actuadores. Na aplicação de Luk *et al.* [115] é apresentado um algoritmo para controlar um robô, Robug IV, de 8 patas. O método baseia-se na simulação evolutiva com o fim de optimizar as margens de estabilidade na sequência de locomoção. O método tem como objectivo determinar padrões ideais de locomoção e melhorar os sistemas de controlo bem

como a estrutura do robô. Além disso, o algoritmo permite determinar a locomoção do robô mesmo tendo um membro inoperante.

Existem outras aplicações onde os AGs têm um papel secundário, como é o caso das aplicações de Lewis e Fagg [116] e de Fukuda *et al.* [117], onde os AGs são usados para optimizar as redes neuronais que, por sua vez, geram o movimento dos robôs. Por outro lado, Farritor e Dubowsky [118] propõem um AG para determinar o melhor plano de acção realizável que um robô pode executar com sucesso. O plano de acção é construído a partir de acções pré-definidas.

Na tabela 4.4 encontram-se realçadas algumas das características dos algoritmos referidos. Assim, a tabela inclui o número de patas e os graus de liberdade de cada robô. De seguida, na coluna "objectivo", são indicados as principais optimizações usadas nas funções de aptidão. Por fim indica-se se os algoritmos recorrem a redes neuronais e a *splines* no planeamento do movimento.

Autores	Membros inferiores	Graus de liberdade	Objectivo	Tolerante a falhas	Redes Neuronais	*splines*
Parker *et al.*	6	2	Locomoção suave	-	-	-
Cabodevila e Abba	2	5	Energia	-	-	-
Arakawa e Fukuda	2	13	Energia	-	-	S
Luk *et al.*	8	8	Margem de estabilidade, Estrutura do robô	S	-	-
Lewis e Fagg		12	Locomoção suave	-	S	-
Fukuda *et al.*	2	5;7;10	Locomoção suave, Estabilidade	-	S	-
Farritor e Dubowsky	6	2	Seleccionar acções pré-definidas (distância e energia)	-	-	-

Tabela 4.4. Aplicações de AGs na locomoção de robôs

4.5.5 Controlo de *grippers* robóticos

Os AGS podem ser aplicados no controlo de *grippers*. Um exemplo disso consiste no controlador proposto por Erkmen e Durna [119] para manusear objectos através de uma garra mecânica com cinco dedos. Os objectivos do mecanismo são: (1) segurar o objecto até a posição final pré-definida; (2) gerar a sequência de configurações óptimas do ponto de vista da manipulabilidade; e (3) manter a estabilidade sem perder o contacto com o obstáculo. Outro exemplo é desenvolvido por Udawatta *et al.* [120] com o fim de obter a acção de controlo óptima para um determinado intervalo de tempo. Para este fim, é usado um conjunto de controladores elementares, dependendo dos elos ou variáveis de estado a controlar. Por outro lado, Yue e Henrich [121] apresentam um método para manusear um objecto deformável de modo a evitar vibrações bruscas. Na sua resolução usam um AG para determinar os parâmetros que permitem ajustar o movimento da garra mecânica.

4.5.6 Calibração de manipuladores

Os AGs são também aplicados na calibração de robôs. Um trabalho nesta área foi o proposto por Zhuang *et al.* [122, 123]. O trabalho tem como objectivo estimar os parâmetros cinemáticos de um manipulador, através de AGs, tendo em vista a sua calibração. O objectivo consiste em encontrar um conjunto de configurações onde o erro cinemático observável, na posição e na orientação do manipulador, seja mínimo. Para esse efeito, cada configuração gerada é movimentada para uma nova posição/orientação e o erro, entre a nova posição e a posição desejada, é medido através de duas câmaras. Outro trabalho é o de Calafiore *et al.* [124] que estudaram vários aspectos relacionados com a calibração dinâmica, de um manipulador robótico, desde a geração de trajectórias até à aquisição de dados, filtragem, estimação dos parâmetros de inércia e de atrito. Por outro lado, um algoritmo híbrido para optimizar o balanceamento do contra-peso de um manipulador robótico é proposto por Coelho *et al.* [125, 126]. Na resolução do problema são optimizados os binários

e as forças de reacção nas juntas usando um algoritmo multi-objectivo para obter a frente de Pareto. O binário deve ser mínimo em todas as posições, velocidades e acelerações.

4.5.7 Aplicações usando aprendizagem

Os AGs são também usados nos sistemas robóticos que seja requerida aprendizagem. Nesta secção são apresentados dois trabalhos. O trabalho desenvolvido por Berlanga *et al.* [127] usa um método co-evolutivo para evoluir controladores neuronais de forma a evitar colisões com obstáculos em robôs Khepera. Esta arquitectura é apresentada como um modo eficiente para aprender a evitar obstáculos usando AGs. O segundo trabalho é proposto por Barberá e Skarmeta [128] que desenvolveram uma plataforma de um robô de modo a executar tarefas de entregas em ambientes fechados, sem qualquer conhecimento prévio do tipo de ambiente. Estes autores usam um AE para ensinar um meta-controlador capaz de executar uma coordenação eficiente do comportamento representado por agentes.

4.5.8 Outras aplicações

Para além das aplicações apresentadas anteriormente existem ainda alguns trabalhos que são descritos de seguida:

- Nakashima *et al.* [129] desenvolveram um método para controlar a vibração de uma carrinha elevador, tendo como objectivo optimizar o ganho, a posição e a aceleração para o movimento da base do robô. A optimização tem ainda em vista a vibração da base do órgão terminal do manipulador bem como a sua posição e sobre-elevação.

- Chapelle e Bidaud [130] usam um algoritmo evolutivo para determinar a equação que modela a cinemática inversa de um manipulador, com juntas do tipo

rotacional e uma geometria arbitrária. A fórmula é determinada pela regressão simbólica evolutiva usando um programa genético.

- Kalra *et al.* [131] apresentam um AG com codificação real, usando a cinemática inversa, para determinar as configurações de um manipulador 2*R*. Para este efeito usa técnicas de nicho para, assim, o algoritmo convergir para os dois mínimos globais.

- Karla e Prakash [132] usam uma rede neuronal para obter a cinemática inversa de um manipulador planar. Os pesos da rede são obtidos usando um AG com codificação real.

4.6 Resumo

Neste capítulo foi apresentado o problema do planeamento de trajectórias e uma classificação do tipo de planeamento. De seguida foram apresentadas formas de representar os obstáculos. Por último, foram descritas várias aplicações robóticas usando AGs, estruturando a revisão do estado da arte nos seguintes grupos:

- geração de trajectórias para robôs móveis,
- geração de trajectórias para manipuladores robóticos,
- selecção e síntese de manipuladores,
- locomoção de robôs,
- controlo de *grippers* robóticos,
- calibração de manipuladores,
- aplicações utilizando aprendizagem,
- outras aplicações.

5 Optimização de trajectórias em tempo real para um manipulador 2*R*

5.1 Introdução

Neste capítulo é proposto um algoritmo tendo em vista a geração de trajectórias para um manipulador 2*R*, em tempo real. O algoritmo tem como objectivo minimizar o deslocamento angular/linear e a oscilação residual da velocidade angular/linear, sem exceder o binário máximo permitido pelos motores das juntas. Para além destas optimizações o algoritmo permite minimizar a duração da trajectória ou da energia requerida ao manipulador quando este executa a tarefa pretendida.

O método proposto usa um AG que calcula previamente as possíveis trajectórias entre todas as células em que o espaço é dividido. As trajectórias encontradas são armazenadas em diversas árvores. Deste modo, quando uma trajectória é necessária, esta é rapidamente construída a partir dessas árvores.

Nesta ordem de ideias o presente capítulo é organizado da seguinte maneira. Na secção 5.2 é descrito o problema que se pretende resolver. De seguida, é proposta a representação usada pelo AG para determinar as trajectórias e os operadores genéticos, respectivamente nas secções 5.3 e 5.4. Na secção 5.5 é apresentado o método de

avaliação das trajectórias e na secção 5.6 é descrito a forma de armazenamento das trajectórias usando um conjunto de árvores. De seguida, é apresentado o algoritmo que permite reconstruir as árvores armazenadas. Na secção 5.8 são ilustrados os resultados de alguns testes. Por último, na secção 5.9 são enumeradas as principais conclusões.

5.2 Formulação do problema

Neste trabalho é considerado um manipulador constituído por dois elos que se pretende mover entre um determinado ponto inicial e um ponto final. Devido ao elevado peso computacional envolvido no planeamento das trajectórias este processo é previamente executado.

Para este fim é proposto um esquema capaz de fornecer uma trajectória óptima em tempo real sempre que for necessário. Assim, é estabelecida uma grelha que divide o ambiente de trabalho do manipulador em várias células. De seguida, usando um AG, são calculadas todas as trajectórias entre cada par de células da grelha. Posteriormente, as trajectórias obtidas são discretizadas, *i.e.*, cada configuração, c_i, de uma trajectória é substituída pela configuração em que o órgão terminal coincide com o centro da célula à qual a configuração c_i pertence. Os resultados são guardados em diversas árvores. Cada célula contribui com uma árvore que guarda a informação de todas as trajectórias cujo órgão terminal incida nessa célula. Obviamente, à medida que se aumenta a resolução da grelha maior é a precisão e a semelhança com a trajectória continua (*i.e.*, com a trajectória não-discretizada).

5.3 Representação da trajectória

Inicialmente é usando um AG para calcular as trajectórias entre todas as células do ambiente de trabalho. Para esse fim, a trajectória é codificada num vector de

valores reais conforme ilustra a figura 5.1. Assim, as trajectórias são codificadas directamente pelo AG num vector de juntas (5.1):

$$\left[\Delta t, (q_1^{(1,T)}, q_2^{(1,T)}), \ldots, (q_1^{(j,T)}, q_2^{(j,T)}), \ldots, (q_1^{(m-2,T)}, q_2^{(m-2,T)})\right] \tag{5.1}$$

A junta i na posição intermédia j e na geração T é designada por $q_i^{(j,T)}$. O vector (5.1) é constituído por $m-2$ configurações e cada configuração é formada pelos dois valores relativos às posições das juntas. Os valores, $q_i^{(j,0)}$, são inicializados no intervalo $]-\pi, +\pi]$. Como as configurações inicial e final se mantêm inalteradas durante a pesquisa do AG, estas não são codificadas no vector (5.1). O parâmetro Δt é introduzido no vector para especificar o tempo entre duas configurações consecutivas. Após o AG ter encontrado uma solução, esta é discretizada e inserida nas árvores correspondentes às células por onde esta trajectória passa.

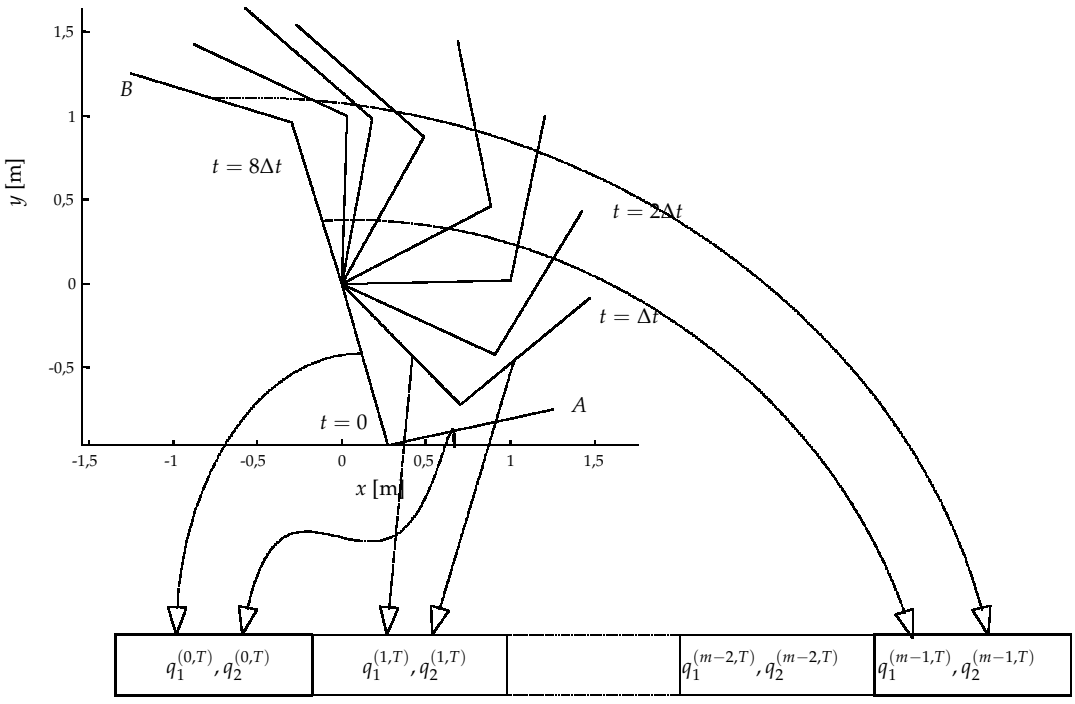


Figura 5.1. Codificação da trajectória

5.4 Operadores usados no algoritmo genético

A população inicial é gerada aleatoriamente e, de seguida, a pesquisa é efectuada através dessa população. Os três operadores usados no planeador genético são: a reprodução, o cruzamento e a mutação, descritos de seguida. O operador de reprodução, por analogia com os sistemas biológicos, baseia-se no desempenho dos elementos da população. Neste caso, é usado uma selecção por torneio-5 [41] de modo a escolher os vectores que darão origem à geração seguinte. Para o operador de cruzamento, os elementos são agrupados em pares, nos quais o operador de cruzamento de ponto simples é executado sobre cada par. O ponto de cruzamento só pode ocorrer entre configurações (*i.e.*, o operador de cruzamento não pode "cortar" uma configuração ao meio). O operador de mutação consiste em modificar vários parâmetros, nomeadamente, o tempo entre duas configurações, Δt, e os ângulos das juntas. Assim, o operador de mutação substitui o valor Δt ou um valor da junta, $q_i^{(j,T)}$, com uma probabilidade de mutação p_{m}. O novo valor $v_i = \{\Delta t, q_i^{(j,T+1)}\}$ é obtido através da expressão $v_i^{(T+1)} = q_i^{(T)} \pm \mathrm{N}(0; 1/\sqrt{2\pi})$, onde N representa a função de distribuição normal.

5.5 Critério de avaliação

O desempenho das trajectórias é medido por dois critérios principais que são usados para decidir o tipo de trajectória pretendido, nomeadamente a duração da trajectória, t_{t}, ou a energia consumida E_{a}. Para além destes índices, são adicionados outros índices para avaliar a qualidade da trajectória do manipulador robótico. Todos estes índices são projectados em funções de penalidade que devem ser minimizadas. Cada índice é calculado individualmente e posteriormente usado na função de aptidão. Inicialmente, antes de ser calculado algum índice, são removidos todos os eventuais deslocamentos ($q_i^{(j+1,T)} - q_i^{(j,T)}$) superiores a π rad. A função de aptidão f adoptada para calcular as trajectórias candidatas é definida pela equação (5.2)

onde os índices f_{CP}, f_q, $f_{\dot{q}}$, f_p , $f_{\dot{p}}$ e f_{sc} são formulados de seguida:

$$f = \beta_1 f_{\mathrm{CP}} + \beta_2 f_{\dot{q}} + \beta_3 f_{\ddot{q}} + \beta_4 f_p + \beta_5 f_{\dot{p}} + \beta_6 f_{\mathrm{sc}} \tag{5.2}$$

A optimização consiste em encontrar um conjunto de parâmetros que minimiza f de acordo com as prioridades atribuídos pelos factores de pesos β_i ($i = 1, .., 6$).

O índice f_{CP} (5.3) fornece uma medida da duração da trajectória t_{t} ou da energia requerida E_{a}, dependendo no critério adoptado. O critério de energia usado para esta análise é a média da energia mecânica requerida durante o tempo total da trajectória t_{t} [133].

$$f_{\mathrm{CP}} = \begin{cases} t_{\mathrm{t}} = m\Delta t, & \text{optimização do tempo} \\ E_{\mathrm{a}} = \sum_{j=0}^{m-1} \sum_{i=1}^{2} |\tau_j \Delta q_i^j|, & \text{optimização da energia} \end{cases} \tag{5.3}$$

O índice f_q é usado para minimizar a distância angular percorrida pelo manipulador robótico dando origem à equação (5.4). Esta equação é usada para optimizar a distância percorrida, pois para uma função $y = g(x)$ o comprimento da função é $\int [1 + (\mathrm{d}g/\mathrm{d}t)^2]\mathrm{d}x$ e, consequentemente, para minimizar a distância da função é adoptada a expressão simplificada $(\mathrm{d}g/\mathrm{d}t)^2\mathrm{d}x$.

$$f_q = \sum_{j=1}^{m-1} \sum_{i=1}^{2} \left(\dot{q}_i^{(j,T)} \right)^2 \tag{5.4}$$

A função $f_{\dot{q}}$ é usada para minimizar a oscilação residual da velocidade angular do manipulador através do critério:

$$f_{\dot{q}} = \sum_{j=1}^{m} \sum_{i=1}^{2} \left(\ddot{q}_i^{(j,T)} \right)^2 \tag{5.5}$$

O critério f_p é introduzido na função de aptidão f para minimizar o comprimento

total da trajectória. Este índice é definido pela equação (5.6) sendo o parâmetro $p^{(w,T)}$ a posição cartesiana w do órgão terminal na geração T e $d(.,.)$ a função que mede a distância entre os dois argumentos.

$$f_p = \sum_{j=1}^{m-1} d\left(p^{(j,T)},\ p^{(j-1,T)}\right)^2 \tag{5.6}$$

A utilização da função $f_{\dot{p}}$ (5.7) na expressão de aptidão é responsável pela minimização da oscilação residual da velocidade cartesiana do órgão terminal.

$$f_{\dot{p}} = \sum_{j=2}^{m-1} \left| d\left(p^{j},\ p^{j-1}\right) - d\left(p^{j-1},\ p^{j-2}\right) \right|^2 \tag{5.7}$$

O índice f_{sc} (5.8) representa a actuação em excesso relativamente ao binário máximo $\tau_{i\ \max}$, que o motor i consegue fornecer. As equações da dinâmica para um manipulador de dois elos podem ser facilmente obtidas pelo Lagrageano [134].

$$f_{\text{sc}} = \sum_{j=0}^{m-1} \left(f_1^j + f_2^j\right) \tag{5.8a}$$

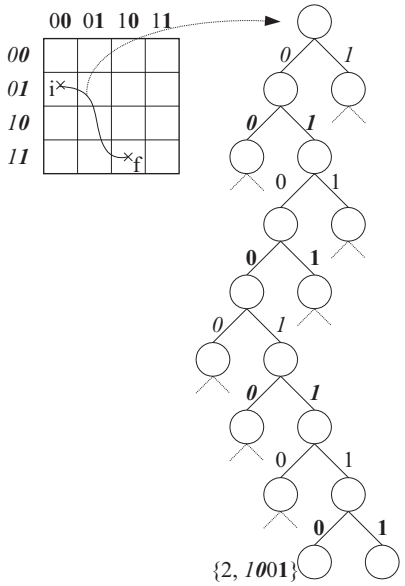
$$f_i^j = \begin{cases} 0, & \text{se } \left|\tau_i^j\right| < \tau_{i\ \max} \\ \left|\tau_i^j\right| - \tau_{i\ \max}, & \text{outros casos} \end{cases} \tag{5.8b}$$

5.6 Representação de trajectórias com árvores

Como foi anteriormente mencionado, o ambiente de trabalho é dividido numa grelha de células. A cada célula resultante dessa divisão é atribuída uma árvore que armazena a informação relativa a todas as trajectórias do manipulador, cujo órgão terminal das configurações incida sobre essa célula. Posteriormente, as trajectórias obtidas são discretizadas, ou seja, cada configuração, c_i, de uma trajectória é substituída pela configuração em que o órgão terminal coincida com o centro da célula à

qual a configuração c_i pertence.

Por exemplo, a figura 5.2 ilustra uma grelha 4×4, a árvore correspondente à célula *0***1**0**1** e a informação da folha correspondente a trajectória entre as células (*0***1**, 0**0**) e (*1***1**, 1**0**). A coordenada da folha, 01001110, é formada pelas células inicial e final: [célula inicial: célula final]. A informação guardada na folha é: (*i*) o número de configurações da trajectória em que o órgão terminal incida na célula e (*ii*) o número da próxima célula. Quando todas as trajectórias forem determinadas e as suas amostras guardadas no grupo das árvores, é chamado o algoritmo podador. Este algoritmo substitui todos os ramos que contenham a mesma informação por uma das suas folhas. No exemplo da figura 5.3 a folha *l* substitui o ramo *n*.



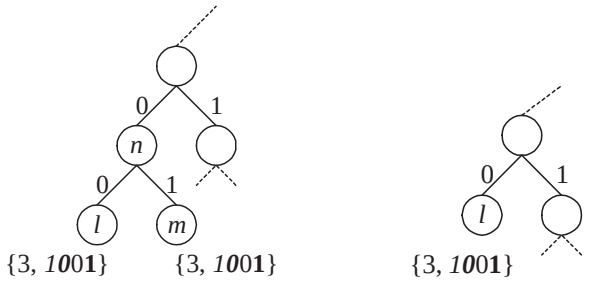
Informação da trajectória:
Ponto inicial: 0100
Ponto final: 1110
coordenada da folha:
01001110

Informação da folha {2, 1001}
{2}: Duas posições do órgão terminal incidem nesta célula
{1001}: Célula da trajectória seguinte

Figura 5.2. Ambiente de trabalho com grelha 4×4 e a árvore da célula 0101

5.7 Reconstrução das trajectórias

A fase em tempo real consiste na reconstrução da trajectória entre a posição actual e a meta desejada. O algoritmo de reconstrução desenvolvido baseia-se nos seguintes pontos:



(a) Árvore inicial

(b) Árvore após ser podada

Figura 5.3. Podador da árvore no nó *n*

1. Calcula a coordenada da folha utilizando as coordenadas dos pontos inicial e final;

2. Usa a árvore correspondente ao ponto inicial como a árvore de pesquisa corrente;

3. Na árvore corrente, segue a coordenada da folha até a alcançar;

4. Quando atingir a folha, fornece o número de pontos que a trajectória permanece na célula. Esta folha também indica a próxima célula (árvore) que será pesquisada;

5. Repetir os passos 3 e 4 até encontrar a folha relativa à célula de destino;

6. Finalmente, a trajectória é reconstruída a partir das configurações das árvores (células) visitadas e dos números previamente devolvidos pelas folhas.

5.8 Resultados das simulações

Esta secção apresenta o resultado de dois conjuntos de simulações. As experiências consistem em deslocar um manipulador robótico desde o ponto inicial A até ao ponto final B, usando dois critérios principais de optimização, nomeadamente, a duração da trajectória t_t e a energia requerida pelo manipulador E_a.

O algoritmo adopta probabilidades de cruzamento e de mutação de $p_c = 0{,}8$ e de $p_m = 0{,}05$, respectivamente, e uma população com 100 elementos de configurações intermédias. Nas experiências são usados vectores de dimensão 7 ($m = 9$) para as trajectórias. O operador de selecção é o torneio-5 com elitismo. Nas simulações, os elos do robô têm um comprimento de 1 m, as juntas são livres de rodar 2π rad e o binário máximo permitido aos actuadores das juntas é de $\tau_{1\ \max} = 16$ Nm e $\tau_{2\ \max} = 5$ Nm para as juntas 1 e 2, respectivamente. O tempo entre duas configurações consecutivas é restringido ao intervalo $0{,}05 \leq \Delta t \leq 1{,}60$ s.

5.8.1 Resultados da primeira simulação

A tabela 5.1 contém a informação referente ao ambiente de trabalho e à trajectória desta simulação. A trajectória consiste em deslocar o manipulador do terceiro até ao primeiro quadrante. A primeira coluna indica o número de células n_C no qual o ambiente de trabalho está dividido, as restantes colunas indicam os pontos iniciais e finais das trajectórias. Estas coordenadas correspondem à discretização dos pontos iniciais e finais da trajectória contínua.

Tabela 5.1. Informação do ambiente de trabalho e da trajectória

Células (n_C)	Ponto inicial A	Ponto final B
4	$(-1{,}00; -1{,}00)$	$(+1{,}00; +1{,}00)$
16	$(-1{,}50; -0{,}50)$	$(+1{,}41; +1{,}41)$
64	$(-1{,}25; -0{,}25)$	$(+1{,}25; +1{,}25)$
256	$(-1{,}13; -0{,}13)$	$(+1{,}38; +1{,}38)$
$+\infty$ (Contínua)	$(-1{,}00; -0{,}20)$	$(+1{,}40; +1{,}40)$

Optimização da duração da trajectória

Esta secção apresenta o resultado da simulação quando se optimiza a duração da trajectória t_t. O intervalo de tempo, entre duas configurações, resultante é de $\Delta t = 0{,}05$ s e os valores resultantes da função de aptidão são de $\{n_C : f\} \equiv \{4 : 200{,}6; 16 :$

117,4; 64 : 40,3; 256 : 25,0; 8 : 21,2}. Da figura 5.4 até à figura 5.7 estão ilustradas as trajectórias resultantes do manipulador para as diferentes discretizações do ambientes de trabalho.

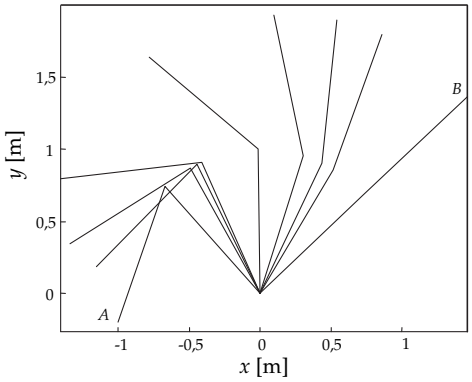
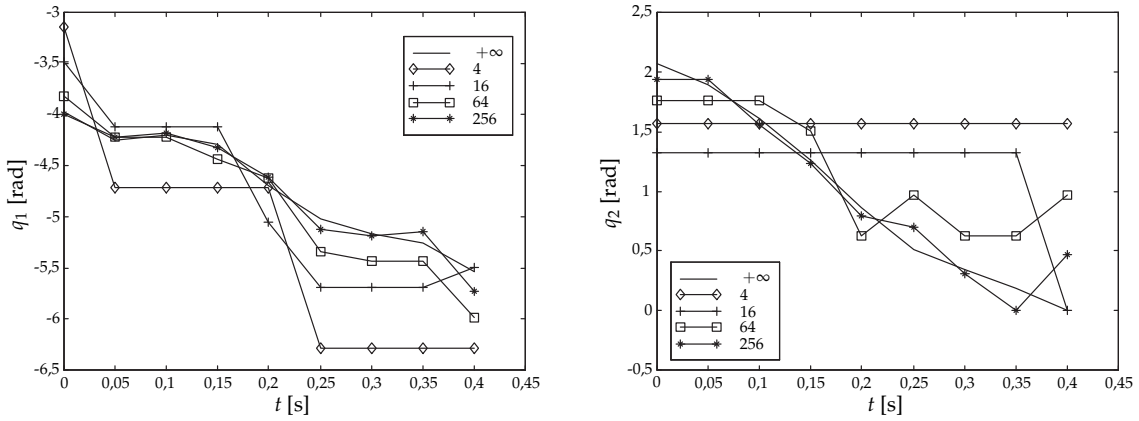


Figura 5.4. Trajectória contínua no plano {x,y}

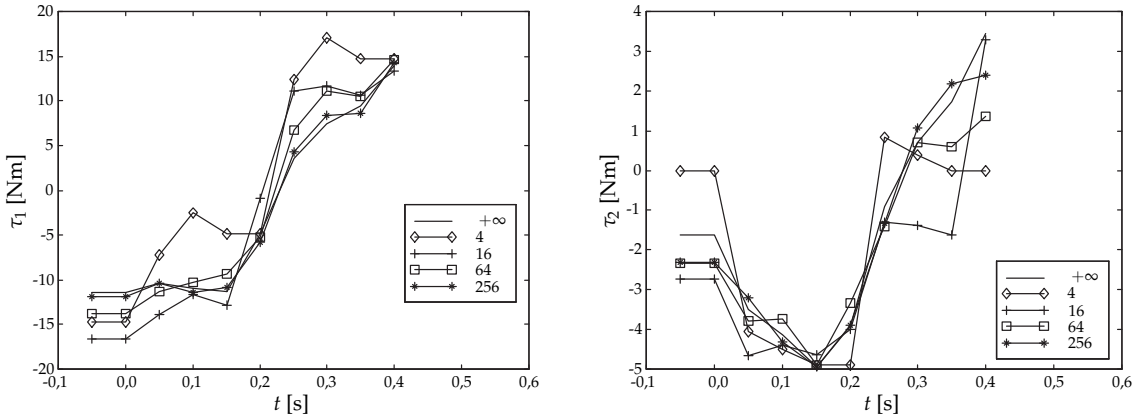


(a) Junta 1 (b) Junta 2

Figura 5.5. Posição das juntas do robô *vs*. tempo, $n_C = \{+\infty, 4, 16, 64, 256\}$

Para os diferentes níveis de discretização do ambiente de trabalho do robô, a junta 1 tem sempre o mesmo tipo de comportamento, enquanto que a trajectória da junta 2 é mais sensível. Nos casos onde $n_C = 64$ e $n_C = 256$ são aqueles em que as trajectórias mais se aproximam da trajectória continua como era de esperar. O tempo obtido $\Delta t = 0{,}05$ s é o valor mais baixo que o AG permite.

Após a quantificação do ambiente de trabalho algumas das trajectórias deixam de



(a) Junta 1 (b) Junta 2

Figura 5.6. Binários das juntas do robô *vs.* tempo, $n_C = \{+\infty, 4, 16, 64, 256\}$

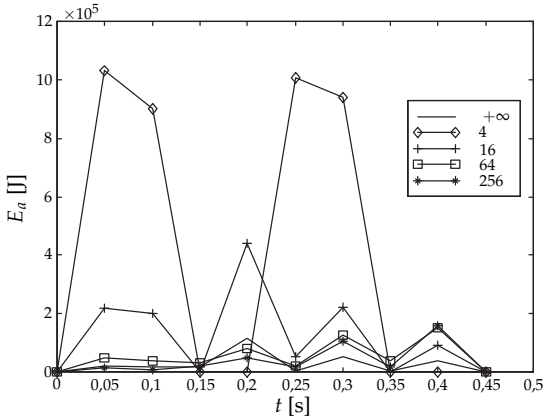


Figura 5.7. Energia $E_a(t)$ *vs.* tempo para a optimização t_t, n_C = $\{+\infty, 4, 16, 64, 256\}$

satisfazer o valor do binário máximo que o motor permite fornecer. De facto, obtemos $\{n_C;\ \tau_1;\ t\} \equiv \{4; 17{,}06; 0{,}30\}$ e $\{n_C;\ \tau_1;\ t\} \equiv \{16; 16{,}60; 0\}$. A tabela 5.2 mostra os resultados de alguns critérios de erro relativamente à energia E_a, nomeadamente o integral quadrático (CEIQ), o integral quadrático multiplicado pelo tempo (CEIQT), o integral absoluto (CEIA) e o integral absoluto multiplicado pelo tempo (CEIAT). O erro consiste na diferença entre a energia requerida pela trajectória quantificada e a trajectória contínua. Em todos os critério o erro da energia diminui com o aumento do valor de n_C, como era de esperar.

Tabela 5.2. Desempenho *vs.* quantificação para a optimização t_t

Critério	$n_C = 4$	$n_C = 8$	$n_C = 16$	$n_C = 256$
CEIQ ($\times 10^{11}$)	1,81	0,11	0,01	0,01
CEIQT ($\times 10^{10}$)	1,12	0,17	0,01	0,01
CEIA ($\times 10^{05}$)	1,98	0,50	0,17	0,15
CEIAT ($\times 10^{04}$)	1,33	0,66	0,10	0,10

Optimização da energia requerida pelo manipulador

Esta secção apresenta as simulações para a optimização quando $f_{CP} = E_a$. O tempo entre duas trajectórias resultantes é de $\Delta t = 0{,}22$ s e o valor de aptidão é de $\{n_C : f\} \equiv \{4 : 93{,}1; 16 : 63{,}2; 64 : 33{,}0; 256 : 13{,}6; 8 : 12{,}3\}$. Os resultados estão ilustrados nas figuras 5.8-5.11.

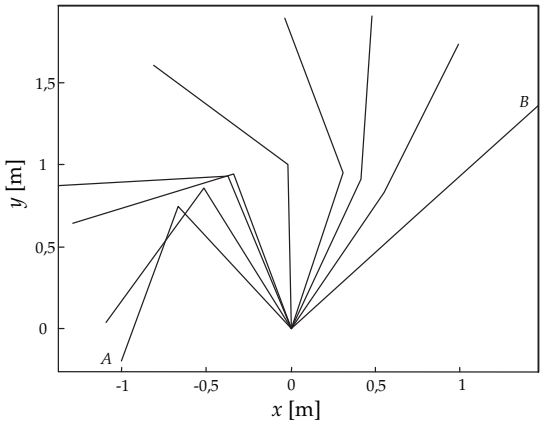
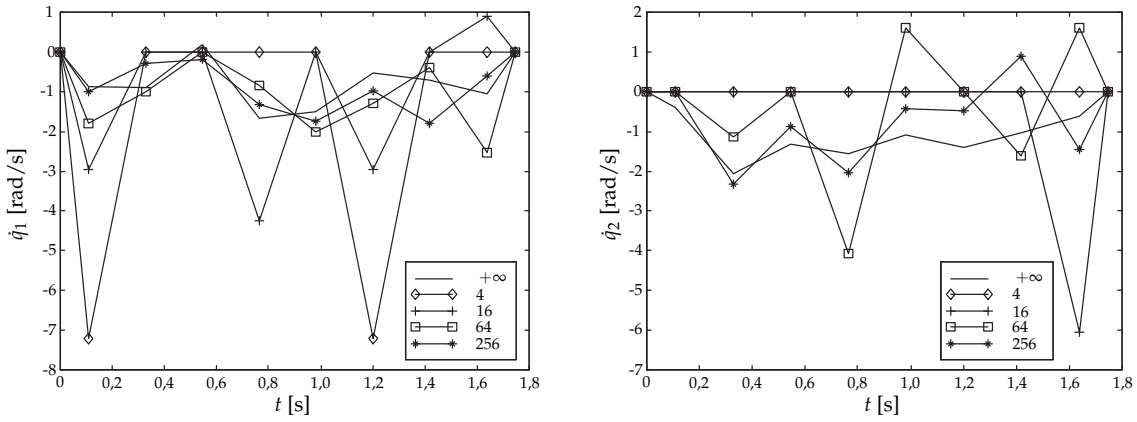


Figura 5.8. Trajectória contínua no plano {x,y}

Nesta secção, apenas as experiências com $n_C = 256$ e $n_C = 64$ satisfazem os limites dos binários requeridos. De facto, foi obtido $\{n_C;\ \tau_1;\ t\} \equiv \{4; 17{,}1; 1{,}54\}$ e $\{n_C;\ \tau_1;\ t\} \equiv \{16; -16{,}1; 0\}$.

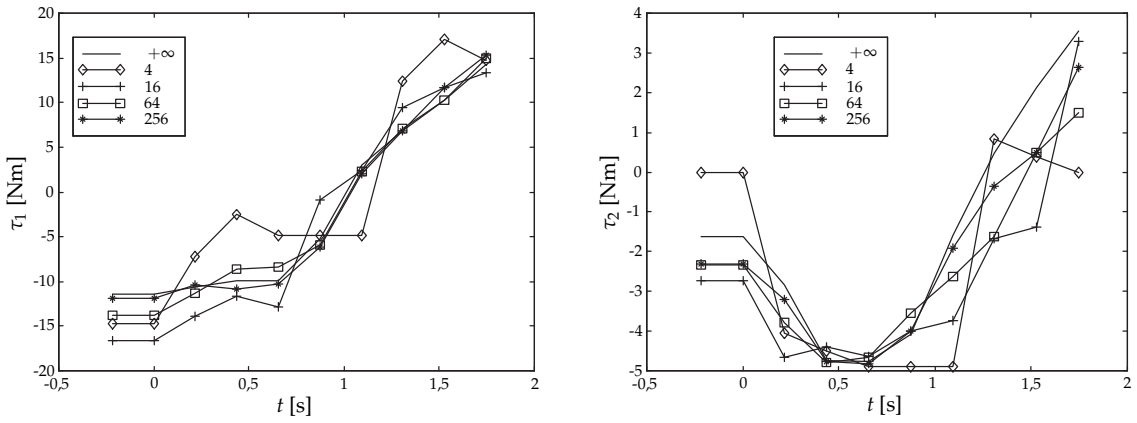
5.8.2 Resultados da segunda simulação

Esta secção apresenta outra simulação com uma trajectória diferente, do segundo para o quarto quadrante, onde os dados se encontram na tabela 5.4. Esta simulação



(a) Junta 1 (b) Junta 2

Figura 5.9. Velocidade angular das juntas do robô *vs.* tempo, $n_{\mathrm{C}} = \{+\infty, 4, 16, 64, 256\}$



(a) Junta 1 (b) Junta 2

Figura 5.10. Binário das juntas do manipulador *vs.* tempo, $n_{\mathrm{C}} = \{+\infty, 4, 16, 64, 256\}$

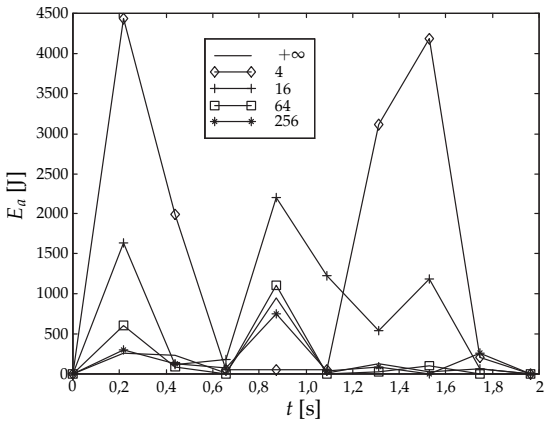


Figura 5.11. Energia $E_{\mathrm{a}}(t)$ *vs.* tempo para a optimização E_{a}, n_{C} = {$+\infty, 4, 16, 64, 256$}

Tabela 5.3. Desempenho *vs.* quantificação para a optimização E_{a}

Critério	$n_{\mathrm{C}} = 4$	$n_{\mathrm{C}} = 8$	$n_{\mathrm{C}} = 16$	$n_{\mathrm{C}} = 256$
CEIQ ($\times 10^7$)	1,04	0,14	0,004	0,002
CEIQT ($\times 10^6$)	2,30	0,56	0,02	0,01
CEIA ($\times 10^3$)	3,10	1,25	0,20	0,15
CEIAT	872,53	479,50	90,78	78,18

usa os mesmos parâmetros que a simulação anterior.

Tabela 5.4. Informação do ambiente de trabalho e da trajectória

Células (n_{C})	Ponto inicial A	Ponto final B
4	$(-1{,}00; +1{,}00)$	$(+1{,}00; -1{,}00)$
16	$(-1{,}41; +1{,}41)$	$(+1{,}50; -0{,}50)$
64	$(-1{,}25; +1{,}25)$	$(+1{,}25; -0{,}75)$
256	$(-1{,}38; +1{,}13)$	$(+1{,}13; -0{,}88)$
$+\infty$ (Contínua)	$(-1{,}25; +1{,}25)$	$(+1{,}25; -0{,}75)$

Nas secções seguintes são apresentados os resultados para a optimização da duração da trajectória t_{t} e para a energia requerida pelo manipulado E_{a}.

Resultados da optimização da duração da trajectória

Esta secção apresenta os resultados da simulação quando se optimiza a duração da trajectória, t_{t}. Analogamente à secção 5.8.1, o intervalo de tempo resultante entre duas configurações é de $\Delta t = 0{,}05$ segundos. Os valores resultantes da função de aptidão são de $\{n_{\mathrm{C}} : f\} \equiv \{4 : 353{,}3; 16 : 179{,}4; 64 : 40{,}8; 256 : 29{,}9; 8 : 26{,}3\}$. Nas figuras 5.12-5.15 encontram-se as trajectórias resultantes do manipulador para as diferentes discretizações do ambiente de trabalho.

Também nestas simulações, para os diferentes níveis de discretização, a junta 1 tem sempre o mesmo tipo de trajectória, enquanto que a trajectória da junta 2 é mais sensível. Também aqui, os casos $n_{\mathrm{C}} = 64$ e $n_{\mathrm{C}} = 256$ são os que se aproximam mais

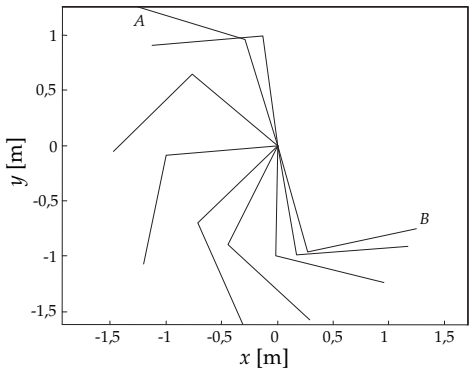
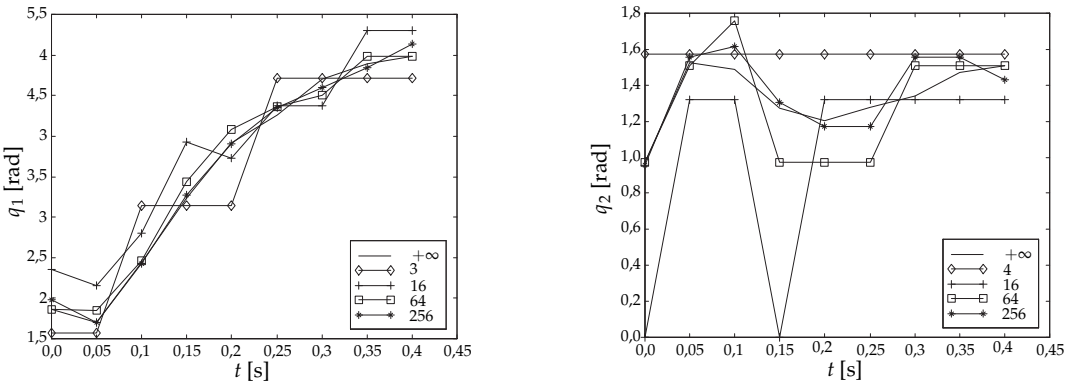


Figura 5.12. Trajectória contínua no plano $\{x,y\}$



(a) Junta 1 (b) Junta 2

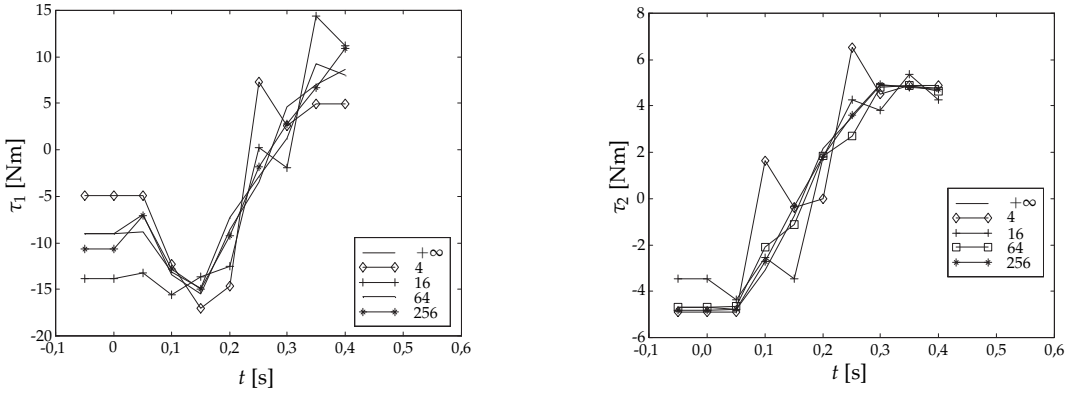
Figura 5.13. Posição das juntas do robô *vs.* tempo, $n_C = \{+\infty, 4, 16, 64, 256\}$

da trajectória continua como seria de esperar.

Nesta simulação, após a quantificação do ambiente de trabalho algumas das trajectórias também violam o binário máximo que os motores podem fornecer. De facto, foi obtido $\{n_C;\ \tau_1;\ t\} \equiv \{4; 17{,}60; 0{,}15\}$, $\{n_C;\ \tau_1;\ t\} \equiv \{4; 16{,}53; 0{,}25\}$ e $\{n_C;\ \tau_2;\ t\} \equiv \{4; 5{,}38; 0{,}35\}$. A tabela 5.5 apresenta os resultados dos critérios de erro referidos anteriormente.

Resultados da simulação quando se optimiza a energia E_a

Esta secção apresenta as simulações quando se optimiza o critério E_a. O tempo entre duas configurações sucessivas obtido é de $\Delta t = 0{,}71$ s e o valor de aptidão é



(a) Junta 1 (b) Junta 2

Figura 5.14. Binários das juntas do robô *vs.* tempo, $n_C = \{+\infty, 4, 16, 64, 256\}$

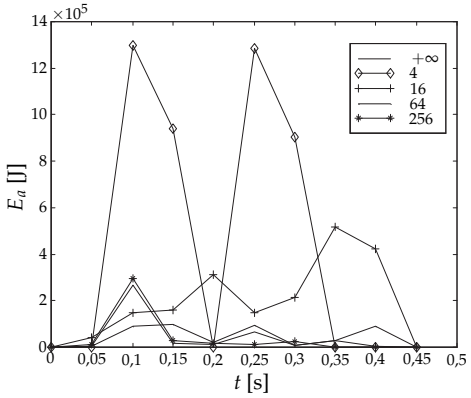


Figura 5.15. Energia $E_a(t)$ *vs.* tempo para a optimização t_t, n_C = {+∞, 4, 16, 64, 256}

Tabela 5.5. Desempenho *vs.* quantificação para a optimização t_t

Critério	$n_C = 4$	$n_C = 8$	$n_C = 16$	$n_C = 256$
CEIQ ($\times 10^{11}$)	2,10	0,29	0,02	$2{,}5 \times 10^{-3}$
CEIQT ($\times 10^{10}$)	1,65	0,15	0,03	$8{,}0 \times 10^{-4}$
CEIA ($\times 10^{5}$)	2,06	0,90	0,19	0,08
CEIAT ($\times 10^{4}$)	1,71	0,63	0,23	0,05

de $\{n_C : f\} \equiv \{4 : 93{,}1; 16 : 93{,}1; 64 : 65{,}6; 256 : 17{,}1; 8 : 12{,}8\}$. Os resultados estão ilustrados nas figuras 5.16-5.19.

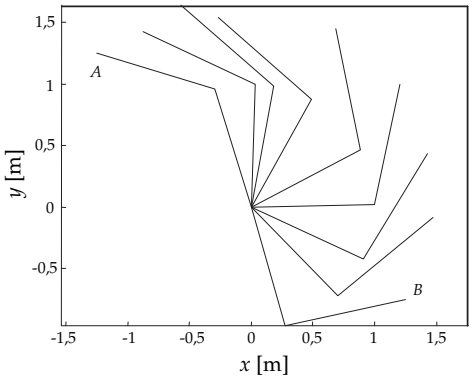
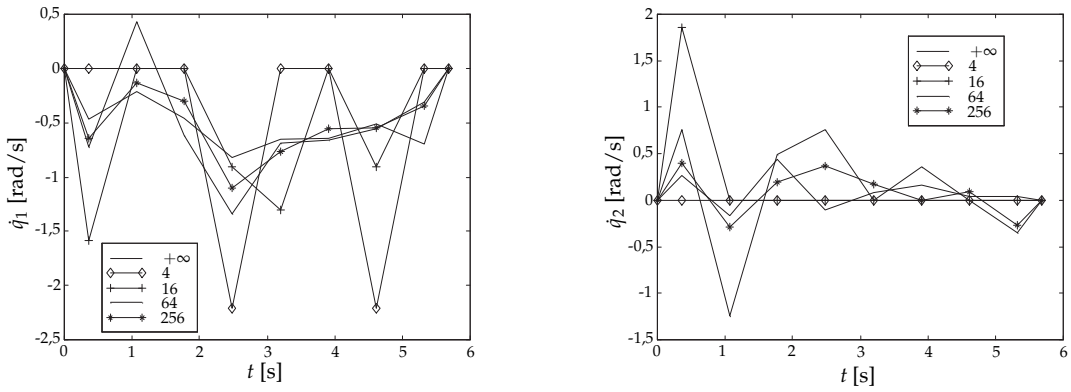


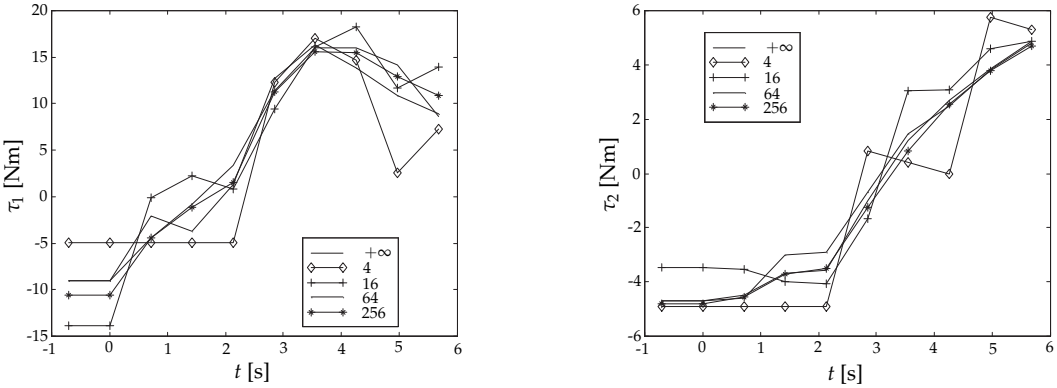
Figura 5.16. Trajectória contínua no plano $\{x,y\}$



(a) Junta 1 (b) Junta 2

Figura 5.17. Velocidade angular das juntas do robô *vs.* tempo, $n_C = \{+\infty, 4, 16, 64, 256\}$

Nesta secção, apenas a experiência com $n_C = 256$ satisfaz os limites dos binários requeridos. De facto, para os restantes valores foi obtido $\{n_C;\ \tau_1;\ t\} \equiv \{4; 17{,}06; 3{,}55\}$, $\{n_C;\ \tau_2;\ t\} \equiv \{4; 5{,}74; 4{,}97\}$, $\{n_C;\ \tau_2;\ t\} \equiv \{4; 5{,}29; 5{,}78\}$, $\{n_C;\ \tau_1;\ t\} \equiv \{16; 16{,}10; 3{,}55\}$, $\{n_C;\ \tau_1;\ t\} \equiv \{16; 18{,}22; 4{,}26\}$ e $\{n_C;\ \tau_1;\ t\} \equiv \{64; 16{,}27; 3{,}55\}$.



(a) Junta 1 (b) Junta 2

Figura 5.18. Binário das juntas do manipulador *vs.* tempo, $n_{\mathrm{C}} = \{+\infty, 4, 16, 64, 256\}$

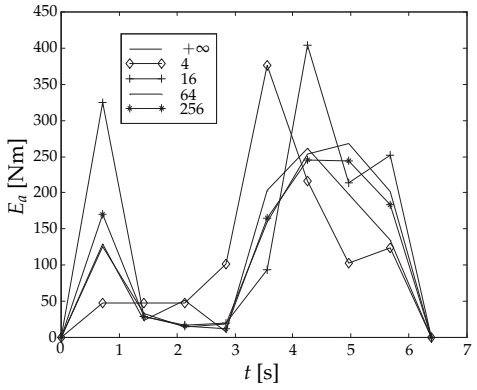


Figura 5.19. Energia $E_{\mathrm{a}}(t)$ *vs.* tempo para a optimização E_{a}, n_{C} = $\{+\infty, 4, 16, 64, 256\}$

Tabela 5.6. Desempenho *vs.* quantificação para a optimização E_{a}

Critério	$n_{\mathrm{C}} = 4$	$n_{\mathrm{C}} = 8$	$n_{\mathrm{C}} = 16$	$n_{\mathrm{C}} = 256$
CEIQ ($\times 10^4$)	6,81	5,16	0,94	0,23
CEIQT ($\times 10^4$)	2,60	4,03	0,33	0,22
CEIA	502,1	375,3	178,9	82,4
CEIAT	379,2	214,4	122,3	74,1

5.8.3 Análise dos resultados

As trajectórias robóticas encontradas são satisfatórias pois têm um comportamento suave, tanto no deslocamento como na velocidade, particularmente no caso contínuo. Para os diferentes tipos de optimização (*i.e.*, t_t e E_a) são obtidos valores numéricos para os índices que parecem ser próximos dos mínimos globais. Por outro lado, à medida que o número de células aumenta são obtidos resultados próximos do caso contínuo. De facto, em termos práticos pode dizer-se que para $n_C = 64$ os resultados são praticamente idênticos ao caso contínuo. No que concerne ao tempo de reconstrução das trajectórias este é desprezável tornando esta aproximação adequada para implementações em tempo real.

5.9 Sumário e conclusões

Foi apresentado, um planeador de trajectórias em tempo real, baseado na cinemática e na dinâmica de manipuladores robóticos. Uma vez que o AG usa a cinemática directa, os pontos singulares não constituem problema. O planeador é constituído por duas fases, a primeira efectuada previamente, calcula todas as trajectórias possíveis entre as células, a segunda etapa corresponde à reconstrução em tempo real de uma trajectória requerida. O algoritmo é capaz de determinar uma trajectória com uma oscilação residual reduzida quer no deslocamento quer na velocidade do manipulador robótico. O planeador pode fornecer as trajectórias com duração ou energia requerida pequena dependendo da optimização solicitada.

6 Optimização de estruturas robóticas

6.1 Introdução

Neste capítulo é proposto um algoritmo genético para gerar a estrutura robótica de um manipulador que melhor se adeqúe à execução de determinadas tarefas. O algoritmo tem como objectivo determinar a estrutura robótica de modo a que as trajectórias efectuadas tenham deslocamentos e velocidades angulares/cartesianas suaves sem ocorrerem colisões com qualquer obstáculo do ambiente de trabalho.

Na resolução do problema é usado um algoritmo genético hierárquico, isto é, um algoritmo incorporando numa camada superior um AG que optimiza as estruturas robóticas. Por sua vez, este AG invoca outros AGs que se encontram numa camada inferior. Os AGs que se encontram nesta segunda camada permitem determinar e optimizar a trajectória com o fim de medir o desempenho dos vectores da população do AG da camada superior.

Nesta ordem de ideias, na secção 6.2 é descrito o algoritmo hierárquico usado para resolver o problema. Na secção 6.3 é proposto um algoritmo que gera estruturas

com juntas do tipo rotacional. De seguida, na secção 6.4 o algoritmo é generalizado para suportar juntas prismáticas. Na secção 6.5 é proposto um algoritmo para gerar estruturas sob o ponto de vista de optimização multi-objectivo. Por último, na secção 6.8 são enumeradas as principais conclusões que decorrem do trabalho desenvolvido.

6.2 Algoritmo hierárquico

Na resolução do problema em questão é usado um AG hierárquico (AGH), apresentado no algoritmo 6.1 e ilustrado na figura 6.1. O AG principal é usado para determinar a estrutura robótica e é denominado AG das *estruturas*. Num segundo nível são executados dois AGs diferentes. Um deles é executado duas vezes para determinar a configuração inicial e final do manipulador (AG das *configurações*), respectivamente. O segundo AG, designado AG das *trajectórias*, é invocado para determinar as configurações intermédias entre as duas configurações calculadas previamente pelo AG_1 das *configurações*.

```
1 Geração aleatória da população;
2 repetir
3     cruzamento;
4     mutação;
5     fusão;
6     duplicação;
7     Avaliação = AG1(Conf. Inicial)+AG1(Conf. Final)+AG2(Trajectória)
8 até enquanto condição de finalização falsa;
```

Algoritmo 6.1. AGH para a definição da estrutura robótica

Para testar eventuais colisões entre o manipulador e os obstáculos, as configurações da trajectória são discretizadas em diversos pontos. Posteriormente, esses pontos são testados com o fim de verificar se se encontram dentro de algum obstáculo, (*pna* – número de pontos dentro de um obstáculo, ver figura 6.2).

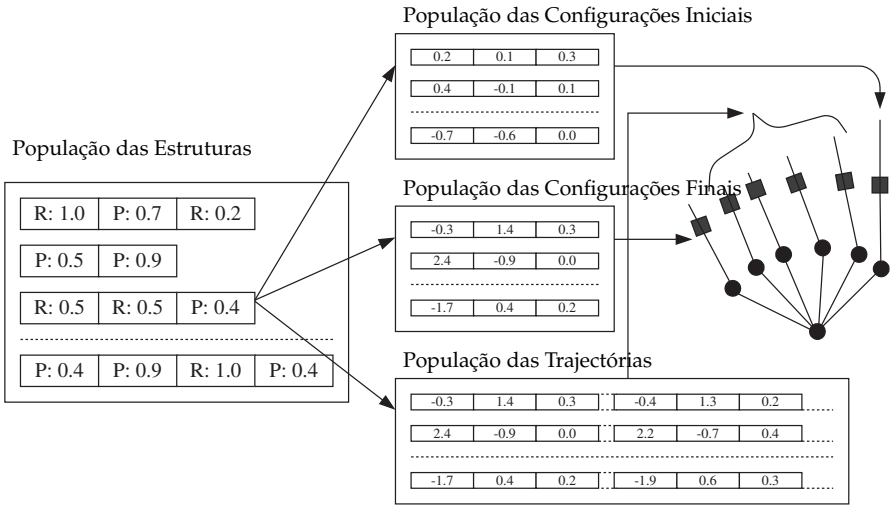


Figura 6.1. Algoritmo hierárquico: Ilustração da topologia populacional adoptada

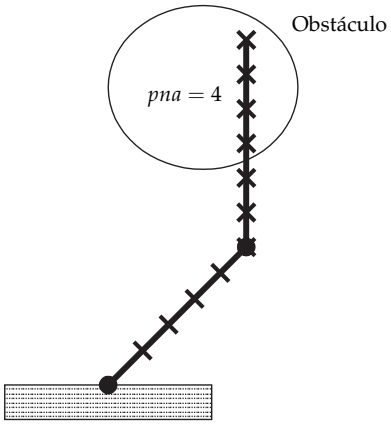


Figura 6.2. Discretização de uma configuração da trajectória, com $nap = 4$

6.3 Síntese de manipuladores com juntas do tipo rotacional

6.3.1 Introdução

Nesta secção são considerados manipuladores robóticos apenas com juntas rotacionais, abreviatura designada por R. Assim, o algoritmo tem como objectivo determinar a estrutura que obtenha o melhor desempenho quando este se desloca entre o

conjunto de pontos A e o conjunto B. Nas experiências são considerados manipuladores de 1 a 4 graus de liberdade (gdl). O comprimento dos elos é restringido ao intervalo $[0;4]$ m, e as juntas são livres de rodar sem restrições em torno dos seus eixos. Consequentemente, o ambiente de trabalho é um circulo com 16 metros de raio, que poderá ter obstáculos circulares e/ou rectangulares.

6.3.2 Representação

A estrutura robótica, na geração T, é codificada pelo vector (6.1) onde l_i é o comprimento do elo i, no intervalo $[0;4]$ m. Com vista a limitar o tempo computacional, o número de gdl é limitado a $k \leq 4$. Neste trabalho todas as variáveis inerentes à optimização das estruturas robóticas são codificados através de valores reais.

$$S = \{l_1^{(T)}, \ldots, l_i^{(T)}, \ldots, l_k^{(T)}\} \tag{6.1}$$

As configurações inicial e final são codificados pelo vector (6.2). Por outro lado, a trajectória é representada, directamente, no espaço das juntas pelo vector (6.3).

$$\{q_1^{(T)}, \ldots, q_k^{(T)}\} \tag{6.2}$$

$$\{(q_1^{(1,T)}, \ldots, q_k^{(1,T)}), \ldots, (q_1^{(j,T)}, \ldots, q_k^{(j,T)}), \ldots, (q_1^{(n-2,T)}, \ldots, q_k^{(n-2,T)})\} \tag{6.3}$$

O vector da trajectória, na geração T, é constituído por $n-2$ genes (configurações) e cada gene é formado por k valores. Sendo $q_i^{(j,T)}$ a variável das juntas i na posição intermédia j. Os valores das juntas $q_i^{(j,0)}$ são inicializados no intervalo $]-\pi;\pi]$. As configurações inicial, $q^{(0,T)}$, e final, $q^{(n-1,T)}$, calculadas previamente pelo AG_1, não são codificadas na trajectória uma vez que se mantêm inalteradas nesta etapa. Por simplicidade e sem restringir o problema, o tempo entre duas trajectórias consecutivas é considerado normalizado de $\Delta t = 1$ s pois pode-se fazer, se necessário, um

reescalonamento dos tempos.

6.3.3 Operadores genéticos utilizados

As populações iniciais são geradas aleatoriamente. E de seguida, a pesquisa é conduzida a partir destas populações. Os operadores usados no planeador são: a reprodução, o cruzamento, a mutação, a fusão e a duplicação, descritos de seguida. No que concerne ao operador de reprodução, as sucessivas gerações de vectores são reproduzidos baseados nos seus valores de aptidão. A selecção dos vectores progenitores que vão participar na geração da nova população é efectuada utilizando a técnica por torneio com cinco elementos, submetida a elitismo. O operador de cruzamento adoptado consiste no operador de ponto simples sendo permitido apenas executar o cruzamento entre genes. O operador de mutação tem a seu cargo várias acções, nomeadamente a modificação do comprimento dos elos e a modificação dos valores da juntas. Assim, o operador de mutação, substitui um valor, de acordo com uma probabilidade, segundo a equação (6.4), onde: φ_i e ψ_i são números uniformemente aleatórios e k_m é um parâmetro que ajusta o grau de mutação.

$$q_i^{(j,T+1)} = q_i^{(j,T)} + k_m \varphi_i \tag{6.4a}$$

$$l_i^{(T+1)} = l_i^{(T)} + k_m \psi_i \tag{6.4b}$$

$$\{\varphi_i, \psi_i\} \sim U[-1;1] \tag{6.4c}$$

Na iteração do AG das *estruturas* dois operadores entram em acção, aleatoriamente. Um duplica um determinado gene enquanto que o outro remove ao acaso um gene com probabilidades p_d e p_r, respectivamente. Estes operadores permitem que a pesquisa não se limite a manipuladores robóticos com o número de juntas constante.

6.3.4 Critério de avaliação

Com o fim de avaliar o desempenho das estruturas, são usados vários critérios para classificar os manipuladores robóticos envolvidos nas operações. Todas as restrições e critérios são introduzidos na função objectivo para serem minimizados. Cada critério é calculado individualmente e posteriormente são agregados na função de aptidão. Assim, a função de aptidão adoptada para medir o desempenho dos robôs candidatos é a seguinte:

$$f = \alpha_1 f_{\mathrm{T}} + \alpha_2 f_{\mathrm{I}} + \alpha_3 f_{\mathrm{F}} \tag{6.5a}$$

$$f_{\mathrm{T}} = \begin{cases} \beta_1 f_q + \beta_2 f_{\dot{q}} + \beta_3 f_p + \beta_4 f_{\dot{p}}, & pna = 0 \\ +\infty, & pna \neq 0 \end{cases} \tag{6.5b}$$

onde: α_i e β_j, $i = 1, 2, 3$; $j = 1, \ldots, 4$; são factores de peso que indicam a importância relativa de cada um dos critérios. As funções f_{I} e f_{F} fornecem uma medida da distância entre os pontos inicial e final desejados e os pontos alcançados pelo manipulador. As funções f_q, $f_{\dot{q}}$, f_p, $f_{\dot{p}}$ e pna são definidas de seguida. A finalidade da optimização consiste em encontrar o conjunto de parâmetros que minimizem f de acordo com os coeficientes α_i pré-seleccionados ($i = 1, 2, 3$).

$$f_q = \sum_{j=0}^{n-1} \sum_{l=1}^{k} \left(\dot{q}_i^{(j\Delta t, T)} \right)^2 \tag{6.6a}$$

$$f_{\dot{q}} = \sum_{j=0}^{n-1} \sum_{l=1}^{k} \left(\ddot{q}_i^{(j\Delta t, T)} \right)^2 \tag{6.6b}$$

$$f_p = \sum_{j=1}^{n-1} d\left(p^j,\ p^{j-1} \right)^2 \tag{6.6c}$$

$$f_{\dot{p}} = \sum_{j=2}^{n-1} \left| d\left(p^j,\ p^{j-1} \right) - d\left(p^{j-1},\ p^{j-2} \right) \right|^2 \tag{6.6d}$$

Tabela 6.1. Lista de trajectórias, para a síntese de manipuladores com juntas rotacionais

Trajectória r_k	Ponto inicial A_i	Ponto final B_i
r_1	$(+2,0;+2,0)$	$(-1,0;+2,0)$
r_2	$(-1,0;+2,0)$	$(+1,0;+1,0)$
r_3	$(-0,5;+0,0)$	$(-1,0;+0,0)$
r_4	$(+3,0;-1,0)$	$(+1,5;-1,0)$

A função, f_q, (equação 6.6a) é usada para minimizar o deslocamento angular percorrido pelo manipulador. O índice, $f_{\dot{q}}$, (equação 6.6b) é responsável pela redução da oscilação residual da velocidade angular. O critério, f_p, (equação 6.6c) é introduzido na função de aptidão para optimizar o deslocamento linear do órgão terminal do manipulador. A função $d(.,.)$ calcula a distância entre os dois argumentos. Por fim, a função , $f_{\dot{p}}$, (equação 6.6d) tem como objectivo diminuir a oscilação da velocidade linear do órgão terminal do manipulador. Por último, a variável *pna* fornece uma medida do conflito existente entre a trajectória e os obstáculos.

6.3.5 Resultados das simulações

Nesta secção são apresentados os resultados de várias simulações que consistem em deslocar o braço robótico desde o ponto inicial A até ao ponto final B (tabela 6.1), para dois tipos de situações:

- O algoritmo optimiza uma estrutura do robô para cada uma das r_k ($k = 1, \ldots, 4$) trajectórias (optimização em série), considerando uma trajectória de cada vez sequencialmente;

- O algoritmo optimiza a estrutura do manipulador para o conjunto de r_k ($k = 1, \ldots, 4$) trajectórias (optimização em paralelo), considerando todas as trajectórias simultaneamente. A estrutura resultante desta optimização realiza todas as trajectórias. Assim, o desempenho da estrutura é a soma individual do desempenho da estrutura quando executa cada uma das trajectórias.

Na resolução do problema proposto, o algoritmo adopta probabilidades de cruzamento e de mutação, respectivamente de $p_c = 0{,}6$ e de $p_m = 0{,}1$. Por seu lado as probabilidades de duplicação e de remoção são de $p_r = p_d = 0{,}1$. Na mutação é usado o parâmetro $k_m = 1{,}8$. A dimensão da população é de 50 elementos para as *estruturas*, de 50 elementos para as *configurações* e de 100 vectores para a população das *trajectórias*. Para as experiências são usados vectores com 10 (ou seja, $n = 12$) configurações intermédias. Os pesos relativos das funções agregadas utilizados são $\alpha_i = \beta_j = 1$ com $i = \{1, \ldots, 3\}$ e $j = \{1, \ldots, 4\}$.

Quando se optimiza apenas uma trajectória não existe qualquer distinção entre os métodos de optimização em série e em paralelo. Assim, na optimização da trajectória r_1 obteve-se o manipulador com a estrutura $S = \{2{,}5324; 1{,}4220\}$. As figuras 6.3 e 6.4 ilustram as configurações sucessivas da trajectória e as correspondentes posições angulares, respectivamente.

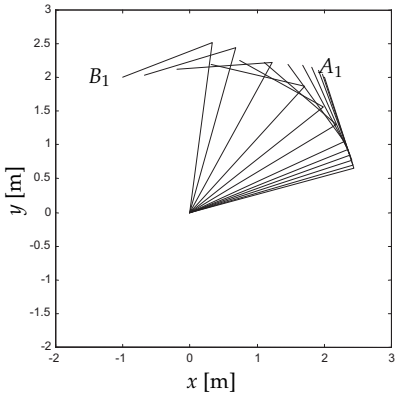


Figura 6.3. Configurações sucessivas da trajectória r_1

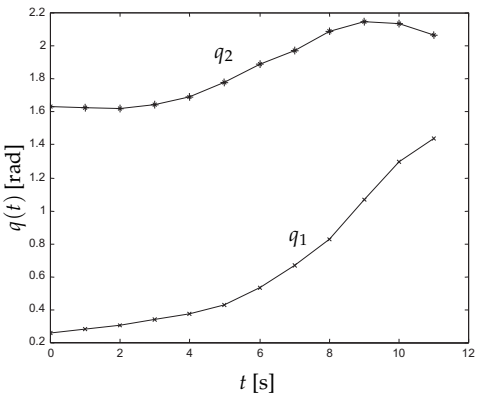


Figura 6.4. Posição das juntas *vs.* tempo

De seguida é realizada a optimização sequencial do grupo de trajectórias, ou seja, é optimizada uma trajectória de cada vez. As estruturas robóticas resultantes da optimização são: $S_1 = \{2{,}7326; 1{,}4446\}$, $S_2 = \{1{,}6166; 1{,}4446\}$, $S_3 = \{1{,}6752; 1{,}4446\}$, $S_4 = \{2{,}3650; 1{,}4446\}$. Os resultados apresentam-se na figuras 6.5 e 6.6. A figura 6.5 representa a trajectória do órgão terminal dos manipuladores enquanto realizam

as quatro trajectórias. A velocidade para o manipulador que realiza a trajectória r_2 encontra-se na figura 6.6.

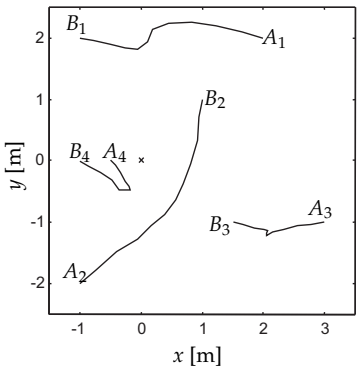


Figura 6.5. Trajectórias do órgão terminal dos manipuladores resultantes da optimização em série

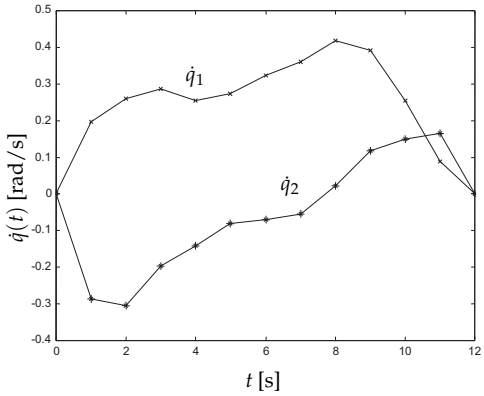


Figura 6.6. Velocidade das juntas *vs.* tempo para a trajectória r_2

Por outro lado, quando o algoritmo é executado para optimizar as 4 trajectórias simultaneamente a estrutura mecânica resultante é $S = \{2{,}6810; 2{,}2563\}$. A figura 6.7 apresenta as trajectórias para o órgão terminal da estrutura. Na figura 6.8 encontra-se a velocidade angular para as juntas do manipulador quando este executa a trajectória r_2.

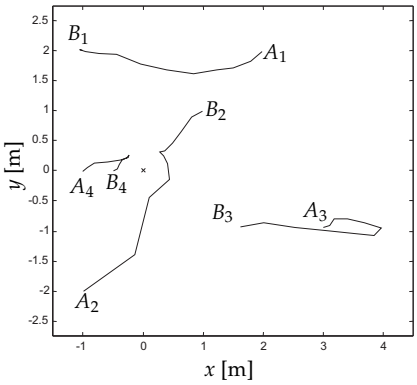


Figura 6.7. Trajectórias para a estrutura resultante da optimização em paralelo

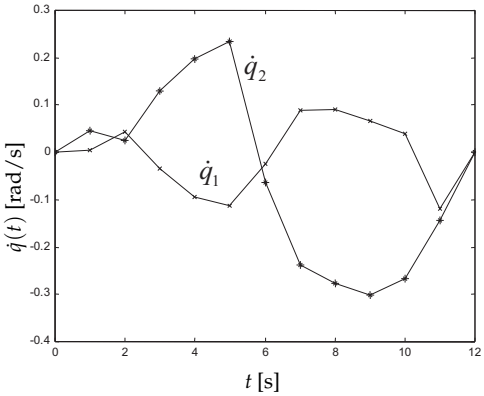


Figura 6.8. Velocidade angular das juntas *vs.* tempo para a trajectória r_2

Quando é inserido um obstáculo de centro $c = (0;2)$ e raio $\rho = 0{,}5$ obtém-se a estrutura RRR com comprimentos $S = \{0{,}8239; 0{,}8369; 1{,}0000\}$. As velocidades das juntas requeridas pelo manipulador para efectuar a trajectória r_1 ilustrada na figura 6.9 encontram-se na figura 6.10.

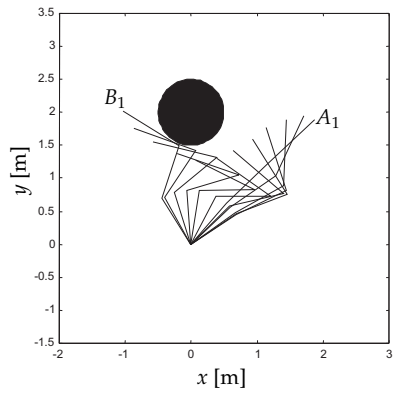


Figura 6.9. Configurações sucessivas da trajectória r_1

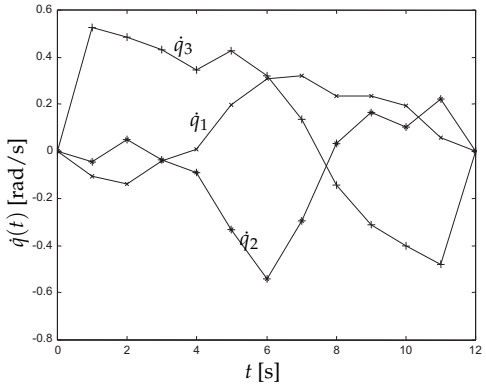


Figura 6.10. Velocidade das juntas *vs.* tempo, trajectória r_1

6.3.6 Análise dos resultados

Os resultados obtidos são satisfatórios pois as soluções evitam os obstáculos e as velocidades das juntas apresentam uma oscilação residual diminuta. Contudo, em ambientes com obstáculos os robôs com mais de dois elos revelam uma capacidade de manipulação superior aos robôs $2R$. Por outro lado, em ambientes sem obstáculos verifica-se que as estruturas $2R$ apresentam o melhor desempenho. Algumas das figuras apresentadas permitem observar que a optimização de certas trajectórias não é a melhor. Isto deve-se ao compromisso entre usar os parâmetros mais adequados para os AGs e a execução do AG em tempo útil. Os resultados das optimizações em série apresentam um desempenho superior ao das optimizações em paralelo devido ao AG ser mais simples. Isto é, quando é executada a optimização em série o AG apenas avalia uma função objecto f relativamente a uma trajectória r_k. Contrariamente, no AG em paralelo o AG agrega todas as funções objectivo f das

trajectórias em simultâneo. Consequentemente, a complexidade do segundo caso é superior, sendo mais difícil encontrar uma solução com desempenho superior ao primeiro caso. Por outro lado, podem existir trajectórias em que o manipulador que apresente o melhor desempenho em cada uma delas tenham estruturas diferentes.

6.4 Síntese robótica com juntas rotacionais e prismáticas

6.4.1 Introdução

Nesta secção, para além de serem consideradas juntas rotacionais (R), são também consideradas juntas do tipo prismático (P). Nas experiências são considerados manipuladores de 1 a 4 gdl. O comprimento de cada elo é restringido ao intervalo $[0;1]$ m. Consequentemente, o ambiente de trabalho é um circulo com 4 metros de raio. Os parâmetros genéticos usados nesta secção são iguais aos adoptados anteriormente em 6.3.

6.4.2 Características do algoritmo genético

Neste caso, a representação deste algoritmo é ligeiramente diferente do anterior, nomeadamente o vector da estrutura S, dado pela expressão (6.7), onde J_i representa o tipo da junta i (R: rotacional; ou P: prismática) e l_i é o correspondente comprimento da junta $i = 1, \ldots, 4$.

$$S_{\{J:l\}} = \{(J_1^{(T)}, l_1^{(T)}), \ldots, (J_i^{(T)}, l_i^{(T)}), \ldots, (J_k^{(T)}, l_k^{(T)})\} \tag{6.7}$$

Os vectores das *configurações* e das *trajectórias* são idênticos aos da secção anterior. No entanto os valores $q_i^{(j,T)}$ são inicializados no intervalo $]-2\pi; 2\pi]$ rad e no intervalo $[0;1]$ m, respectivamente para as juntas do tipo R e P. No que concerne ao

operador de mutação, além das tarefas descritas anteriormente, tem como função adicional permitir comutar o tipo da junta com uma probabilidade p_{com}.

A função de aptidão é dada pela equação (6.8) onde as funções f_q, $f_{\dot{q}}$, f_p, $f_{\dot{p}}$ e pna foram definidas no algoritmo anterior.

$$f = \alpha_1 f_{\text{T}} + \alpha_2 f_{\text{I}} + \alpha_3 f_{\text{F}} \tag{6.8a}$$

$$f_{\text{T}} = \begin{cases} \beta_1 f_q + \beta_2 f_{\dot{q}} + \beta_3 f_p + \beta_4 f_{\dot{p}}, & pna = 0 \\ +\infty, & pna \neq 0 \end{cases} \tag{6.8b}$$

6.4.3 Resultados das simulações

Nesta secção, são apresentados os resultados de várias simulações. As experiências consistem em mover um braço robótico desde o ponto inicial A até ao ponto final B (tabela 6.2), para dois tipos de situações:

- O algoritmo optimiza uma estrutura do robô para cada uma das r_k trajectórias com $k = 1, \ldots, 5$ (optimização em série), considerando uma trajectória de cada vez de forma sequencial;
- O algoritmo optimiza a estrutura do manipulador para o conjunto de r_k trajectórias com $k = 1, \ldots, 5$ (optimização em paralelo), considerando todas as trajectórias simultaneamente.

Tabela 6.2. Lista de trajectórias, para a síntese de manipuladores com juntas do tipo R e P

Trajectória r_i	Ponto inicial A_i	Ponto final B_i
r_1	$(+2{,}0; +2{,}0)$	$(-1{,}0; +2{,}0)$
r_2	$(-1{,}0; +2{,}0)$	$(+1{,}0; +1{,}0)$
r_3	$(+1{,}0; +3{,}0)$	$(-1{,}0; +0{,}0)$
r_4	$(+3{,}0; -1{,}0)$	$(+1{,}5; -1{,}0)$
r_5	$(-1{,}0; -3{,}0)$	$(-1{,}0; -1{,}0)$

O algoritmo adopta para as probabilidades de cruzamento $p_{\mathrm{c}} = 0{,}8$ e de mutação $p_{\mathrm{m}} = 0{,}05$. As probabilidades de duplicação, remoção de elos e de comutação do tipo da junta são $p_{\mathrm{d}} = p_{\mathrm{r}} = p_{\mathrm{com}} = 0{,}01$. Na mutação é usado o parâmetro $k_m = 1{,}8$. A dimensão das populações é de $\{30; 50; 100\}$ elementos para a população das *estruturas*, *configurações* e das *trajectórias*, respectivamente. Nas experiências são usadas trajectórias com 10 ($n = 12$) configurações intermédias e um número de $T = 11$ gerações para cada optimização. A selecção é baseada no torneio-5 com elitismo. Os pesos adoptados nas experiências são $\alpha_i = \beta_j = 1$ com $i = \{1, \ldots, 3\}$ e $j = \{1, \ldots, 4\}$. O espaço de trabalho contém um obstáculo circular de centro $c = (0; 2)$ e raio $\rho = 0{,}5$.

A optimização unicamente da trajectória r_1 resulta na estrutura robótica seguinte:

$$S_{\{J:l\}} = \{[R : 1{,}0000][P : 0{,}8824][P : 0{,}5945][P : 0{,}8366]\}$$

.

As figuras 6.11 e 6.12 mostram os resultados da solução encontrada.

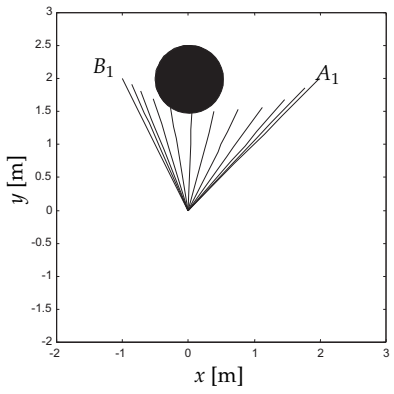


Figura 6.11. Configurações sucessivas, optimização da trajectória r_1

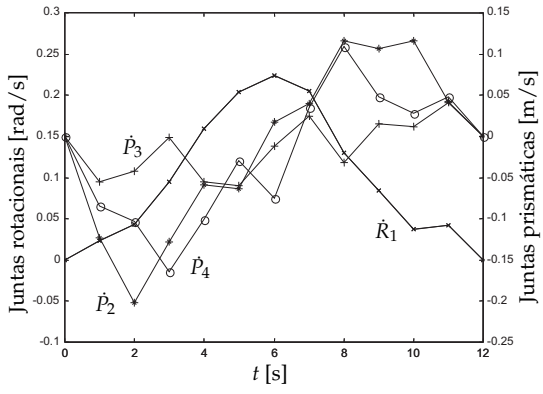


Figura 6.12. Velocidade das juntas *vs.* tempo, optimização da trajectória r_1

Considerando o grupo de trajectórias, na optimização série, resulta nas seguintes estruturas:

$$
\begin{aligned}
S_{1,\{J:l\}} &= \{[R:1{,}0000][P:0{,}7393][P:0{,}5641][R:0{,}6718]\} \\
S_{2,\{J:l\}} &= \{[R:0{,}5632][P:0{,}4643][P:0{,}5460][P:0{,}7479]\} \\
S_{3,\{J:l\}} &= \{[R:0{,}9504][P:0{,}7374][P:0{,}6805][P:0{,}8839]\} \\
S_{4,\{J:l\}} &= \{[R:0{,}9803][P:1{,}0000][P:1{,}0000][P:0{,}8312]\} \\
S_{5,\{J:l\}} &= \{[R:1{,}0000][P:0{,}8347][P:0{,}9961][P:0{,}8087]\}
\end{aligned}
$$

As cinco estruturas optimizadas estão ilustradas na figura 6.13 através das trajectórias do órgão terminal. Na figura 6.14 apresentam-se as posições angulares da trajectória r_3.

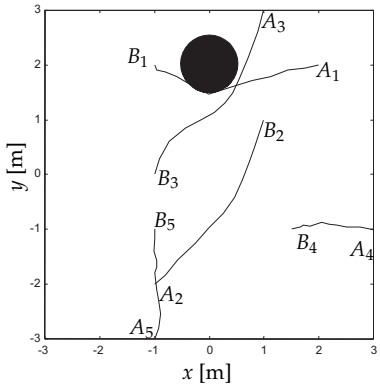


Figura 6.13. Trajectórias do órgão terminal do manipulador

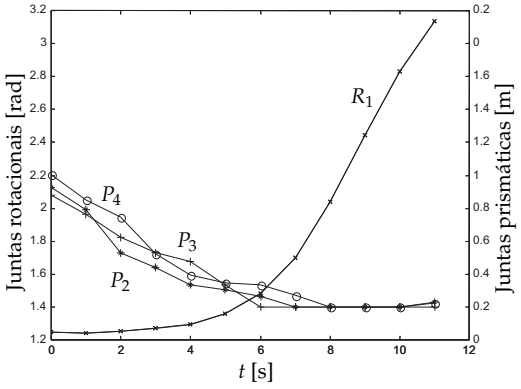


Figura 6.14. Posição das juntas *vs.* tempo, optimização da trajectória r_3

Na figura 6.15 encontra-se o valor de aptidão do melhor elemento da população *versus* o número de gerações. As transições abruptas $\{a, b, c, d\}$ que ocorrem durante a evolução são devidas à mudança da trajectória a optimizar. A transição 'c' corresponde a uma melhoria no valor de aptidão (ao contrário das restantes transições). Esta melhoria deve-se principalmente ao comprimento da nova trajectória $\overline{A_4B_4}$ ser mais curto do que a da trajectória $\overline{A_3B_3}$.

Para a optimização paralela, isto é, para a optimização simultâneas das 5 trajectórias, obtém-se a estrutura seguinte:

$$S_{\{J:l\}} = \{[R:1,0000][P:0,7053][P:1,0000][P:0,6010]\}$$

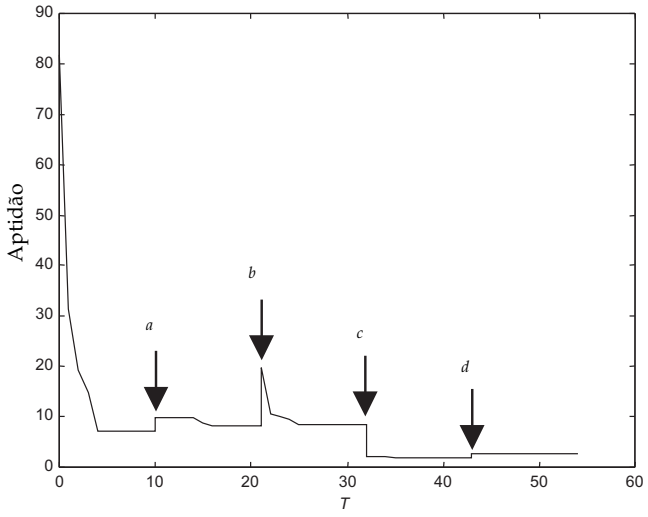


Figura 6.15. Evolução da melhor solução *vs.* número de gerações, na optimização em série

Na figuras 6.16 e 6.17 encontram-se, respectivamente as trajectórias do órgão terminal do manipulador e as posições das juntas para a tarefa 3. Pode reparar-se que a optimização em paralelo não consegue obter resultados tão bons como a série devido ao correspondente aumento da complexidade da função de aptidão.

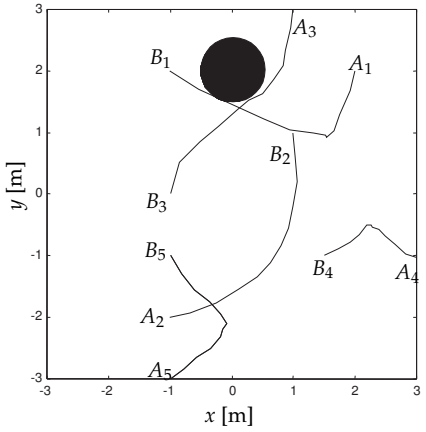


Figura 6.16. Trajectórias do órgão terminal do manipulador

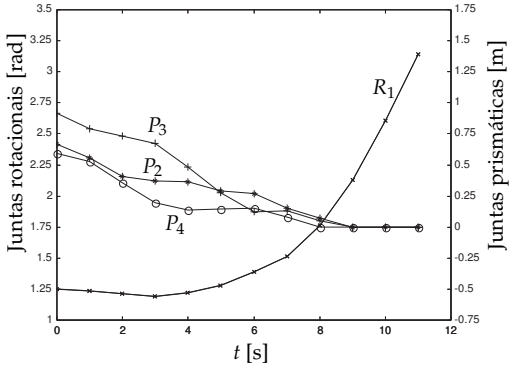


Figura 6.17. Posição das juntas *vs.* tempo, optimização da trajectória r_3

Num segundo conjunto de experiências é inserido um segundo obstáculo rectangular no ambiente de trabalho e optimizam-se as trajectórias de acordo com o esquema

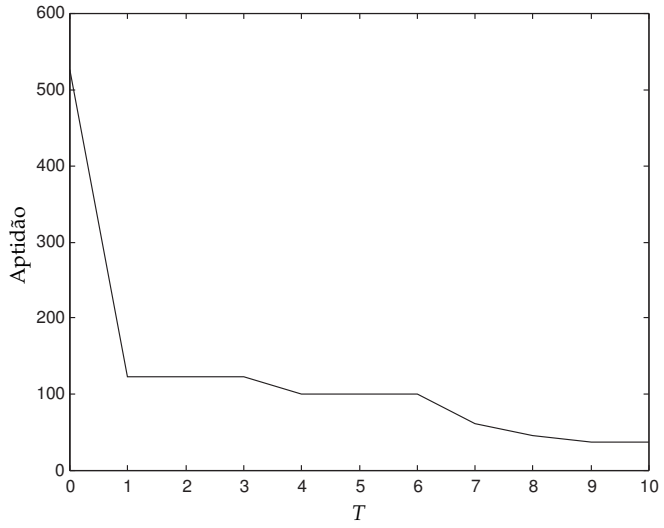


Figura 6.18. Evolução da melhor solução *vs.* número de gerações, na optimização em paralelo

em série. As estruturas resultantes são:

$$
\begin{aligned}
S_{1,\{J:l\}} &= \{[P:0,3875][R:1,0000][P:1,0000][P:0,7689]\} \\
S_{2,\{J:l\}} &= \{[R:0,4888][P:1,0000][P:0,9250]\} \\
S_{3,\{J:l\}} &= \{[R:0,6641][R:0,8629][P:1,0000][P:0,7881]\} \\
S_{4,\{J:l\}} &= \{[P:0,3359][R:0,9353][P:0,9802][P:1,0000]\} \\
S_{5,\{J:l\}} &= \{[P:0,2894][R:1,0000][P:1,0000][P:0,9780]\}
\end{aligned}
$$

Na figura 6.18 encontra-se a evolução da melhor trajectória, na figura 6.19 estão representadas as configurações da trajectória r_2 e na figura 6.20 estão desenhadas as velocidades das juntas em função do tempo para a trajectória r_2. Para as restantes trajectórias o obstáculo não interfere no resultado das optimizações correspondentes.

6.4.4 Análise dos resultados

Conclui-se que aumentando o gdl melhora a capacidade de manipulação do robô, e consequentemente, aumenta a sua capacidade de executar as tarefas, especialmente

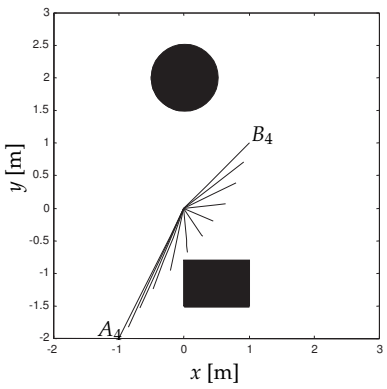


Figura 6.19. Trajectórias do órgão terminal do manipulador

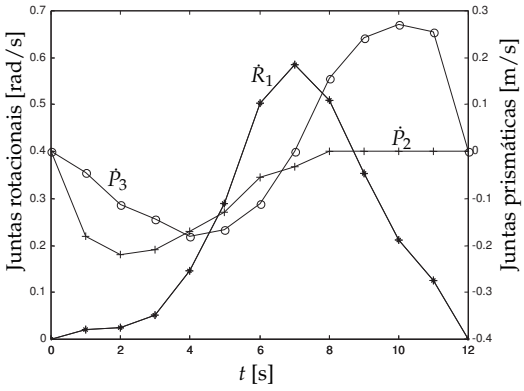


Figura 6.20. Velocidade das juntas *vs.* tempo, optimização da trajectória r_2

quando o ambiente inclui obstáculos.

Foram realizadas diferentes experiências, tanto em ambientes sem obstáculos como em ambientes com diversos tipos de obstáculos localizados em diferentes posições. Os resultados obtidos indicam que para o primeiro eixo do robô é obtida uma taxa de 88,3% de juntas rotacionais e uma taxa de 11,7% de juntas prismáticas. Consequentemente, conclui-se que o robô com a primeira junta rotacional tem um desempenho superior (no sentido de ser mais versátil) para executar diversas tarefas.

6.5 Síntese robótica multi-objectivo

6.5.1 Introdução

Nesta secção é efectuada a síntese de manipuladores robóticos segundo uma perspectiva de optimização multi-objectivo. Assim, neste estudo são considerados manipuladores com juntas rotacionais e prismáticas entre 1 e 4 gdl. O comprimento de cada elo pode variar com incrementos de 0,1 m no intervalo $[0{,}1; 1]$ m, as juntas rotacionais podem rodar em torno dos seus eixos 2π rad e as juntas prismáticas podem variar no intervalo $[0{,}1; 1]$ m.

Na resolução do problema é adoptado um AGMO hierárquico para gerar a estrutura do manipulador. Este algoritmo é composto por 3 AGs uni-objectivo. A representação da estrutura e da trajectória do manipulador usada nesta secção é idêntica à adoptada anteriormente em 6.4.

O AE hierárquico adoptado na resolução do problema é constituído por quatro AGs (ver figura 6.1). Um AGMO determina a estrutura do manipulador. Além disso, para cada estrutura, isto é, para cada elemento da população principal, são executados três outros AGs. Dois desses AGs são usados para determinar a configuração inicial e a final da trajectória, enquanto que o terceiro AG é responsável pela identificação das configurações intermédias da trajectória.

6.5.2 Operadores genéticos multi-objectivo

Os operadores usados neste algoritmo consistem na selecção, cruzamento e mutação, duplicação e fusão descritos de seguida. O operador de selecção utilizado é a selecção por posto. De modo a promover a diversidade da população é utilizado o método de partilha com $\sigma_{\text{partilha}} = 0{,}01$ e $\alpha = 2$ independentemente da dominância das soluções. O operador de cruzamento usado é o SBX [45]. Por seu lado, o operador de mutação pode traduzir-se nas seguintes acções:

- mudar o tipo da junta,
- mudar o comprimento de uma junta prismática,
- mudar o ângulo de uma junta rotacional.

O operador de mutação substitui um gene, com uma determinada probabilidade, de acordo com a equação (6.9) na geração T, onde $\mathrm{N}(\mu, \sigma)$ é a função de probabilidade normal com média μ e desvio padrão σ.

$$q_i^{(j,T+1)} = q_i^{(j,T)} + \mathrm{N}(0, 1/\sqrt{2\pi}) \tag{6.9}$$

Os operadores usados na optimização das estruturas são:

- o operador de duplicação, com probabilidade p_d, que divide um elo em dois com comprimento metade do comprimento inicial;
- o operador de fusão, com probabilidade p_r, que une dois elos consecutivos;
- o operador de mutação, com probabilidade p_{m} que modifica o comprimento de um elo ou valor de uma junta de acordo com a equação (6.10).

Em todas as operações o comprimentos dos elos é salvaguardado. No final de cada geração do AG das *estruturas*, a nova população é seleccionada baseada no esquema de ordenamento MaxiMin (ver capítulo 7).

$$l_i^{(T+1)} = l_i^{(T)} + \mathrm{N}(0, 1/\sqrt{2\pi}) \tag{6.10}$$

6.5.3 Critério de evolução

Para avaliar as soluções são utilizados três índices de desempenho $\{f_{\tau_i}, f_{\tau_f}, f_q\}$ (6.11). A optimização consiste em minimizar estes critérios de modo a encontrar a frente óptima de Pareto. Para calcular o desempenho de cada solução todos os saltos que ocorram em juntas rotacionais, em instantes consecutivos, superiores a π rad ($|q_i^{((j+1)\Delta t,T)} - q_i^{(j\Delta t,T)}| > \pi$), são reajustados, adicionando ou removendo um valor múltiplo de 2π de modo a eliminar esse salto.

$$f_\tau = g \sum_{j=1}^{k} \sum_{i=j}^{k} m_i \sum_{p=1}^{i-1} l_p(\cos(\theta_p)(j \leq p) + 0.5\cos(\theta_i)) \tag{6.11a}$$

$$\theta_p = \sum_{i=1}^{p} q_i \tag{6.11b}$$

$$f_q = \sum_{j=1}^{n} \sum_{l=1}^{k} \left(\dot{q}_l^{(j\Delta t,T)}\right)^2 \tag{6.11c}$$

O binário gravitacional (6.11a) das posições extremas do manipulador é usado com o intuito de minimizar a energia requerida aos actuadores do manipulador quando este realiza longas pausas. Por fim, a distância f_q (6.11c) é usada para minimizar o deslocamento angular percorrido pelo manipulador.

6.6 Resultados da simulação

As experiências realizadas consistem em mover um manipulador robótico entre os pontos $A \equiv \{1{,}0; 0{,}8\}$ e $B \equiv \{-0{,}4; 1{,}2\}$, sendo o número de configurações utilizado de $n = 9$ e o número de gerações de $T^{(c,t,s)} = \{200, 15000, 1200\}$ para as populações das *configurações* (c), *trajectórias* (t) e *estruturas* (s). A dimensão da população consiste em $pop_{\text{size}}^{(c,t,s)} = \{200, 100, 100\}$ e as probabilidades usadas são: $p_{\text{d}} = 0{,}1$, $p_{\text{r}} = 0{,}1$, $p_{\text{c}} = 0{,}8$ e $p_{\text{m}} = 0{,}1$, respectivamente para as probabilidades de duplicação, remoção, cruzamento e mutação.

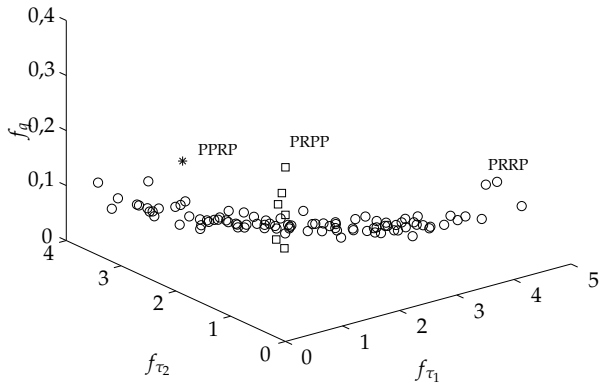
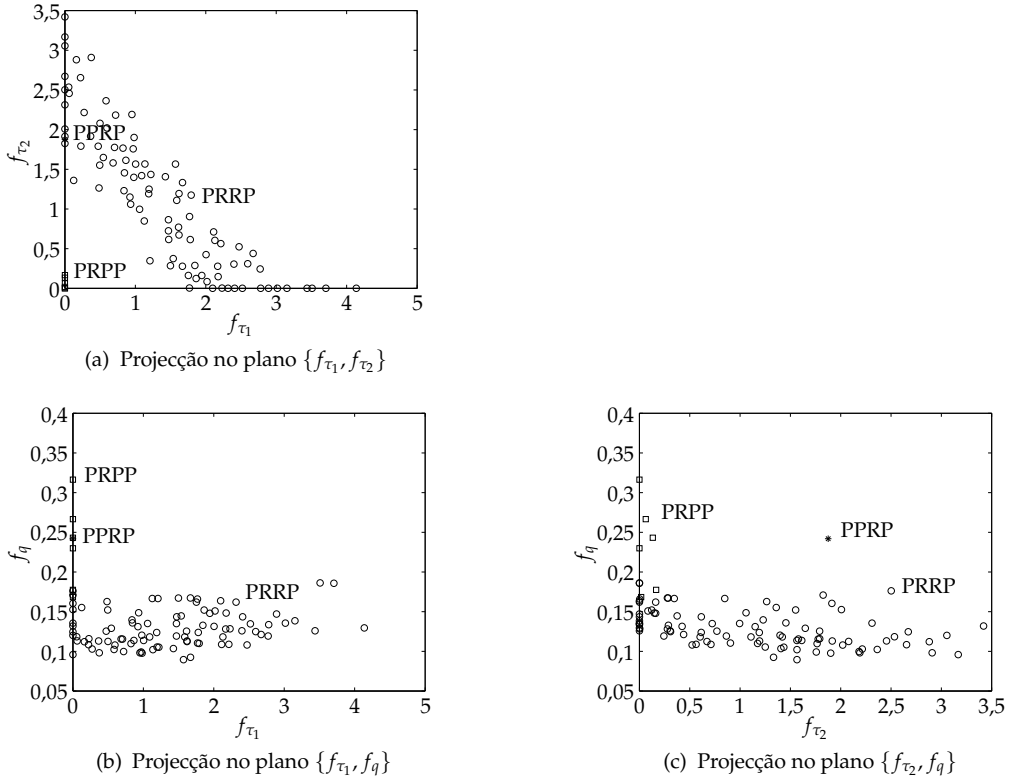


Figura 6.21. Frente óptima de Pareto

O algoritmo determina a frente não-dominada mantendo uma boa distribuição das soluções ao longo da frente de Pareto (ver figuras 6.21 e 6.22), obtendo-se os valores para o índice baseado na distância SP $= 0{,}072$ e para o índice do grafo de distâncias mínimo MDG $= 0{,}122$. Contudo, as soluções ao longo do eixo f_q são poucas, relativamente aos restantes objectivos, devido a ser utilizado o esquema de ordenamento



(a) Projecção no plano $\{f_{\tau_1}, f_{\tau_2}\}$

(b) Projecção no plano $\{f_{\tau_1}, f_q\}$

(c) Projecção no plano $\{f_{\tau_2}, f_q\}$

Figura 6.22. Frente óptima de Pareto $\{f_{\tau_1}, f_{\tau_2}, f_q\}$ e projecções nos planos: $\{f_{\tau_1}, f_{\tau_2}\}$, $\{f_{\tau_1}, f_q\}$ e $\{f_{\tau_2}, f_q\}$

MaxiMin sem harmonização das escalas.

O desempenho das soluções extremas da frente óptima de Pareto varia consideravelmente de acordo com a sua posição na frente. Entre essas soluções extremas existe uma enorme quantidade de soluções que têm um comportamento intermédio entre as primeiras que podem ser elegidas de acordo com a preferência do agente de decisão.

No fim da optimização, o tipo de estruturas presentes na população é apresentado na tabela 6.3, onde as letras 'P' e 'R' significam, respectivamente juntas prismática e rotacional. Na tabela 6.4 encontram-se as estruturas dos robôs com melhor desempenho em pelo menos um dos objectivos. Na figura 6.23, 6.24 e 6.25 estão apresentadas as melhores estruturas obtidas, respectivamente para os objectivos f_{τ_1}, f_{τ_2} e f_q. Nas figuras (*a*) estão ilustradas as sucessivas configurações dos manipuladores

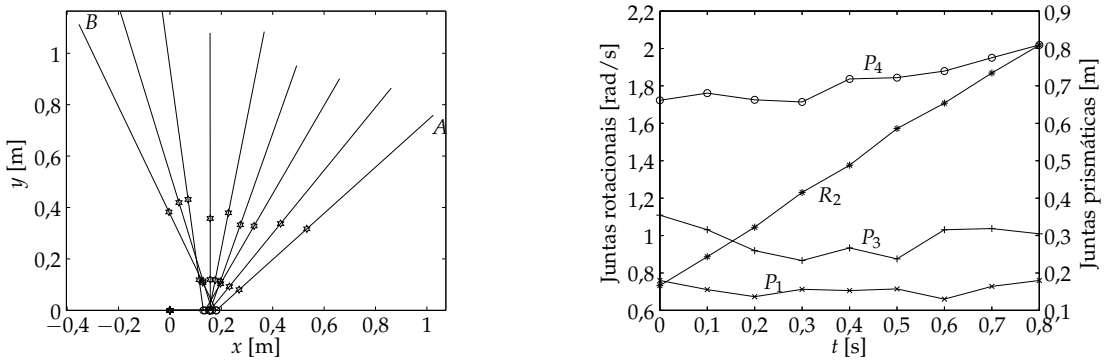
Tabela 6.3. Número de soluções não-dominadas obtidas por tipo de estrutura

Estrutura	Número de soluções
PRRP	93
PRPP	6
PPRP	1

Tabela 6.4. Estruturas com melhor desempenho para um dos objectivos

Objectivo	Estrutura	l_1 cm	l_2 cm	l_3 cm	l_4 cm
f_{τ_1}	PRPP	18	12	36	99
f_{τ_2}	PRPP	10	10	38	96
f_q	PRRP	11	14	39	97

onde um circulo 'o' representa uma junta rotacional e uma estrela '✡' indica uma junta prismática. Por outro lado, nas figuras (*b*) pode ser visualizado a posição das juntas rotacionais e prismáticas. As legendas para as juntas rotacionais e prismáticas encontram-se, respectivamente no lado esquerdo e direito das figuras, sendo J_i a junta $i = \{1, \ldots, 4\}$ do tipo $J = \{\mathrm{R}, \mathrm{P}\}$.



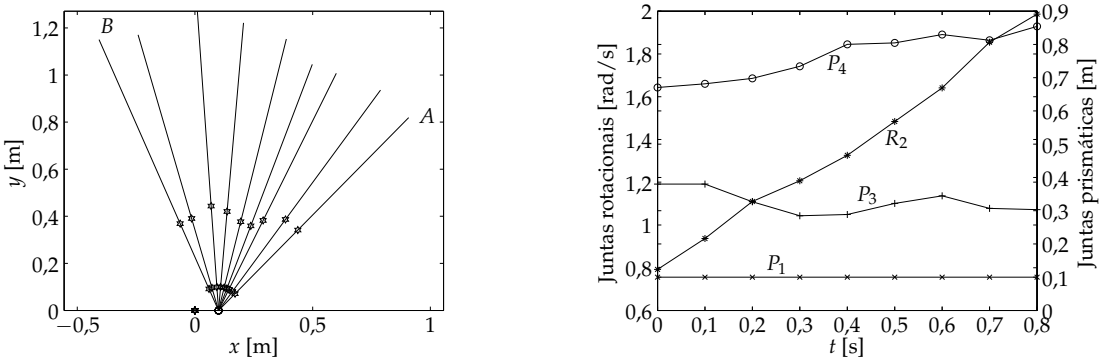
(a) Configurações sucessivas

(b) Posição das juntas *vs.* tempo

Figura 6.23. Manipulador PRPP com melhor desempenho no objectivo f_{τ_1}

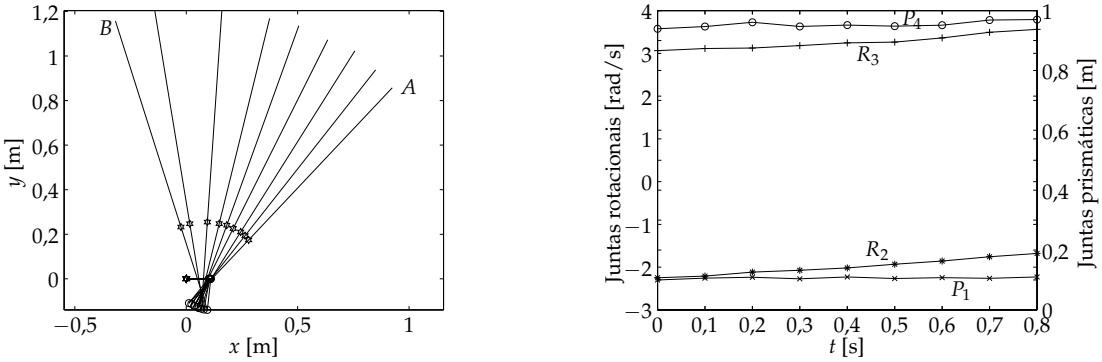
6.7 Análise dos resultados

Analisando a frente de Pareto pode verificar-se que a frente é descontínua. Uma das frentes é composta maioritariamente por robôs do tipo PRRP enquanto que a segunda frente é composta por manipuladores com estrutura PRPP. Por outro lado,



(a) Configurações sucessivas (b) Posição das juntas *vs.* tempo

Figura 6.24. Manipulador PRPP com melhor desempenho no objectivo f_{τ_2}



(a) Configurações sucessivas (b) Posição das juntas *vs.* tempo

Figura 6.25. Manipulador PRRP com melhor desempenho no objectivo f_q

analisando o número final de eixos, conclui-se que quanto maior é o número de elos maior é a capacidade de o robô se deslocar e atingir os objectivos propostos. As estruturas obtidas têm uma junta rotacional junto da base do robô. A partir das figuras (*b*) verifica-se que o deslocamento das juntas é próximo do ideal o que leva a concluir que o algoritmo encontra robôs muito perto das soluções óptimas.

6.8 Conclusões

Neste capítulo foram propostas técnicas baseadas na utilização de AGs para a síntese de robôs, particularmente na resolução dos seguintes problemas:

- Síntese de manipuladores com juntas do tipo rotacional;
- Síntese de manipuladores com juntas do tipo rotacional e prismáticas

A técnica proposta utiliza um AG hierárquico que pode sintetizar manipuladores considerando as trajectórias uma a uma, de forma sequencial, ou todas, de forma simultânea.

A resolução do problema em questão requer a optimização de várias funções objectivo que foram abordadas de duas formas alternativas:

- Agregando as várias funções numa só função;
- Considerando os objectivos separadamente utilizando um AGMO.

Os algoritmos propostos são capazes de encontrar as metas com bom desempenho. Além de que, qualquer obstáculo no ambiente de trabalho não representa uma dificuldade significativa para o algoritmo encontrar a solução. Uma vez que o AG usa a cinemática directa os pontos singulares não levantam qualquer problema. Para concluir, o algoritmo determina a estrutura que mais se adapta a um número de tarefas a desempenhar, mantendo um bom desempenho de manipulação.

O gerador de estruturas multi-objectivo tem a vantagem de fornecer um conjunto de soluções representativas da frente não-dominada. Perante o conjunto de soluções o agente de decisão tem que optar pela solução final a alto nível, isto é, o agente pode visualizar o comportamento de uma solução seleccionada da frente não-dominada. Dependendo do desempenho apresentado em cada objectivo, o agente de decisão pode ir seleccionado outras soluções na sua vizinhança, na direcção do compromisso desejado, até encontrar a solução pretendida. Contrariamente, na optimização uni-objectivo atribui-se um peso a cada objectivo para obter uma solução óptima em vez de uma família de soluções óptimas. Após obter a solução, e se esta não apresentar o desempenho desejado, o agente alterara os pesos da função de aptidão de modo heurístico e repete o processo de optimização. Com este processo o

operador tem que esperar um certo tempo sempre que seja necessário executar de novo o algoritmo.

7 Optimização multi-objectivo de trajectórias robóticas

7.1 Introdução

A maioria dos problemas em engenharia requer a optimização de vários critérios simultaneamente. Quando os problemas são muito complexos incluindo variáveis discretas e contínuas e simultaneamente não existe conhecimento prévio relativo ao espaço objectivo, deve ser usado um método robusto como os AGs.

A optimização de trajectórias para manipuladores robóticos é um problema que envolve a utilização de várias funções objectivo. A resolução deste tipo de problemas pode beneficiar com a optimização multi-objectivo, particularmente quando os objectivos se dizem conflituosos, ou seja, a melhoria de um critério degrada um ou mais dos restantes. Nestes casos, o compromisso óptimo entre os mesmos só é possível considerando os conceitos de não-dominância da teoria da optimização propostos por Pareto e integrados com sucesso nos AGMO. A grande desvantagem de utilização de AGMO, que consiste no aumento do tempo computacional com o número de objectivos considerados, é atenuada progressivamente com o advento de processadores mais rápidos. Isto faz com que as técnicas propostas neste capítulo tenham viabilidade no contexto da aplicabilidade industrial.

Este capítulo trata da optimização de trajectórias de manipuladores robóticos considerando simultaneamente vários objectivos. Assim, nas secções 7.2 e 7.3 é formulado o problema e são estudadas as frentes locais existentes no espaço dos objectivos para um manipulador $2R$. Na secção 7.4 é optimizada a trajectória de um manipulador $2R$ para dois e cinco objectivos. De seguida, nas secções 7.5 e 7.6 encontram-se, respectivamente os resultados para um manipulador $3R$ com 5 objectivos e são descritas outras experiências complementares. Por último, na secção 7.7 encontram-se as principais conclusões.

7.2 Formulação do problema

Pretende-se mover um manipulador robótico entre dois pontos tomando simultaneamente em consideração vários objectivos. Os objectivos considerados neste trabalho são: o deslocamento angular, o deslocamento cartesiano do órgão terminal, a oscilação residual da velocidade angular, a oscilação residual da velocidade linear do órgão terminal e a energia requerida pelo manipulador para efectuar o movimento. Assim, pretende-se determinar um conjunto de soluções não-dominadas pertencentes à frente óptima de Pareto. A solução final será depois escolhida pelo agente de decisão tendo em atenção o compromisso dos objectivos que achar mais apropriado.

A representação do vector, na geração T, que corresponde à trajectória é a seguinte:

$$[\{q_1^{(\Delta t,T)}, q_2^{(\Delta t,T)}\}, \{q_1^{(2\Delta t,T)}, q_2^{(2\Delta t,T)}\}, \ldots, \{q_1^{((n-2)\Delta t,T)}, q_2^{((n-2)\Delta t,T)}\}] \qquad (7.1)$$

O número de configurações da trajectória é de $n = 8$. Como as configurações inicial e final se mantêm inalteradas durante toda a evolução, estas configurações não estão codificadas na trajectória. O tempo considerado entre duas configurações sucessivas é de $\Delta t = 0{,}1$ s. As funções objectivo para calcular o desempenho encontram-se representados na equação (7.2) para minimizar, respectivamente: o deslocamento

angular, $f_{\dot{q}}$, a oscilação residual da velocidade angular, $f_{\ddot{q}}$, o deslocamento e a oscilação residual do órgão terminal (f_p e $f_{\dot{p}}$), e a energia, f_{E_a}, necessária ao manipulador para efectuar o movimento. A função f_{E_a} calcula a energia mecânica média durante a trajectória considerando que a energia negativa não é recuperável pelos motores quando apresentam trabalho negativo. O índice i especifica o número de juntas do manipulador.

$$f_{\dot{q}} = \sum_{j=1}^{n} \sum_{l=1}^{i} \left(\dot{q}_l^{(j\Delta t,T)} \right)^2 \tag{7.2a}$$

$$f_{\ddot{q}} = \sum_{j=1}^{n} \sum_{l=1}^{i} \left(\ddot{q}_l^{(j\Delta t,T)} \right)^2 \tag{7.2b}$$

$$f_p = \sum_{j=2}^{n} d\left(p_j,\ p_{j-1}\right)^2 \tag{7.2c}$$

$$f_{\dot{p}} = \sum_{j=3}^{n} \left\{ d\left(p_j,\ p_{j-1}\right) - d\left(p_{j-1},\ p_{j-2}\right) \right\}^2 \tag{7.2d}$$

$$f_{E_a} = (n-1)T\ P_a = \sum_{j=1}^{n} \sum_{l=1}^{i} |\tau_l . \Delta q_l^{(j\Delta t,T)}| \tag{7.2e}$$

Ao longo deste capítulo são apenas apresentados os resultados no espaço dos objectivos uma vez que a dimensão do espaço dos atributos é da ordem de $i(n-2)$ o que torna complexa a sua análise.

Na resolução do problema é usado um algoritmo multi-objectivo, sendo a selecção efectuada pelo método do posto de Pareto (*Pareto ranking*) proposto por Goldberg e o esquema de partilha com $\sigma_{\text{partilha}} = 0{,}01$ e $\alpha = 2$ efectuado no domínio dos parâmetros. Isto é, a aptidão é atribuída de acordo com o posto da solução, sendo posteriormente afectada pelo valor do seu contador de nicho. Para calcular o contador de nicho, a métrica utilizada entra em linha de conta com todas as soluções da população independentemente do seu posto. Nos operadores de cruzamento e de mutação os elementos que vão fazer parte da geração seguinte são os melhores do conjunto entre os progenitores e os seus descendentes.

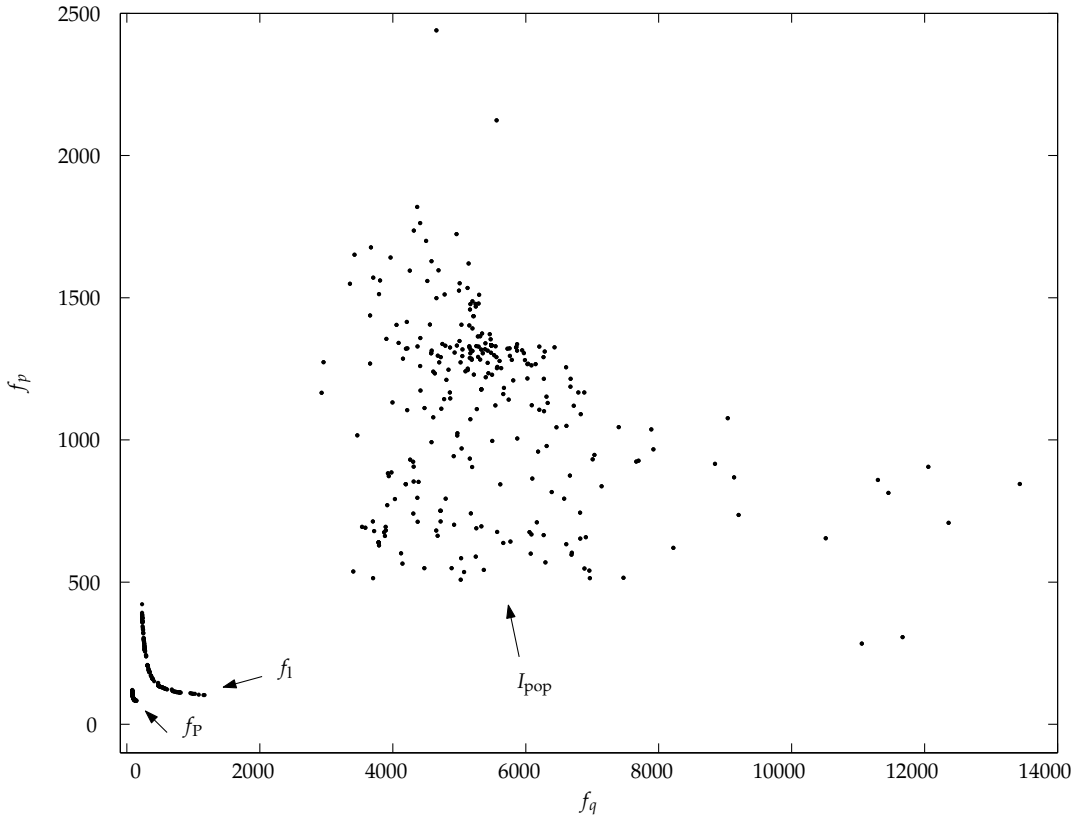
7.3 Convergência do manipulador 2*R*

Nesta secção é estudada a convergência do algoritmo para as frentes locais quando se optimiza a trajectória de um manipulador robótico. Para este fim é realizado um conjunto de experiências que consistem em deslocar um manipulador 2*R* entre o ponto $A \equiv \{1{,}2; -0{,}3\}$ e o ponto $B \equiv \{-0{,}5; 1{,}4\}$. A optimização é efectuada conjuntamente no deslocamento angular e no deslocamento do órgão terminal. As configurações inicial e final do manipulador são determinadas usando a cinemática inversa. Nas simulações usam-se os seguintes parâmetros (definidos previamente): $pop_{\mathrm{dim}} = 300$; $T_{\mathrm{t}} = 1500$; $p_{\mathrm{c}} = 0{,}6$; $p_{\mathrm{m}} = 0{,}05$; $l = 1$ m e $m = 1$ kg.

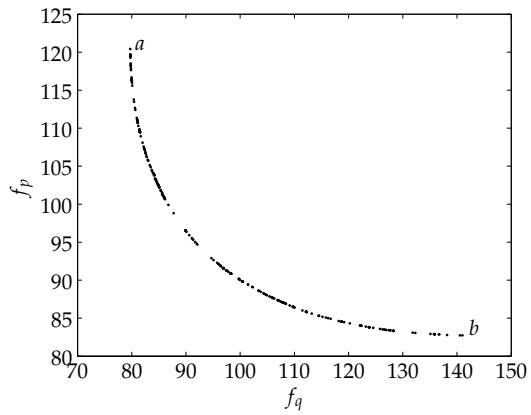
Em todas as simulações efectuadas o algoritmo convergiu para duas frentes que se ilustram na figura 7.1. Uma das frentes é obtida quando o manipulador planar se desloca contornando a sua base no sentido contrário ao dos ponteiros dos relógios (figura 7.2(a)), que corresponde à frente óptima de Pareto representada na figura 7.1(a) como f_{P} e ilustrada separadamente na figura 7.1(b). A outra frente local corresponde ao contorno da base no sentido dos ponteiros do relógio (figura 7.2(b)), denotada na figura 7.1(a) como f_{l} e representada separadamente na figura 7.1(c).

Na figura 7.1(a), está representada a população inicial, I_{pop}, que deu origem à frente de Pareto, f_{P}. Está também ilustrada a frente local obtida a partir de uma outra experiência. Verifica-se que em 80,1% dos testes de simulação efectuados (num conjunto de 100) o algoritmo converge para a frente óptima de Pareto. Nos restantes casos, o algoritmo converge para a frente local.

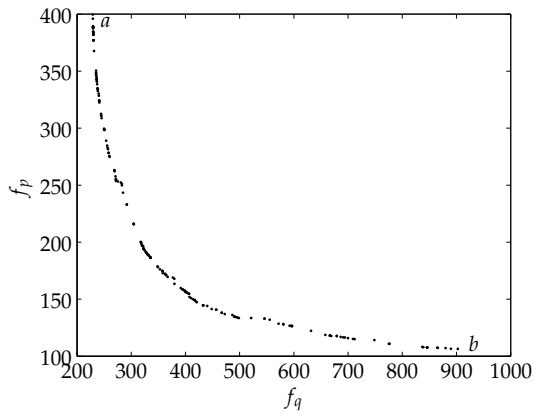
Um dos problemas associados à utilização dos AEs no contexto de problemas de optimização é a chamada convergência prematura, que advém da impossibilidade de utilizar populações de tamanho infinito [135]. Este fenómeno ocorre quando a função objectivo tem algum óptimo local com uma gama de valores larga ou o seu óptimo global encontra-se numa gama bastante pequena. A relação entre a convergência para o óptimo global e a geometria da função é muito importante. Se a



(a) Frente de Pareto (f_P), frente local (f_l) e população inicial (I_pop)

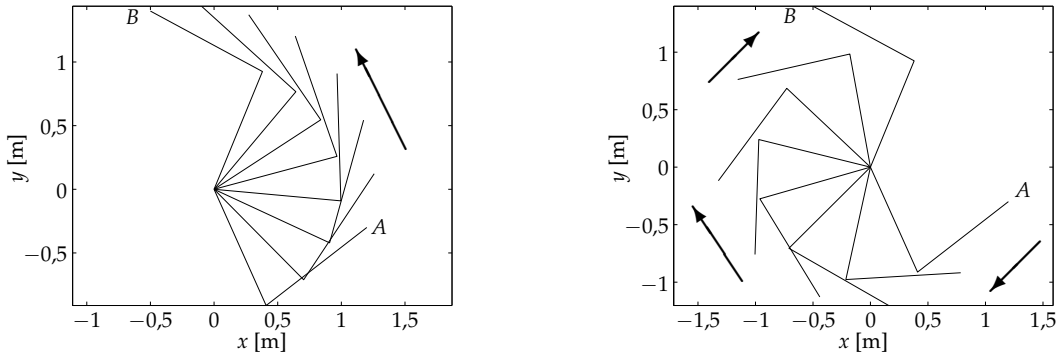


(b) Frente óptima de Pareto, f_P



(c) Frente local, f_l

Figura 7.1. Frentes locais e população inicial



(a) Trajectória pertencente a frente óptima de Pareto

(b) Trajectória pertencente a frente local

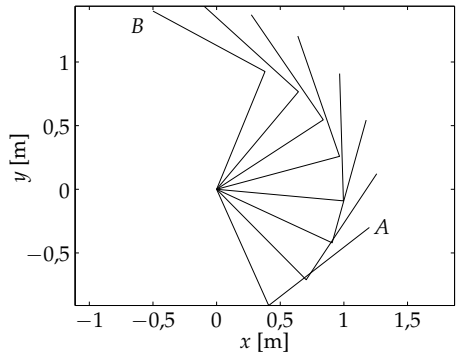
Figura 7.2. Trajectórias das frentes locais

população do AE ficar "encurralada" num óptimo local de geometria extensa, então é difícil aos operadores de variação produzir descendentes que superem os progenitores. No segundo caso, se o óptimo global está localizado numa geometria relativamente pequena e se a população não o encontrou até ao momento, a probabilidade dos operadores de variação de produzir descendentes nessas regiões estreitas é bastante pequena. É o que acontece nos 19,9% dos casos onde o algoritmo converge para a frente local ficando encurralado nessa frente. De facto, as soluções da frente de Pareto são tão diferentes da frente local que os operadores de variação têm uma probabilidade extremamente baixa de transformar uma solução da frente local na frente de Pareto.

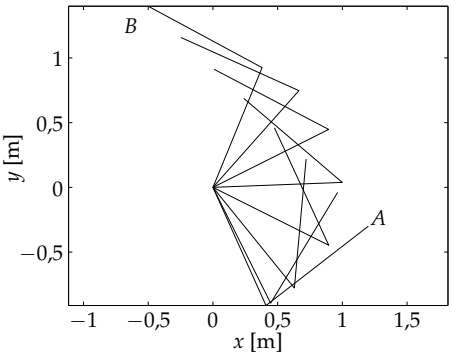
Na figura 7.3 encontram-se representados: a configuração do manipulador robótico e o deslocamento das juntas para as soluções extremas da frente óptima de Pareto, representadas por '*a*' e '*b*' na figura 7.1(b).

7.4 Experiências para o manipulador 2*R*

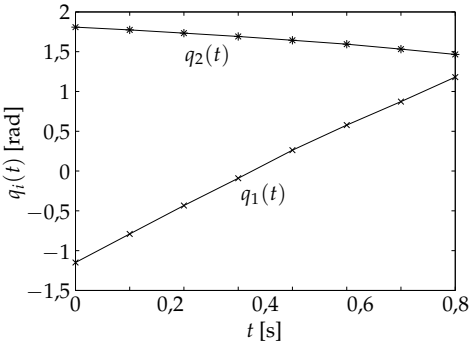
Nesta secção apresentam-se os restantes resultados das experiências, para o manipulador 2*R*, quando são optimizados os cinco objectivos dois a dois (2*D*) e para os cinco objectivos simultaneamente (5*D*).



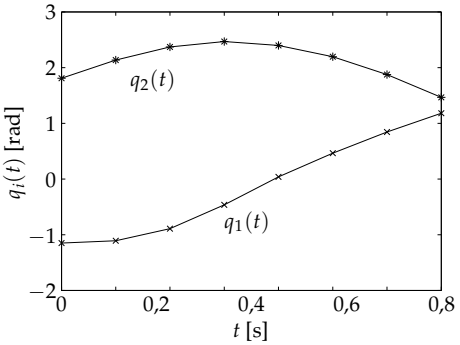
(a) Configurações sucessivas da solução *a* (figura 7.1(b))



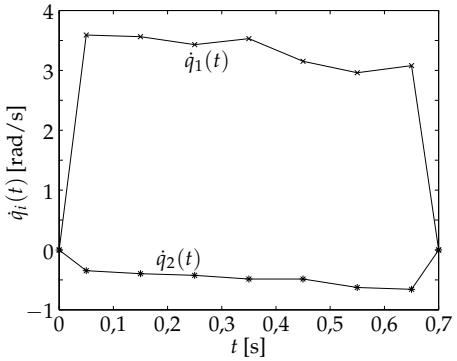
(b) Configurações sucessivas da solução *b* (figura 7.1(b))



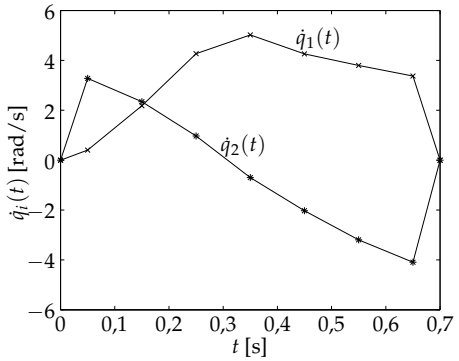
(c) Deslocamento das juntas, solução *a* (figura 7.1(b))



(d) Deslocamento das juntas, solução *b* (figura 7.1(b))



(e) Velocidade das juntas, solução *a* (figura 7.1(b))



(f) Velocidade das juntas, solução *b* (figura 7.1(b))

Figura 7.3. Trajectórias pertencentes à frente óptima de Pareto, para o manipulador 2*R* na optimização O(q, p)

A figura 7.4 apresenta as frentes óptimas de Pareto das optimizações quando são optimizados dois objectivos de cada vez, *i.e.*, O$(q, \dot{q})$, O(q, E_a), O(q, p), O$(q, \dot{p})$, O$(\dot{q}, E_a)$, O$(\dot{q}, p)$, O$(\dot{q}, \dot{p})$, O(E_a, p), O$(E_a, \dot{p})$ e O$(p, \dot{p})$. Como pode ser observado o algoritmo encontra as frentes de Pareto com uma boa distribuição de soluções nas diversas experiências.

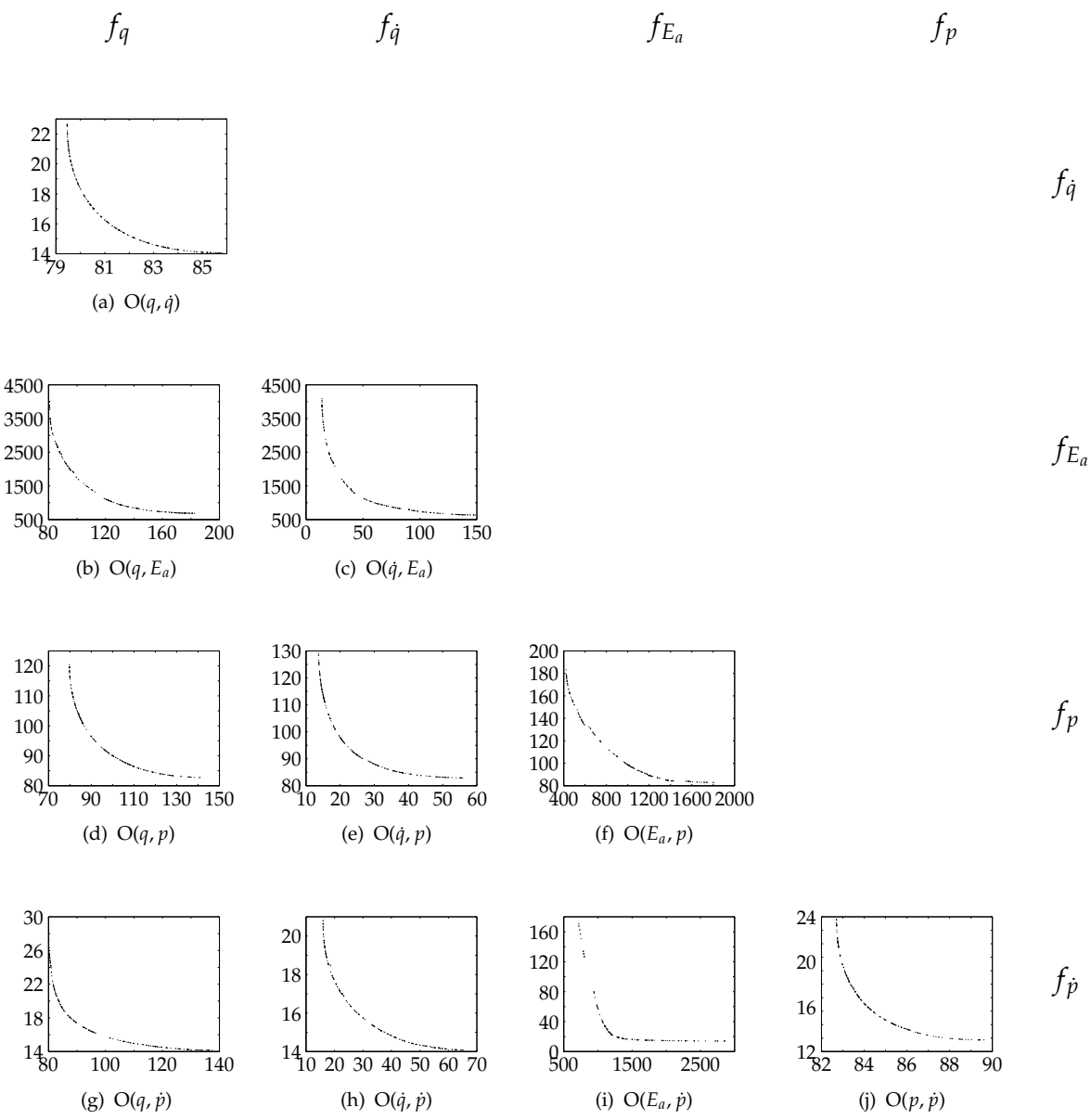
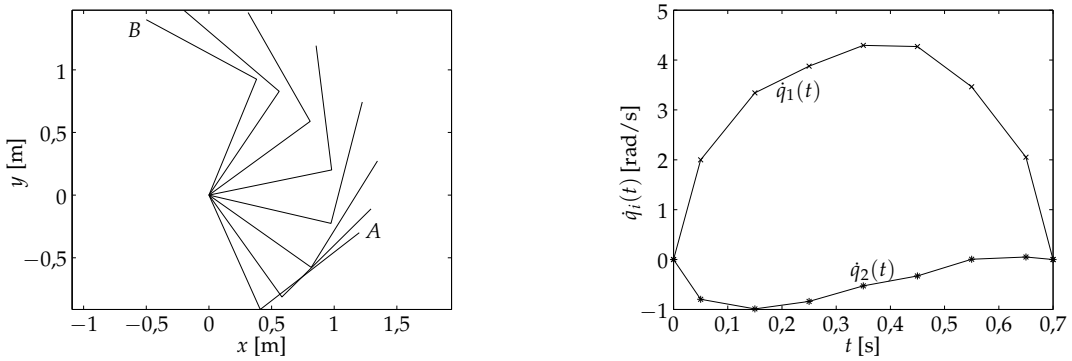


Figura 7.4. Frentes óptimas de Pareto 2*D*

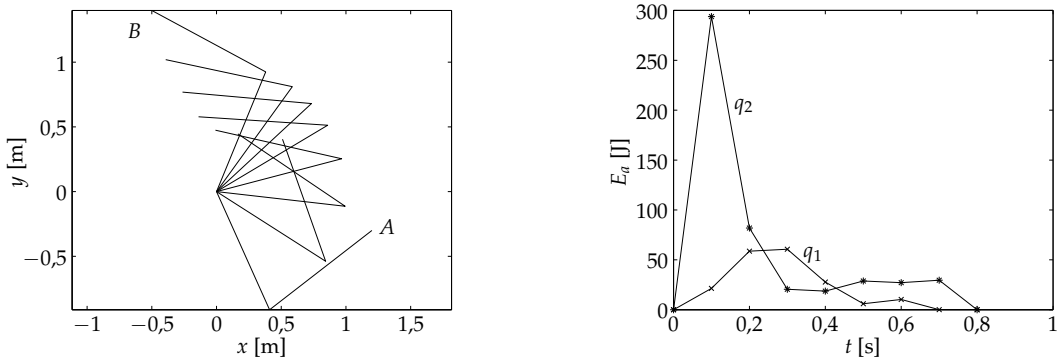
Na figura 7.5 encontram-se os resultados quando se optimiza o deslocamento angular e a oscilação residual da velocidade angular O($q, \dot{q}$). A solução apresentada, $s_{\dot{q}\ \min}$, corresponde à solução não-dominada cujo valor no objectivo $\dot{q}$ é mínimo. Como era esperado a solução não apresenta qualquer oscilação residual (ver figura 7.5(b)). Por outro lado, na figura 7.6 ilustra o resultado para a solução $f_{E_a\ \min}$ na optimização O(q, E_a). Como se pode observar o manipulador descreve uma trajectória de modo a despender a menor energia possível. Finalmente, na figura 7.7 são

apresentados os resultados da experiência quando se minimiza o deslocamento angular e a velocidade linear do órgão terminal do manipulador. Quando se escolhe uma solução intermédia de uma das frentes de Pareto obtém-se uma solução cujo comportamento é uma combinação das soluções apresentadas nas figuras 7.3, 7.5, 7.6 e 7.7.



(a) Configurações sucessivas do manipulador robótico (b) Velocidade angular

Figura 7.5. Solução $s_{\dot{q}\ \min}$ na optimização O($q, \dot{q}$)

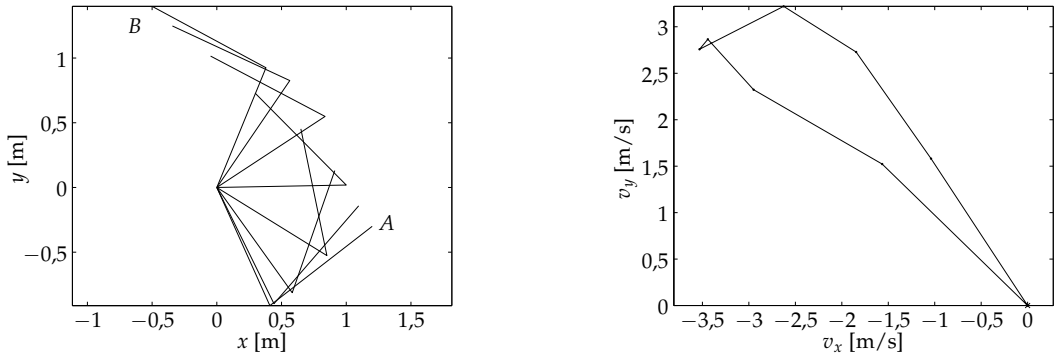


(a) Configurações sucessivas do manipulador robótico (b) Energia E_a

Figura 7.6. Solução $s_{E_a\ \min}$ na optimização O(q, E_a)

De seguida, o manipulador planar 2*R* é optimizado considerando os cinco objectivos simultaneamente (5*D*), descritos pelas equações (7.2). As figuras 7.8 e 7.9 mostram os resultados obtidos com o número de gerações $T_{\mathrm{t}} = 50000$ e o número de vectores de $pop_{\mathrm{dim}} = 1000$.

A figura 7.8 apresenta os valores de compromisso (*tradeoffs*) normalizados para toda



(a) Configurações sucessivas do manipulador robótico

(b) Velocidade cartesiana v_x *vs.* v_y

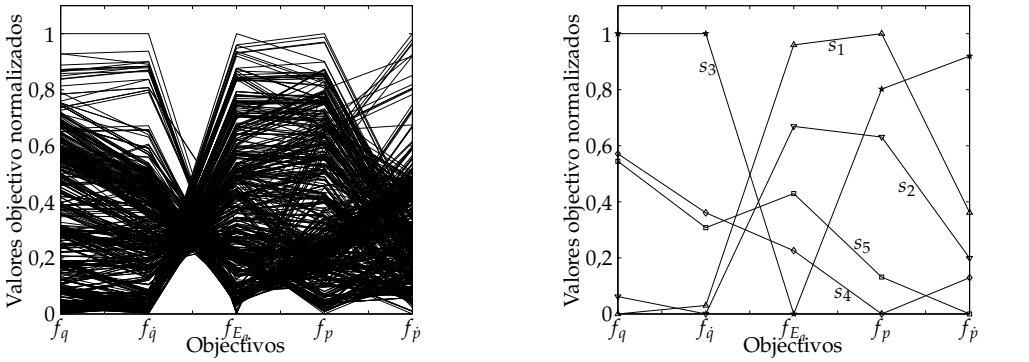
Figura 7.7. Solução $s_{\dot{p}\ \min}$ na optimização O(q, $\dot{p}$)

a população e para as melhores soluções $s_i = \{s_1, s_2, s_3, s_4, s_5\}$ relativas aos objectivos $O_i = \{f_q, f_{\dot{q}}, f_{E_a}, f_p, f_{\dot{p}}\}$. A tabela 7.1 contém o intervalo dos valores obtidos numa simulação.

Pode ver-se que o algoritmo 5D não consegue obter resultados tão bons como os obtidos com o algoritmo 2D devido ao aumento significativo da complexidade do espaço de pesquisa. Contudo, as soluções encontradas apresentam uma boa distribuição (figura 7.8(a)) tendo os resultados valores próximos dos obtidos nas frentes 2D respectivas. Por outro lado, quando se considera apenas a optimização 2D pode acontecer que a frente óptima de Pareto 2D contenha soluções dominadas do ponto de vista da optimização 5D. Estas soluções não são tomadas em conta quando se considera a optimização 5D.

Da figura 7.8(b) pode ser concluído que f_q e $f_{\dot{q}}$ ou f_p e $f_{\dot{p}}$ são objectivos conflituosos com um compromisso pequeno entre eles. Por outro lado, o objectivo f_{E_a} apresenta o maior compromisso entre os objectivos restantes. Na figura 7.9 estão os resultados para as soluções s_i obtidas. Para a trajectória em estudo, os resultados indicam que quanto mais perto o manipulador se desloca perto da sua base menor é a energia consumida (figuras 7.9(e), 7.9(g) e 7.9(h))

Na figura 7.10 encontram-se as projecções da frente 5D nos diferentes planos 2D. Em



(a) Soluções obtidas

(b) Melhores soluções em pelo menos um objectivo

Figura 7.8. Relação entre os objectivos f_q, $f_{\dot{q}}$, f_{E_a}, f_p e $f_{\dot{p}}$ das soluções da frente não-dominada, para o robô 2*R*

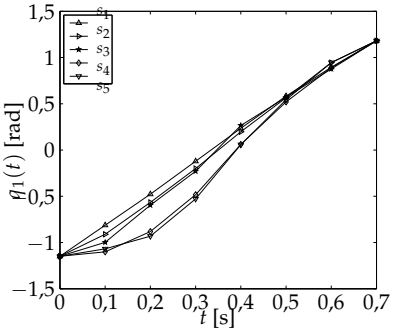
Tabela 7.1. Intervalo dos objectivos na optimização 5*D*

	f_q (rad^2/s^2)	$f_{\dot{q}}$ (rad^4/s^4)	f_{E_a} (J)	f_p (m^2/s^2)	$f_{\dot{p}}$ (m^4/s^4)
min	79,8	18,2	1056,7	83,5	15,0
max	182,3	101,7	4602,7	121,8	56,4

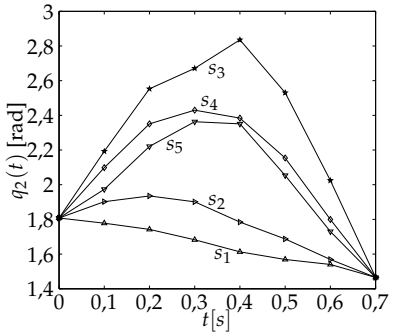
cada plano, também se encontra a frente não-dominada correspondente à optimização 2*D* relativa aos objectivos do plano. Pode observar-se que a optimização 5*D* não consegue obter uma frente tão boa quanto a optimização 2*D*. Isto deve-se ao aumento da dimensão do espaço de pesquisa e consequentemente à redução da relação entre a população e a dimensão do espaço de pesquisa diminuir drasticamente. A distância entre as frentes 2*D* e 3*D*, 4*D* e 5*D* obtida aumenta à medida que se foi aumentado o número de objectivos.

7.5 Optimização de trajectórias para um manipulador 3*R*

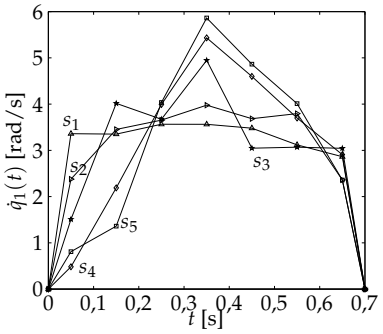
Nesta secção apresenta-se a optimização com cinco objectivos e os resultados da simulação para um manipulador 3*R* num ambiente de trabalho com um obstáculo circular de centro $c = (1; 0{,}4)$ e raio $\rho = 0{,}4$. Os elos do manipulador têm um comprimento de $l = 0{,}66$ m e uma massa de $m = 0{,}66$ kg. As configurações extremas



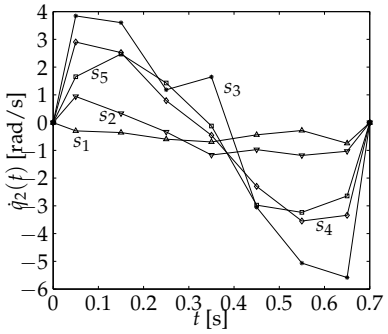
(a) Posição da junta 1 *versus* tempo



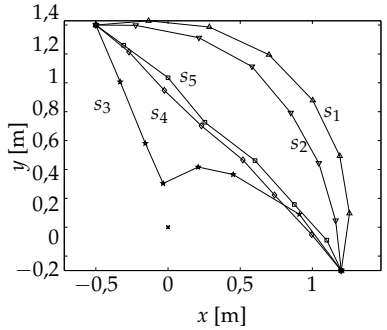
(b) Posição da junta 2 *versus* tempo



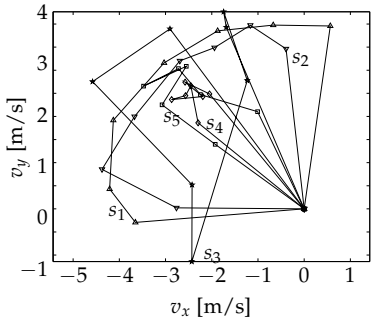
(c) Velocidade da junta 1 *versus* tempo



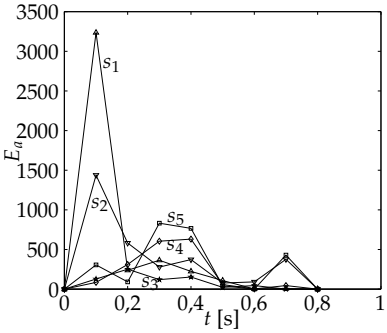
(d) Velocidade da junta 2 *versus* tempo



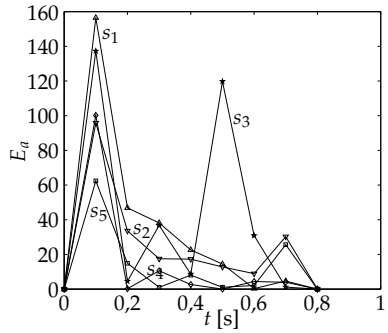
(e) Movimento cartesiano



(f) Velocidade cartesiana

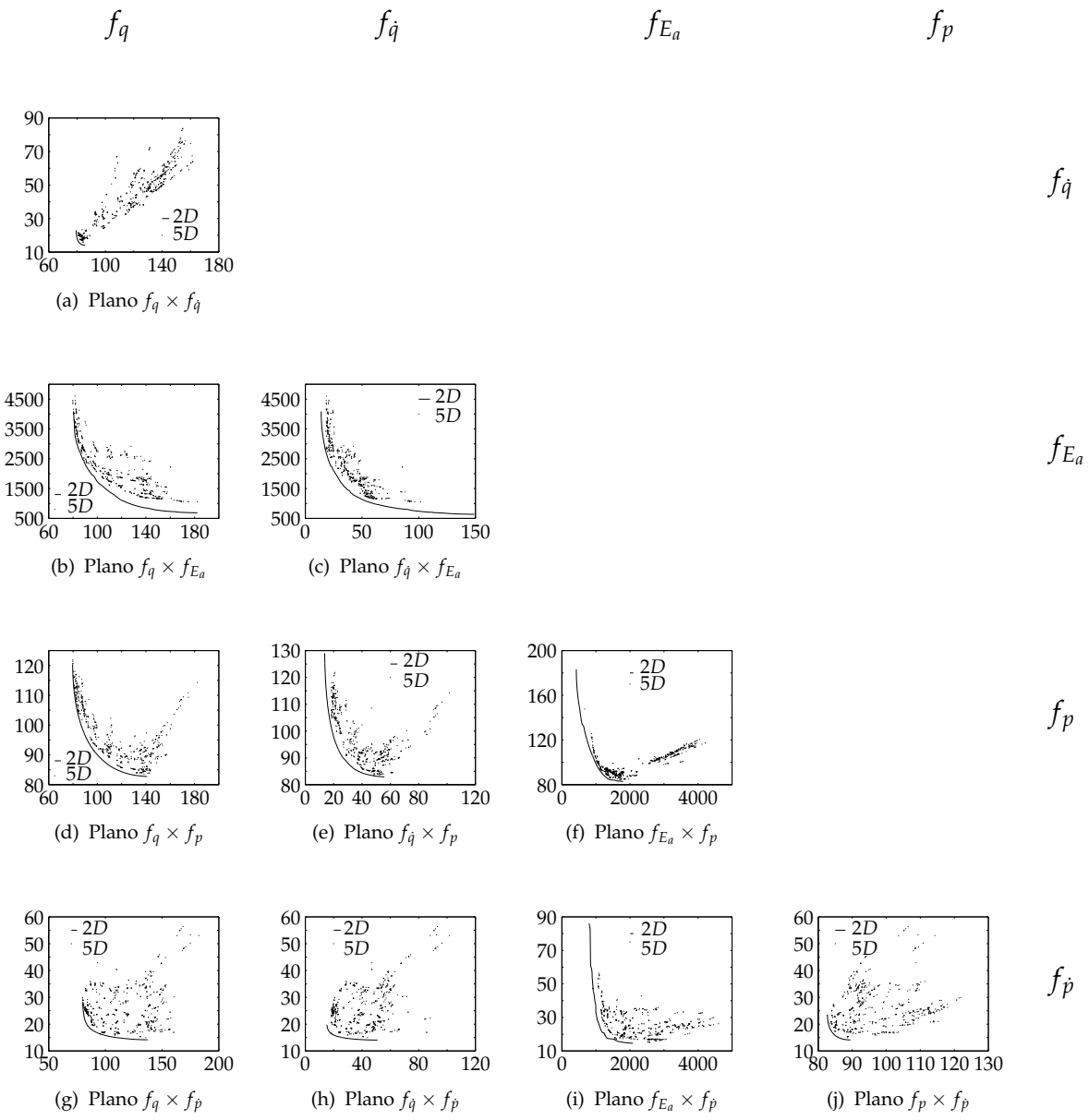


(g) Energia requerida para a junta 1



(h) Energia requerida para a junta 2

Figura 7.9. Comportamento das melhores soluções, para o manipulador 2R relativamente a cada um dos cinco objectivos considerados



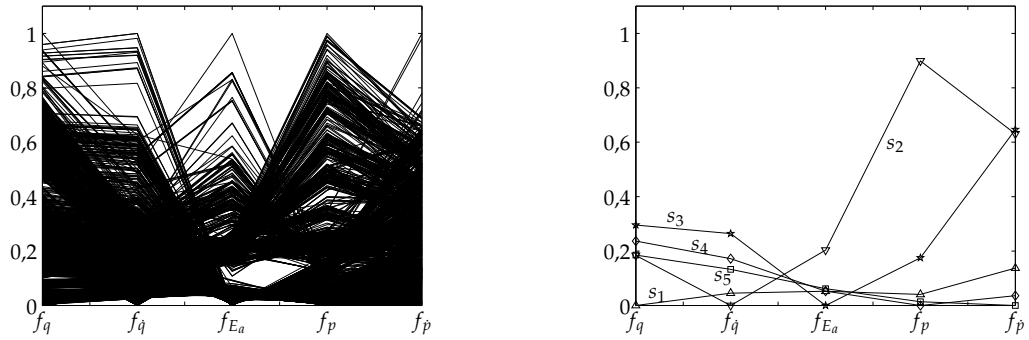
(a) Plano $f_q \times f_{\dot{q}}$ (b) Plano $f_q \times f_{E_a}$ (c) Plano $f_{\dot{q}} \times f_{E_a}$ (d) Plano $f_q \times f_p$ (e) Plano $f_{\dot{q}} \times f_p$ (f) Plano $f_{E_a} \times f_p$ (g) Plano $f_q \times f_{\dot{p}}$ (h) Plano $f_{\dot{q}} \times f_{\dot{p}}$ (i) Plano $f_{E_a} \times f_{\dot{p}}$ (j) Plano $f_p \times f_{\dot{p}}$

Figura 7.10. Projecções da Frente 5*D* nos diferentes planos, para o robô 2*R*

foram escolhidas de modo a obter posições idênticas ao manipulador 2*R*.

Na figura 7.11 encontram-se os valores de aptidão normalizados da população e das soluções que têm o melhor desempenho relativamente a um dos objectivo. Também nesta simulação se obteve uma boa distribuição das soluções. Na tabela 7.2 estão indicados os intervalos nos quais variam as soluções não-dominadas.

Na figura 7.12 são apresentadas as projecções da frente 5*D* nos diversos planos 2*D*.



(a) Soluções obtidas

(b) Melhores soluções em pelo menos um objectivo

Figura 7.11. Relação entre os objectivos f_q, $f_{\dot{q}}$, f_{E_a}, f_p e $f_{\dot{p}}$ das soluções da frente não-dominada para o manipulador 3*R*

Tabela 7.2. Intervalo dos objectivos na optimização 5*D* para o robô 3*R*

	f_q (rad^2/s^2)	$f_{\dot{q}}$ (rad^4/s^4)	f_{E_a} (J)	f_p (m^2/s^2)	$f_{\dot{p}}$ (m^4/s^4)
min	155,1	52,8	446,9	74,4	12,9
max	638,0	731,9	20531,3	333,7	108,3

É notório que as soluções têm uma boa distribuição em todo o espaço de pesquisa. Pode verificar-se que a frente óptima de Pareto não é continua o que dá a impressão de existirem duas frentes. A descontinuidade das frentes surge devido às soluções óptimas relativamente a todos os objectivos não serem obtidas quando o manipulador se desloca no mesmo sentido.

Na figura 7.13 encontram-se as configurações resultantes das melhores soluções relativas a cada um dos objectivos.

7.6 Outras simulações

Nesta secção são apresentadas duas simulações. A primeira diz respeito a uma experiência com um manipulador de dois elos e com três objectivos. Esta simulação, tem como finalidade salientar a forma da frente óptima de Pareto para experiências a três dimensões. A segunda experiência adopta um robô com três elos e um obstáculo e tem como objectivo estudar o efeito do obstáculo na frente não-dominada.

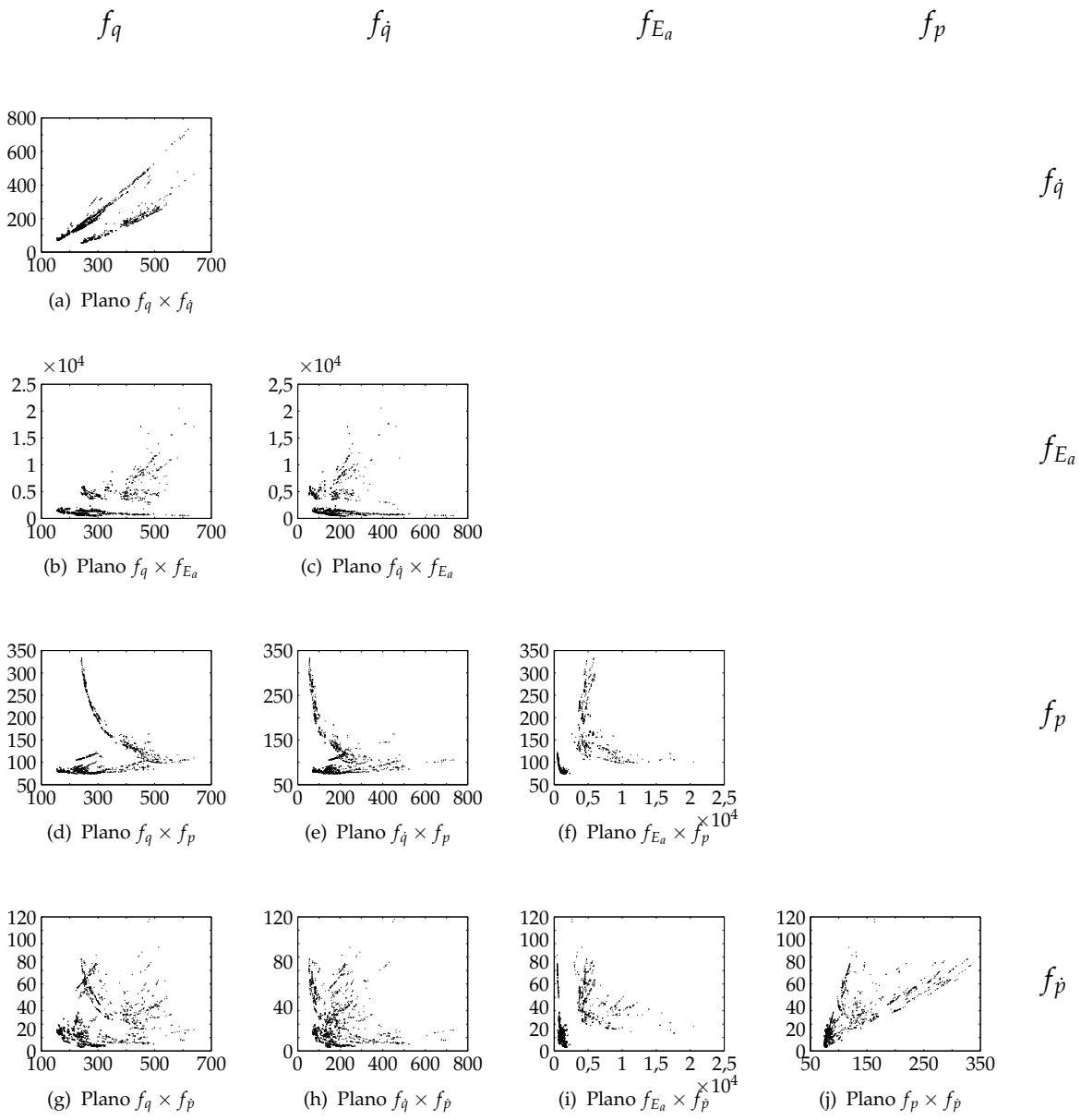
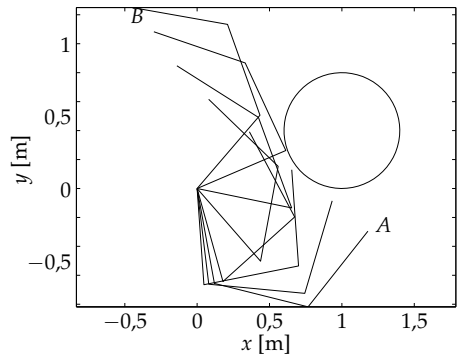


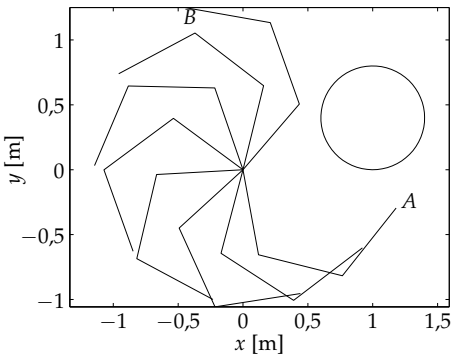
Figura 7.12. Frentes locais 5D nos diversos planos, para o manipulador 3R

Na resolução destas experiências, o número de elementos da população usado é 600 e o número de gerações é 30000. Os parâmetros restantes utilizados são os mesmos que os da secção anterior.

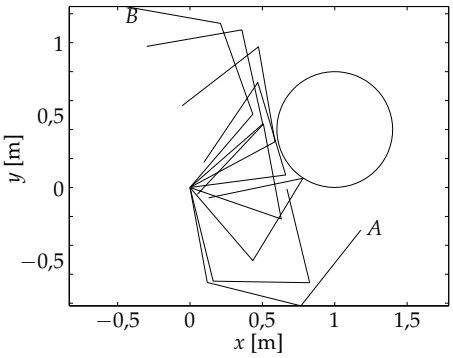
Na figura 7.14(a), são apresentados os resultados, para o manipulador 2R, quando se optimizam três objectivos nomeadamente o deslocamento angular das juntas, o deslocamento linear do órgão terminal e a energia requerida pelo manipulador para



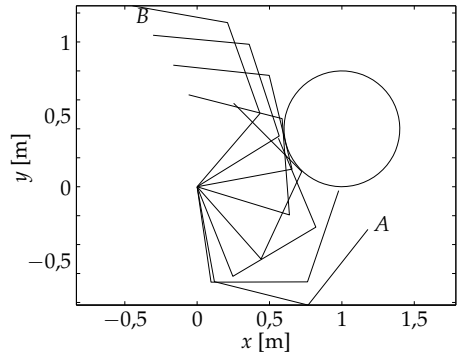
(a) Configurações sucessivas para a melhor trajectória do objectivo f_q



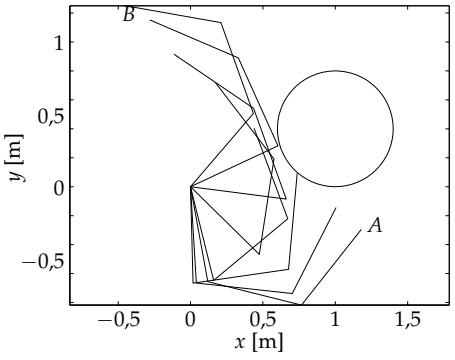
(b) Configurações sucessivas para a melhor trajectória do objectivo $f_{\dot{q}}$



(c) Configurações sucessivas para a melhor trajectória do objectivo f_{E_a}



(d) Configurações sucessivas para a melhor trajectória do objectivo f_p

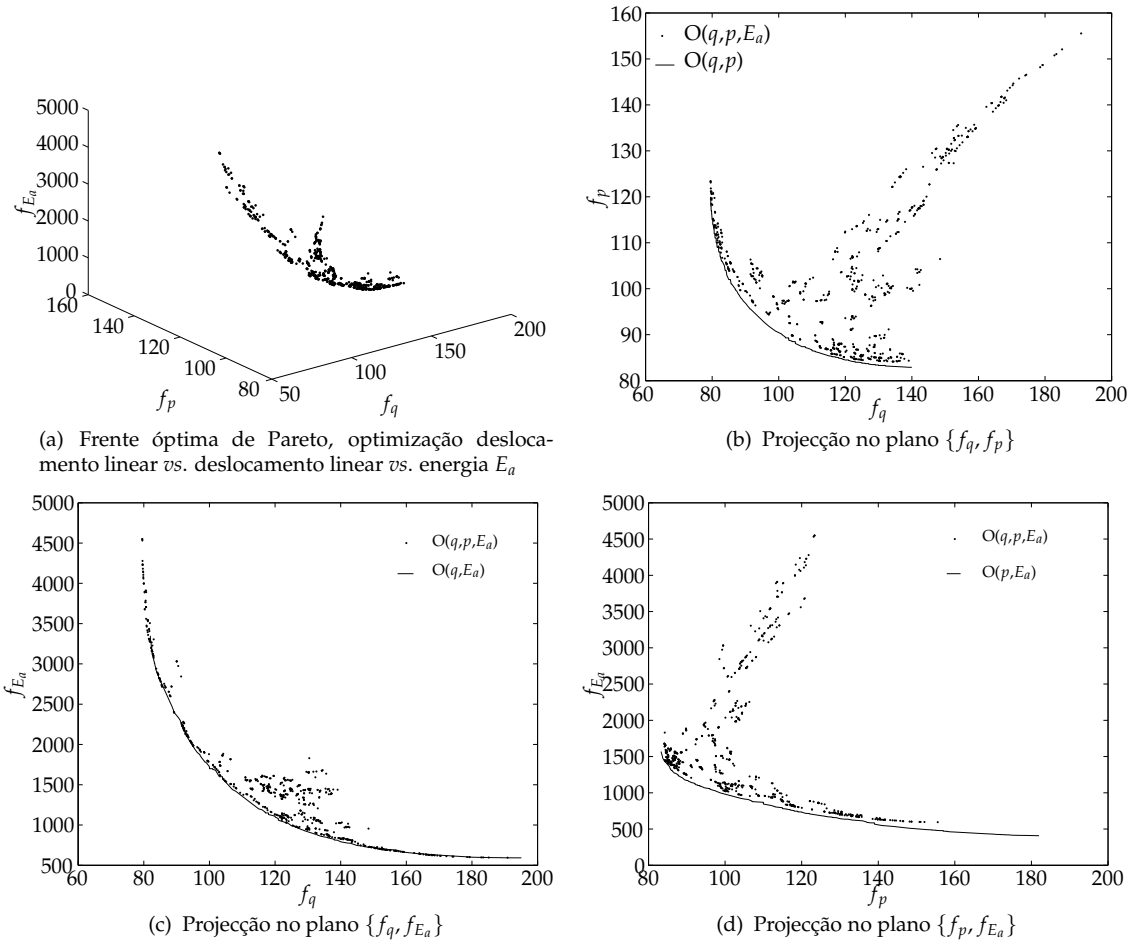


(e) Configurações sucessivas para a melhor trajectória do objectivo $f_{\dot{p}}$

Figura 7.13. Melhores trajectórias obtidas em cada um dos objectivos, para o manipulador 3*R*

executar a tarefa. A frente obtida é contínua tendo a forma de um 'Y' concavo.

Nas figuras 7.14(b)-7.14(d) encontra-se a frente óptima de Pareto projectada nos diferentes planos: $\{q, p\}$, $\{q, E_a\}$ e $\{p, E_a\}$. Nestas figuras também se encontram sobrepostas as frentes das optimizações obtidas com a optimização referente a dois objectivos do plano em questão. Comparando os resultados, pode-se concluir que o algoritmo com três objectivos converge para a frente óptima de Pareto semelhante à frente obtida com os dois objectivos correspondentes.



(a) Frente óptima de Pareto, optimização deslocamento linear *vs.* deslocamento linear *vs.* energia E_a

(b) Projecção no plano $\{f_q, f_p\}$

(c) Projecção no plano $\{f_q, f_{E_a}\}$

(d) Projecção no plano $\{f_p, f_{E_a}\}$

Figura 7.14. Frente óptima de Pareto e suas projecções, para o manipulador 2R com 3 objectivos

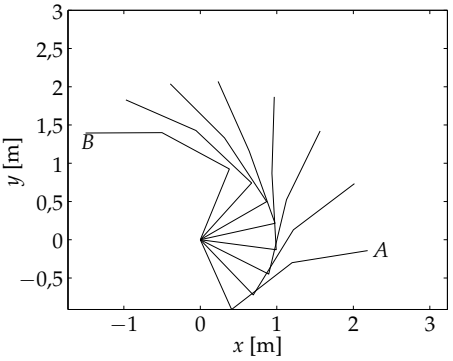
Na figura 7.15, encontram-se os resultados para um manipulador 3R, com $l = 1$ m e $m = 1$ kg para a optimização do deslocamento angular e do deslocamento linear do órgão terminal entre duas configurações $A \equiv \{-1{,}15; 1{,}81; -0{,}50\}$ e $B \equiv$

$\{1{,}18; 1{,}47; 0{,}50\}$. Os resultados dizem respeito ao ambiente com e sem obstáculos. O obstáculo considerado é circular de centro $c = (2;2)$ e raio $\rho = 1$. Na figura 7.15(c) então representadas as frentes óptimas de Pareto f_1 e f_2 para o ambiente com e sem obstáculos, respectivamente. Assim, ao introduzir o obstáculo a frente óptima de Pareto é reduzida. Consequentemente, no ambiente com obstáculos não se consegue obter valores com um custo tão baixo como no ambiente sem obstáculos para a função do deslocamento angular, o que se pode constatar nas figuras 7.15(a), 7.15(b) e 7.15(c). No outro extremo da frente, as soluções obtidas '*b*' e '*d*' são praticamente idênticas concluindo-se, assim, que a optimização da distância linear do órgão terminal não é afectada pelo obstáculo.

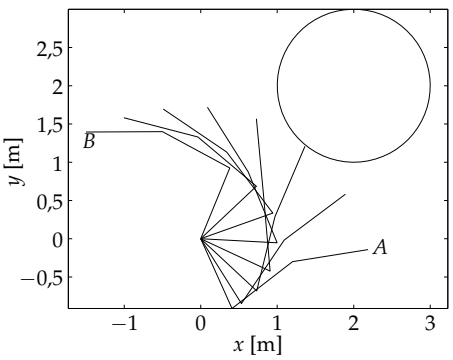
7.7 Conclusões

Neste capítulo é resolvido o planeamento de trajectórias robóticas numa perspectiva multi-objectivo com vista a encontrar um conjunto de soluções pertencentes a frente óptima de Pareto. Os resultados apresentados demonstram claramente que o algoritmo encontra essa frente ou uma muito próxima. Adicionalmente, os resultados permitem concluir que o algoritmo consegue obter uma boa distribuição de soluções ao longo da frente. A presença de obstáculos no ambiente pode afectar a frente óptima de Pareto mas não constitui uma dificuldade acrescida na resolução do problema. À medida que se aumenta o número de objectivos, a dimensão do algoritmo aumenta exponencialmente obtendo-se um resultado de menor qualidade. Este inconveniente pode ser contrariado à custa do aumento da população e do número de gerações. No entanto, a optimização $2D$ pode incluir soluções dominadas no ponto de vista da optimização $5D$.

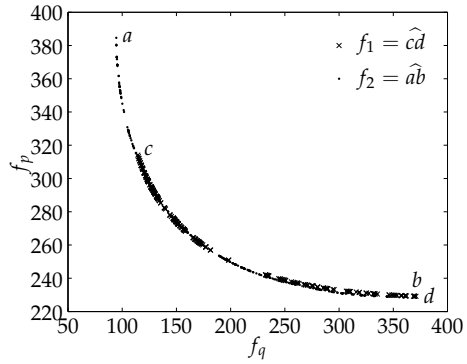
O método proposto permite ajudar o agente de decisão na escolha da melhor solução fornecendo-lhe um conjunto representativo de soluções pertencentes à frente não-dominada.



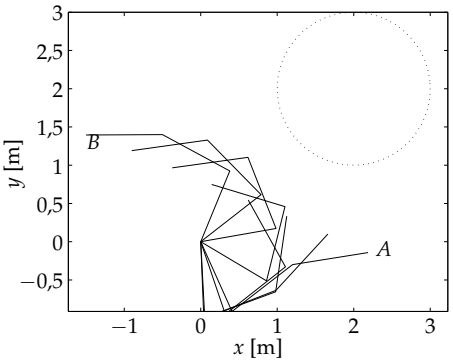
(a) Configurações sucessivas do manipulador, solução *a*



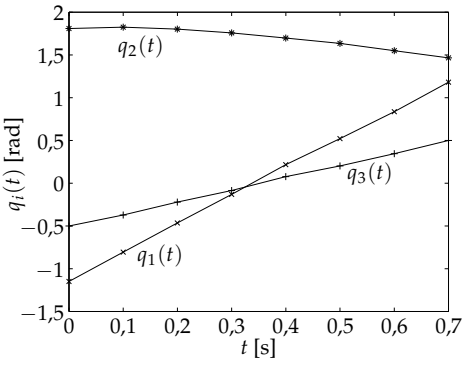
(b) Configurações sucessivas do manipulador, solução *c*



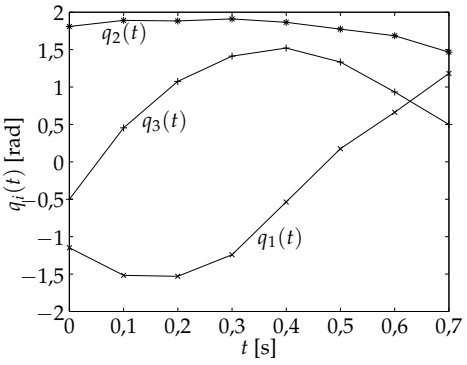
(c) Frente óptima de Pareto, f_1 ambiente com obstáculo, f_2 ambiente sem obstáculos



(d) Configurações sucessivas do manipulador, solução *b*



(e) Deslocamento angular do manipulador, solução *a*



(f) Deslocamento angular do manipulador, solução *b*

Figura 7.15. Frente óptima de Pareto, configurações sucessivas e deslocamento angular para o robô 3*R*

8 Algoritmo de selecção multi-objectivo MaxiMin

8.1 Introdução

Recentemente, a pesquisa na área dos AEs demonstrou que existem grandes potencialidades para a resolução de problemas multi-objectivo (PMO). Assim, nos últimos anos tem sido proposta uma grande quantidade de métodos multi-objectivo baseados em AEs (ver capítulo 3). Têm sido também desenvolvidas várias métricas para medir o desempenho desses algoritmos. A eficiência de um AE multi-objectivo (AEMO) pode ser medida pelas seguintes características:

- A distância entre a frente não-dominada, encontrada pelo algoritmo, e a frente óptima de Pareto deve ser minimizada (válido quando se conhece *a priori* a frente óptima de Pareto).

- As soluções não-dominadas devem estar uniformemente distribuídas em toda a frente.

- A extensão da frente não-dominada deve ser maximizada, sendo cada objectivo coberto por um vasto número de soluções não-dominadas.

Guardar soluções da frente óptima de Pareto de modo a garantir uma boa diversidade de soluções e uma extensão grande da mesma é um problema que consome bastantes recursos computacionais (e tempo) dentro dos AEMO. O peso computacional está directamente relacionado com os níveis de diversidade e distribuição das soluções que se pretende obter no AEMO. Quanto mais elevado forem estes níveis, mais tempo de processamento computacional será necessário. Por exemplo, o algoritmo NSGA-II usa o método pombalino que tem um peso computacional de $O(pop_{\dim} \log pop_{\dim})$. Por outro lado, o algoritmo SPEA usa o método de agrupamento que tem uma complexidade computacional de $O((pop_{\dim})^3)$. É sabido que para problemas com dois objectivos a diferença em termos da diversidade das soluções obtidas não é significativa, para três ou mais objectivos o algoritmo SPEA tem provado que é claramente melhor [136], embora a custo de uma carga computacional mais elevada. Assim, novos métodos que encontrem a frente não-dominada com um peso computacional razoável são bastante desejáveis e consequentemente são alvo de investigação [137].

A organização do capítulo é a seguinte. Na secção 8.2 é apresentada uma técnica nova cujo objectivo primordial é a dispersão das soluções da frente não-dominada, utilizando métricas no espaço dos objectivos. De seguida, na secção 8.3 é apresentada uma técnica que permite estudar a convergência de um algoritmo. Na secção 8.4 são descritos dois métodos para medir a diversidade das soluções da frente óptima de Pareto e a extensão da frente não-dominada. Na secção 8.5, para validar o algoritmo MaxiMin, insere-se este no algoritmo NSGA-II (substituindo o algoritmo pombalino) testando-o em funções bem conhecidas. De seguida, na secção 8.6, para medir a convergência dos algoritmos, estudar a distribuição das soluções ao longo da frente óptima de Pareto e medir a extensão da sua frente é optimizado um conjunto de trajectórias para manipuladores $2R$, $3R$ e $4R$, sendo realizada a análise dos resultados. Por último, na secção 8.7 são enunciadas as principais conclusões.

8.2 Operador de selecção MaxiMin

Nesta secção é apresentado um novo método que promove a dispersão das soluções no espaço dos objectivos. O método elitista proposto é aplicado na fase da selecção, baseado em frentes não-dominadas, e consiste em seleccionar as N soluções, entre as população progenitora P (ou arquivo) e a descendente D, de modo a obter as soluções mais bem distribuídas. O algoritmo proposto pode ser implementado num grande número de AEMO mas, no âmbito deste trabalho apenas será inserido essencialmente no algoritmo NSGA-II, por este ser um dos métodos mais utilizado entre os AEMO.

O algoritmo aqui apresentado tem como objectivo escolher as melhores N soluções com distribuição e com extensão máxima de uma população inicial com dimensão M ($M > N$). De modo análogo ao algoritmo NSGA-II, o método proposto é baseado em frentes não-dominadas [45]. Contudo, quando a última frente a ser considerada não puder ser totalmente incluída na nova população, o algoritmo MaxiMin é chamado para seleccionar as últimas soluções, em substituição do algoritmo pombalino do NSGA-II. A ideia principal do algoritmo MaxiMin consiste em ir diminuindo as maiores áreas da frente que não sejam representadas por nenhuma solução à medida que se vai constituindo a população seguinte.

Considere-se o exemplo ilustrado na figura 8.1. Se se pretender seleccionar as j soluções mais dispersas obtêm-se as soluções $S = \{a, b\}$, $S = \{a, b, c\}$, $S = \{a, b, c, d, e\}$, $S = \{a, b, c, d, e, f, g, h, i\}$, respectivamente para $j = \{2, 3, 5, 9\}$. Este resultado pode ser obtido usando dois conjuntos: R e S. O conjunto R contém todas as soluções a considerar enquanto que o conjunto S é usado para guardar as soluções com melhor distribuição que vão sendo retiradas do conjunto R. Inicialmente, retiram-se de R as melhores soluções relativamente a cada objectivo. Neste caso, retiram-se duas soluções: 'a' e 'b' e colocam-se no conjunto S, garantindo assim a máxima extensão da frente. De seguida retira-se de R a solução mais distante ao conjunto S (*i.e.* 'c') e

insere-se neste conjunto. As próximas soluções a serem seleccionadas são '*d*' e '*e*'. O processo termina quando for atingido o número de soluções pretendido.

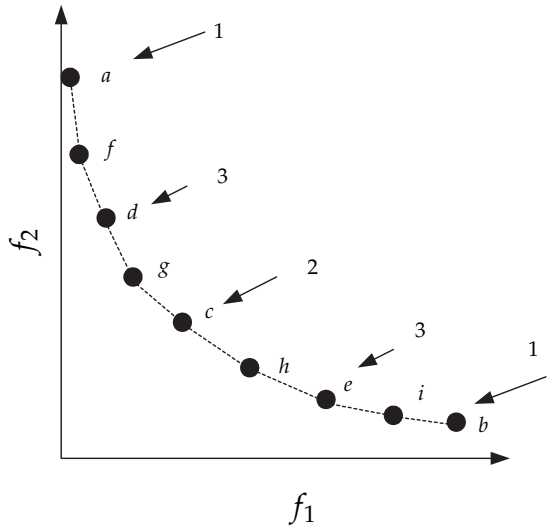


Figura 8.1. Soluções da frente não-dominada

Apresenta-se o algoritmo MaxMin encontra-se em 8.1 e é descrito pormenorizadamente de seguida, utilizando a notação da tabela 8.2.

Inicialmente, agrupam-se as populações progenitora P (ou arquivo) e descendente D (linha 1). De seguida, a partir da população R são retiradas as soluções que pertencem à frente não-dominada (linha 4). No caso do número de soluções ser maior que o número pretendido retiram-se as soluções óptimas relativamente a cada um dos objectivos (linhas 5:9). Caso contrário, vão-se retirando as frentes não--dominadas e inserem-se na nova população até quando a frente a ser considerada não couber na população final (linhas 10:13). Posteriormente, vão-se retirando da população A as soluções cuja distância à população S é maior, inserindo-as nesta (linhas 17:22). O processo termina quando a população final tiver N ($N = pop_{\dim}$) soluções.

O cálculo da distância entre uma solução de A e o conjunto S é fornecido através da distância mínima entre a solução de A e todas as soluções de S (8.1a). A solução seleccionada é aquela que apresenta o maior valor (8.1b). Este algoritmo requer no

```
1  início
2      R = S ∪ D;
3      S = ∅;
4      A = getND(R);
5      se #A > pop_dim então
6          repetir de i = 1 até n_obj
7              S = S∪ getMin(R, i);
8          fim repetir
9      fim
10     enquanto #S + #A ≤ pop_dim faça
11         S = S ∪ A;
12         A = getND(R);
13     fim
14     repetir de j = 1 até #A
15         c_{a_j} = min_{s_i ∈ S} {‖f_{a_j} − f_{s_i}‖};
16     fim repetir
17     enquanto #S < pop_dim faça
18         k = getMaxCi(A);
19         S = S ∪ k;
20         repetir de l = 1 até #A
21             c_{a_l} = min{‖f_{a_l} − f_k‖, c_{a_l}};
22         fim repetir
23     fim
24 fim
```

Algoritmo 8.1. Algoritmo do operador de selecção multi-objectivo MaxiMin

Tabela 8.1. Descrição das variáveis e funções do algoritmo multi-objectivo MaxMin

	Descrição
P	População progenitora, ou arquivo
D	População descendente
A, R	Populações auxiliares
S	População final
k	Solução com distância máxima
i, j, l	Contadores
c_{a_i}	Norma euclidiana mínima entre a solução i e a população S
getMin(X,i)	Retira da população X a solução cujo valor objectivo i é mínimo
getMaxCi(X)	Retira a solução cujo valor da norma euclidiana c_i é máxima
getND(X)	Retira todas as soluções não-dominadas, sem repetição, da população X

máximo $O(N^2)$ computações.

$$c_{a_j} = \min_{s_i \in S} \parallel f_{a_j} - f_{s_i} \parallel \tag{8.1a}$$

$$S = S \cup \max_{a_j \in A} a_j \tag{8.1b}$$

De seguida exemplifica-se o algoritmo MaxiMin abordando a minimização de um problema com dois objectivos. Considerando as populações P e D, com $M = 12$ e $pop_{\text{dim}} = 6$, ilustradas na figura 8.2 e representadas em seguida por:

$$P = \{p_1 = (1;10); p_2 = (3;7); p_3 = (4,2); p_4 = (8;9); p_5 = (9;1); p_6 = (7;3)\}$$

$$D = \{d_1 = (2;7); d_2 = (8,3); d_3 = (6;5); d_4 = (9;4); d_5 = (7;6); d_6 = (5,10)\}$$

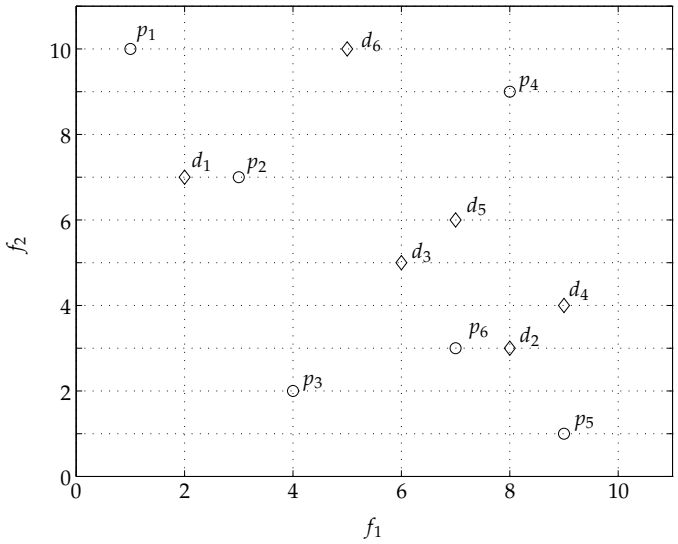


Figura 8.2. Populações P e D com dimensão 6

onde $s_i = (o_1, o_2)$ representa a solução i com valores o_1 e o_2, respectivamente para os objectivos f_1 e f_2. O algoritmo começa por fazer a união das duas populações (linha 2):

$$R = \{p_1, p_2, p_3, p_4, p_5, p_6, d_1, d_2, d_3, d_4, d_5, d_6\}$$

De seguida retiram-se os elementos não-dominados da população R (linha 4):

$$\begin{aligned} R &= \{p_2, p_4, p_6, d_2, d_3, d_4, d_5, d_6\} \\ A &= \{p_1, p_3, p_5, d_1\} \end{aligned}$$

Como a soma das soluções dos conjuntos A e S é inferior à dimensão da população (linha 10), os elementos de A são todos inseridos na população S ($\#A + \#S \leq pop_{\text{dim}}$) (linha 11), resultando as expressões:

$$\begin{aligned} R &= \{p_2, p_4, p_6, d_2, d_3, d_4, d_5, d_6\} \\ A &= \emptyset \\ S &= \{p_1, p_3, p_5, d_1\} \end{aligned}$$

Retiram-se novamente os elementos não-dominados da população T (linha 12):

$$\begin{aligned} R &= \{p_4, d_2, d_4, d_5, d_6\} \\ A &= \{p_2, p_6, d_3\} \\ S &= \{p_1, p_3, p_5, d_1\} \end{aligned}$$

Como $\#A + \#S > pop_{\text{dim}}$, calculam-se as normas euclidianas mínimas das soluções da população A à população S (linhas 14:16):

$$
\begin{aligned}
c_{p_2} &= \min(\| p_2 - p_1 \|, \| p_2 - p_3 \|, \| p_2 - p_5 \|, \| p_2 - d_1 \|) = 1 \\
c_{p_6} &= 2{,}8 \\
c_{d_3} &= 3{,}6
\end{aligned}
$$

insere-se a solução com a norma máxima, d_3 (linha 18), na população S (linha 19) e recalculam-se os valores c_{a_i} linhas (20-22)

$$
\begin{aligned}
R &= \{p_4, d_2, d_4, d_5, d_6\} \\
A &= \{p_2, p_6\} \\
S &= \{p_1, p_3, p_5, d_1, d_3\} \\
c_{p_2} &= \min(1; \| p_2 - d_3 \|) = \min(1; 3{,}6) = 1 \\
c_{p_6} &= \min(2{,}8; 2{,}2) = 2{,}2
\end{aligned}
$$

Com a selecção do elemento p_6 termina-se o processo de obtenção da população S que constituirá a nova população progenitora ou arquivo. A população final está ilustrada na figura 8.3 e é representada de seguida por S:

$$S = \{p_1, p_5, d_1, p_3, d_3, p_6\}$$

A execução deste processo, no final de cada geração, leva a que a escolha das soluções favoreça as soluções mais dispersas, no espaço dos objectivos, de acordo com as frentes não-dominadas. No exemplo anterior isto verifica-se com a exclusão p_2 em detrimento das soluções p_6 e d_3.

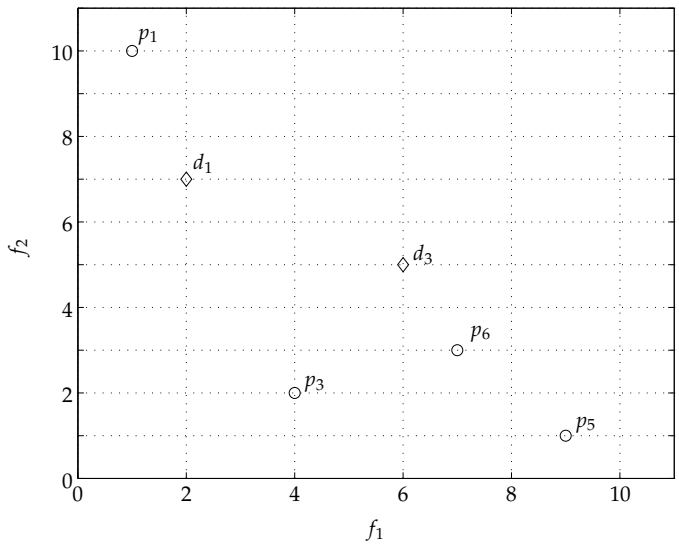


Figura 8.3. População S após a execução do algoritmo MaxiMin para o exemplo ilustrado na figura 8.2

8.3 Estudo da convergência das frentes

Nesta secção é apresentado um método que permite analisar a convergência de um algoritmo para a frente óptima de Pareto. O método pode ser aplicado em problemas em que a frente não-dominada possa ser modelada por uma função analítica. O método baseia-se na estimação dos parâmetros de uma função de forma a modelar a frente de Pareto obtida no conjunto de testes. Se após várias testes, os parâmetros forem semelhantes então o algoritmo convergiu sempre para a mesma frente. A identificação da função a usar é fornecida pela experiência do utilizador ou através de um programa genético. Para validar o modelo usa-se o método dos mínimos quadrados.

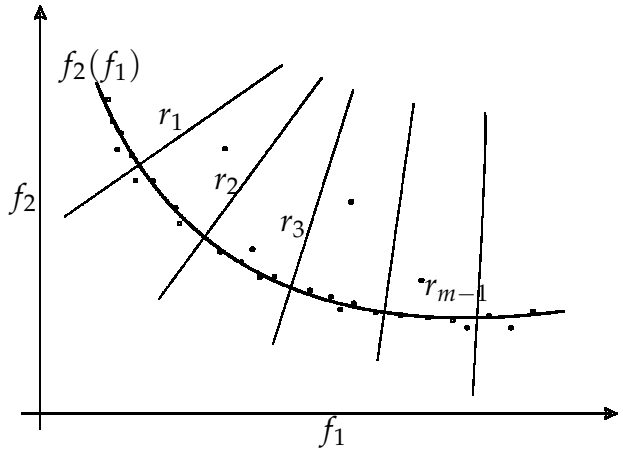


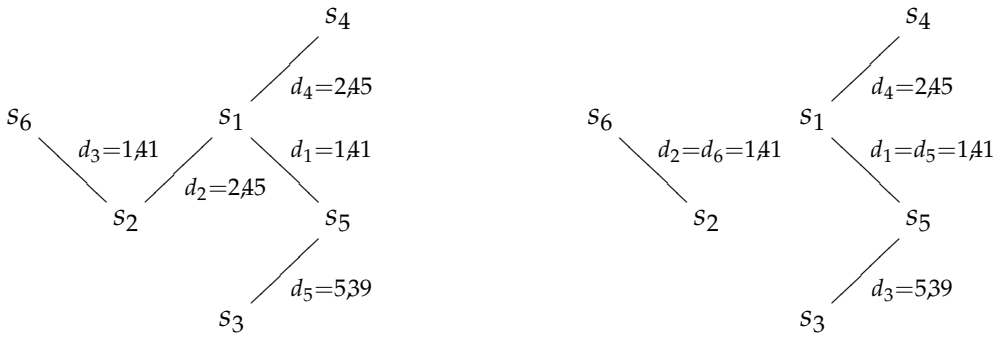
Figura 8.4. Rectas normais que dividem a frente não-dominada

8.4 Métodos para calcular a distribuição e extensão das soluções

São apresentados dois métodos para determinar a distribuição das soluções ao longo da frente óptima de Pareto e um outro para calcular a extensão desta.

O primeiro método proposto para calcular a distribuição da frente não-dominada, o processo inicia-se determinando uma função que descreva a frente óptima de Pareto (ver secção 8.3). A função deve ser válida entre as soluções extremas da frente. De seguida, divide-se a frente modelada através de rectas normais (figura 8.4). Entre cada duas rectas normais é afectado um intervalo I_i e as soluções localizadas nesse intervalo são contabilizadas. A extensão da frente é medida usando a distância da função modelada entre as soluções extremas encontradas pelo algoritmo.

O segundo método para calcular a diversidade da população é baseado no grafo de distâncias mínimas e denominado por MDG (*minimal distance graph*). O índice é calculado através do desvio padrão das distâncias que constituem o grafo de distâncias mínimas entre as soluções (8.2). Assim, o índice de desempenho usa os comprimentos mínimos que ligam todas as soluções não-dominadas da população. Por exemplo, considerem-se as soluções: $\{s_1 = (3,5,7),\ s_2 = (2,6,5),\ s_3 = (7,7,2),\ s_4 = (5,4,8),\ s_5 = (4,5,6)\},\ s_6 = (1,7,5)\}$. As distâncias mínimas que unem todas



(a) Grafo de distâncias mínimas (b) Distâncias do índice *SP*

Figura 8.5. Distâncias usadas pelos índices MDG e SP

as soluções são: $\{\overline{s_1s_2}, \overline{s_1s_4}, \overline{s_1s_5}, \overline{s_2s_6}, \overline{s_3s_5}\}$ como ilustra a figura 8.5(a). Este método permite relacionar todas as soluções e contrariamente ao método Δ'[1] (8.3) pode ser usado em problemas de qualquer dimensão. Também não usa a mesma distância mais do que uma vez, em oposição ao método SP (8.4) (figura 8.5(b)).

$$\mathrm{MDG}(S) = \sqrt{\frac{1}{\#S-2}\sum_{i=1}^{\#S-1}(d_i-\overline{d})^2} \tag{8.2}$$

$$\Delta'(S) = \sum_{i=1}^{\#S-1}\frac{|d_i-\overline{d}|}{\#S-1} \tag{8.3}$$

$$\mathrm{SP}(S) = \sqrt{\frac{1}{\#S-1}\sum_{i=1}^{\#S}(d_i-\overline{d})^2} \tag{8.4a}$$

$$d_i = \min_{s_k\in S\wedge s_k\neq s_i}\sum_{m=1}^{M}|f_m(s_i)-f_m(s_k)| \tag{8.4b}$$

[1]O método Δ' e SP são descritos na secção 3.5.1

8.5 Desempenho do algoritmo MaxiMin

8.5.1 Introdução

Para medir o desempenho do algoritmo de ordenação MaxiMin, compara-se este com os algoritmos pombalino [45] e o de agrupamento [63]. Para este fim é usado o algoritmo NSGA-II com o método pombalino, com o MDG (em vez do método pombalino) e com o de agrupamento (em substituição do método pombalino). Os resultados são comparados através de estatísticas obtidas num conjunto de $n = 101$ simulações efectuadas em funções bem estabelecidas.

8.5.2 Funções de teste

O conjunto de testes, para estudar o desempenho do algoritmo proposto, é formado por cinco funções bem conhecidas (8.5-8.8). Três delas $\{F_1, F_2, F_3\}$, propostas por Zitzler *et al.* [63] como ZDT1 (8.5), ZDT2 (8.6) e ZDT3 (8.7), cada uma delas com dois objectivos $\{f_1, f_2\}$. As outras duas $\{F_4, F_5\}$, usadas por Deb *et al.* em [138] como DTLZ2 (8.8) e DTLZ4, cada uma com três objectivos $\{f_1, f_2, f_3\}$.

$$F_1 = \begin{cases} f_1(\overrightarrow{x}) &= x_1 \\ g(\overrightarrow{x}) &= 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}} \\ f_2(\overrightarrow{x}) &= g(\overrightarrow{x}) h(f_1, g) \end{cases} \tag{8.5}$$

$$F_2 = \begin{cases} f_1(\overrightarrow{x}) &= x_1 \\ g(\overrightarrow{x}) &= 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) &= 1 - \left(\frac{f_1}{g}\right)^2 \\ f_2(\overrightarrow{x}) &= g(\overrightarrow{x}) h(f_1, g) \end{cases} \tag{8.6}$$

$$F_3 = \begin{cases} f_1(\overrightarrow{x}) &= x_1 \\ g(\overrightarrow{x}) &= 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} \sin(10\pi f_1) \\ f_2(\overrightarrow{x}) &= g(\overrightarrow{x}) h(f_1, g) \end{cases} \tag{8.7}$$

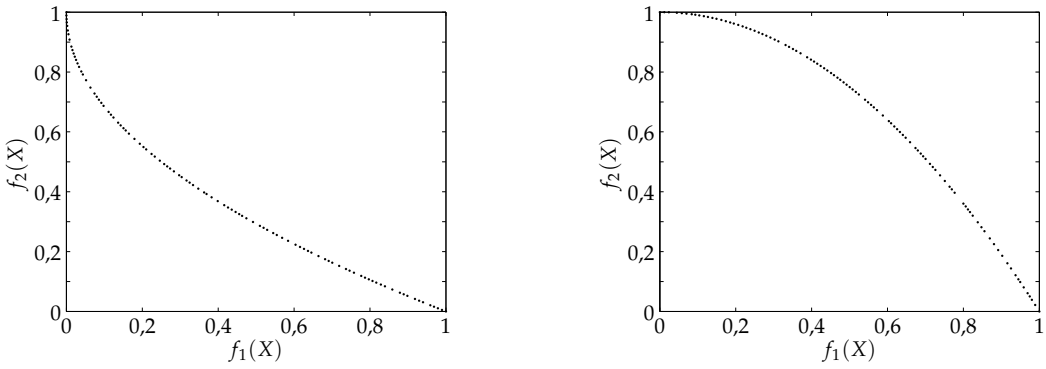
$$F_4 = \begin{cases} f_1(\overrightarrow{x}) &= [1 + g(\overrightarrow{x})] \cos(x_1 \pi/2) \cos(x_2 \pi/2) \\ f_2(\overrightarrow{x}) &= [1 + g(\overrightarrow{x})] \cos(x_1 \pi/2) \sin(x_2 \pi/2) \\ f_3(\overrightarrow{x}) &= [1 + g(\overrightarrow{x})] \sin(x_1 \pi/2) \\ g(\overrightarrow{x}) &= 1 + 9 \sum_{i=3}^{m} (x_i - 0.5)^2 \end{cases} \tag{8.8}$$

A função F_5 é idêntica a F_4 mas tem a seguinte mudança de variáveis $x_i \rightarrow x_i^{\alpha}$ com $\alpha = 100$. O vector $\overrightarrow{x}$ é formado por m parâmetros, $m_j = \{30, 30, 30, 12, 12\}$, $j = 1, \ldots, 5$, para $F_j = \{F_1, F_2, F_3, F_4, F_5\}$, com cada valor compreendido no intervalo $x_i \in [0, 1]$, $i = 1, \ldots, m_j$. O desempenho obtido é medido pelos índices MDG, Δ' e SP.

8.5.3 Resultados das funções teste

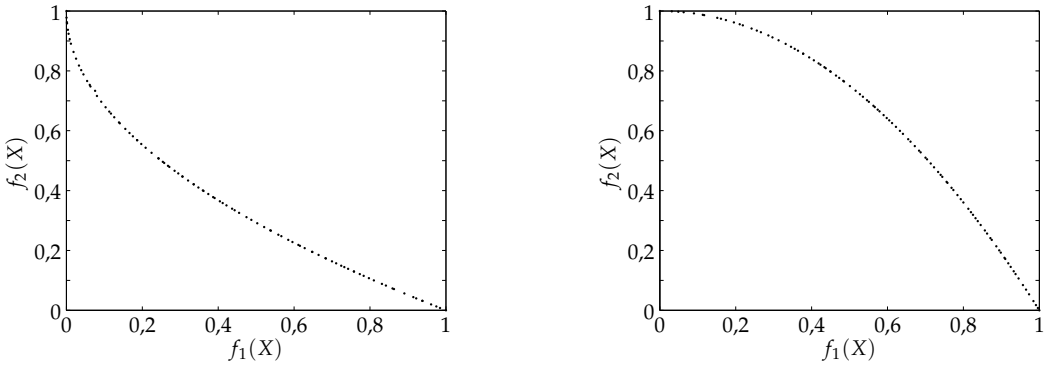
Para comparar os resultados dos três algoritmos são efectuadas $n = 101$ simulações e são analisados os resultados estatísticos como a mediana, média aritmética e desvio padrão. São também analisadas a melhor e a pior soluções obtidas.

Quando se optimizam as funções F_1 e F_2 todos os algoritmos apresentam uma boa diversidade (figuras 8.6-8.8). Na tabela 8.2 e 8.3 encontram-se os resultados estatísticos obtidos, respectivamente para a função F_1 e F_2. Nas duas últimas linhas estão os valores da melhor (linha Min) e pior (linha Max) solução encontradas nas experiências realizadas. Pode observar-se pelas tabelas 8.2 e 8.3, que o algoritmo MaxiMin obtém melhores resultados nos indices de desempenho em relação ao algoritmo pombalino. Adicionalmente, os piores resultados encontrados pelo esquema de ordenação MaxiMin estão na mesma gama que os melhores resultados obtidos pelo



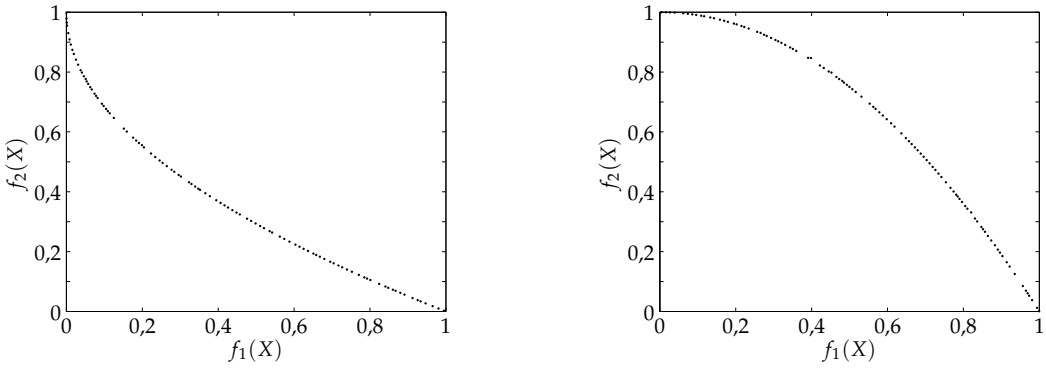
(a) Optimização da função F_1 (b) Optimização da função F_2

Figura 8.6. Optimização de F_1 e F_2 com o algoritmo MaxiMin



(a) Optimização da função F_1 (b) Optimização da função F_2

Figura 8.7. Optimização de F_1 e F_2 usando o algoritmo pombalino



(a) Optimização da função F_1 (b) Optimização da função F_2

Figura 8.8. Optimização de F_1 e F_2 utilizando o algoritmo de agrupamento

Tabela 8.2. Resultados da função de teste F_1

(a) Índice Δ'

	MaxiMin Δ'	Pombalino Δ'	Agrupamento Δ'
Mediana	**0,0667**	0,0747	0,0672
Média	**0,0668**	0,0752	0,0676
D. padrão	**0,0012**	0,0031	0,0032
Min	0,0643	0,0683	**0,0623**
Max	**0,0693**	0,0830	0,0846

(b) Índices SP e MDG

	MaxiMin		Pombalino		Agrupamento	
	SP	MDG	SP	MDG	SP	MDG
Mediana	0,0052	**0,0051**	0,0080	0,0072	**0,0048**	0,0058
Média	0,0052	**0,0051**	0,0080	0,0072	**0,0049**	0,0059
D. padrão	**0,0004**	**0,0002**	0,0006	0,0005	0,0012	0,0011
Min	0,0042	**0,0046**	0,0065	0,0061	**0,0034**	0,0047
Max	**0,0059**	**0,0056**	0,0095	0,0087	0,0151	0,0155

Tabela 8.3. Resultados da função de teste F_2

(a) Índice Δ'

	MaxiMin Δ'	Pombalino Δ'	Agrupamento Δ'
Mediana	**0,0668**	0,0745	0,0670
Média	**0,0667**	0,0750	**0,0667**
D. padrão	**0,0012**	0,0034	0,0026
Min	0,0635	0,0663	**0,0594**
Max	**0,0698**	0,0856	0,0723

(b) Índices SP e MDG

	MaxiMin		Pombalino		Agrupamento	
	SP	MDG	SP	MDG	SP	MDG
Mediana	0,0053	**0,0051**	0,0080	0,0072	**0,0049**	0,0057
Média	0,0053	**0,0051**	0,0080	0,0072	**0,0049**	0,0057
D. padrão	**0,0004**	**0,0002**	0,0006	0,0006	0,0006	0,0005
Min	0,0039	0,0047	0,0062	0,0059	**0,0034**	**0,0043**
Max	0,0064	**0,0057**	0,0097	0,0090	**0,0062**	0,0068

algoritmo pombalino. Contudo, em relação ao método de agrupamento os índices de desempenho não são consensuais, uma vez que fornecem resultados contraditórios. De qualquer forma, nos três métodos, os valores estatísticos resultantes estão muito próximos não sendo possível indicar com clareza o melhor método.

Na optimização da função de teste F_3 o algoritmo pombalino apresenta melhores resultados seguido do algoritmo de agrupamento. No entanto, o algoritmo de selecção MaxiMin obtém o melhor resultado em termos do valor mínimo atingido. A razão pela qual o algoritmo MaxiMin não obtém tão bons resultados deve-se à dificuldade do algoritmo em convergir, em certas experiências, para toda a frente óptima de Pareto, como é ilustrado na figura 8.9(a). Os resultados das experiências encontram-se na tabela 8.4.

Tabela 8.4. Resultados da função de teste F_3

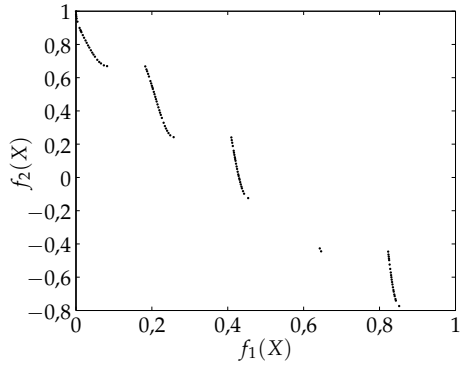
(a) Índice Δ'

	MaxiMin Δ'	Pombalino Δ'	Agrupamento Δ'
Mediana	0,1224	**0,1214**	0,1243
Média	0,1214	**0,1213**	0,1239
D. padrão	0,0064	**0,0051**	0,0064
Min	**0,1089**	0,1108	0,1113
Max	0,1382	0,1381	**0,1415**

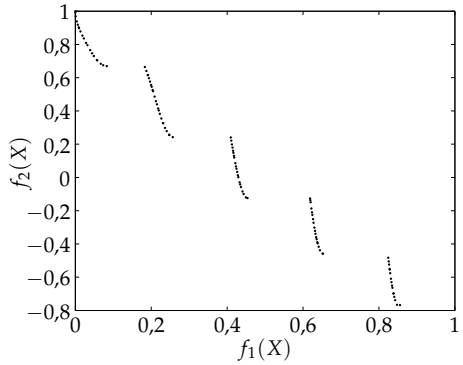
(b) Índices SP e MDG

	MaxiMin		Pombalino		Agrupamento	
	SP	MDG	SP	MDG	SP	MDG
Mediana	0,0136	0,0385	0,0082	**0,0340**	**0,0077**	0,0373
Média	0,0146	0,0371	**0,0106**	**0,0348**	0,0120	0,0365
D. padrão	0,0109	0,0060	**0,0073**	**0,0062**	0,0087	0,0063
Min	**0,0035**	**0,0267**	0,0056	0,0272	0,0036	0,0274
Max	0,0520	0,0486	0,0518	0,0544	**0,0505**	**0,0461**

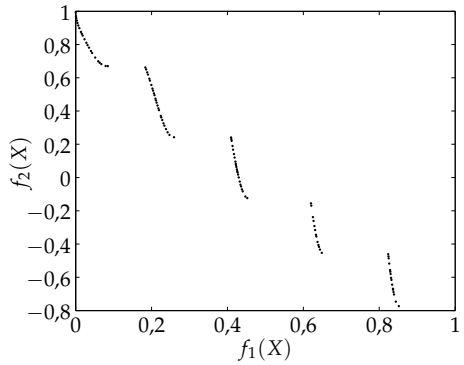
Para as funções de teste F_4 e F_5 o esquema de selecção MaxiMin obtém resultados claramente superiores ao método pombalino. Através dos resultados das simulações (figuras 8.10–8.15 e tabelas 8.5–8.6) pode observar-se que que o pior resultado obtido pelo algoritmo MaxiMin é superior ao obtido pelo método pombalino. No



(a) Optimização usando o algoritmo MaxiMin



(b) Optimização usando o algoritmo pombalino



(c) Optimização usando o algoritmo de agrupamento

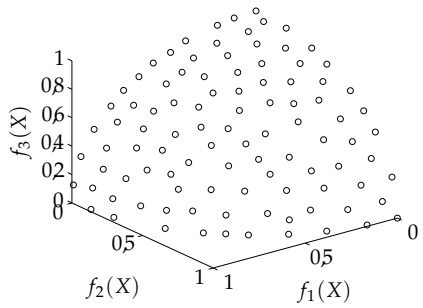
Figura 8.9. Optimização da função de teste F_3

que respeita a comparação entre os algoritmos MaxiMin e de agrupamento pode observar-se que em relação a F_4 os métodos apresentam resultados equivalentes. Contudo, na função de teste F_5 o método de selecção MaxiMin apresenta melhores resultados.

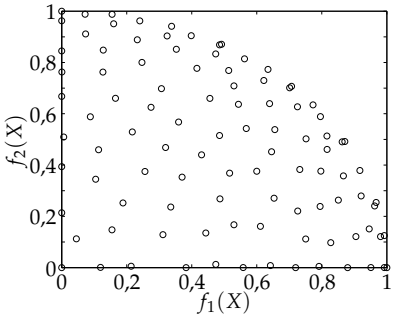
Tabela 8.5. Resultados da função de teste F_4

	MaxiMin		Pombalino		Agrupamento	
	SP	MDG	SP	MDG	SP	MDG
Mediana	**0,0286**	**0,0188**	0,0600	0,0484	0,0288	**0,0188**
Média	**0,0290**	0,0196	0,0606	0,0486	0,0291	**0,0193**
D. padrão	0,0038	0,0041	0,0056	0,0035	**0,0034**	**0,0028**
Min	0,0201	**0,0146**	0,0489	0,0426	**0,0200**	0,0148
Max	0,0477	0,0428	0,0776	0,0598	**0,0415**	**0,0304**

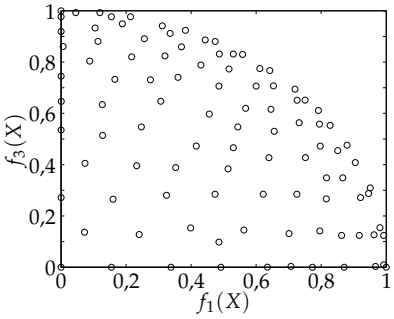
Como os valores estatísticos estão muito próximos, para aferir a superioridade de algum dos métodos, estes valores são submetidos ao teste de Mann-Whitney (ver



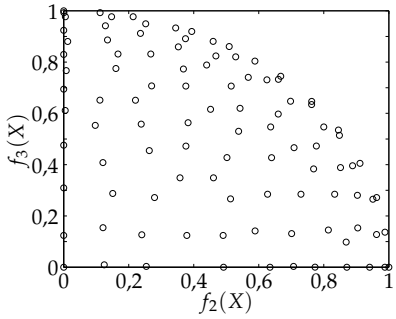
(a) Frente não dominada



(b) Projecção no plano $f_1 \times f_2$

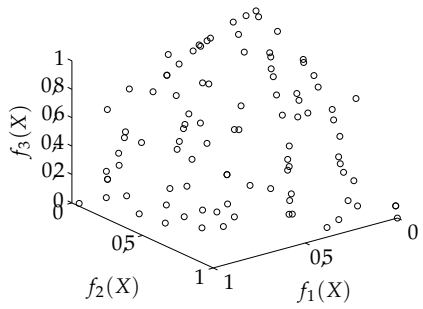


(c) Projecção no plano $f_1 \times f_3$

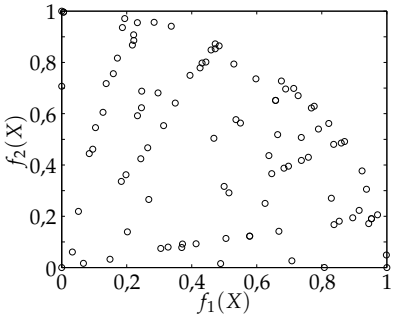


(d) Projecção no plano $f_2 \times f_3$

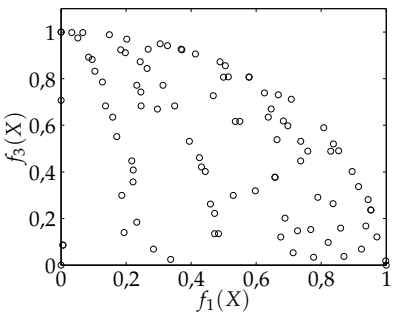
Figura 8.10. Optimização da função F_4 usando o algoritmo de selecção MaxiMin



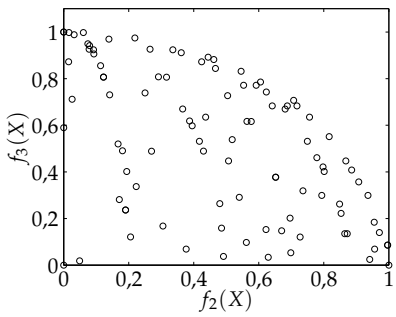
(a) Frente não dominada



(b) Projecção no plano $f_1 \times f_2$

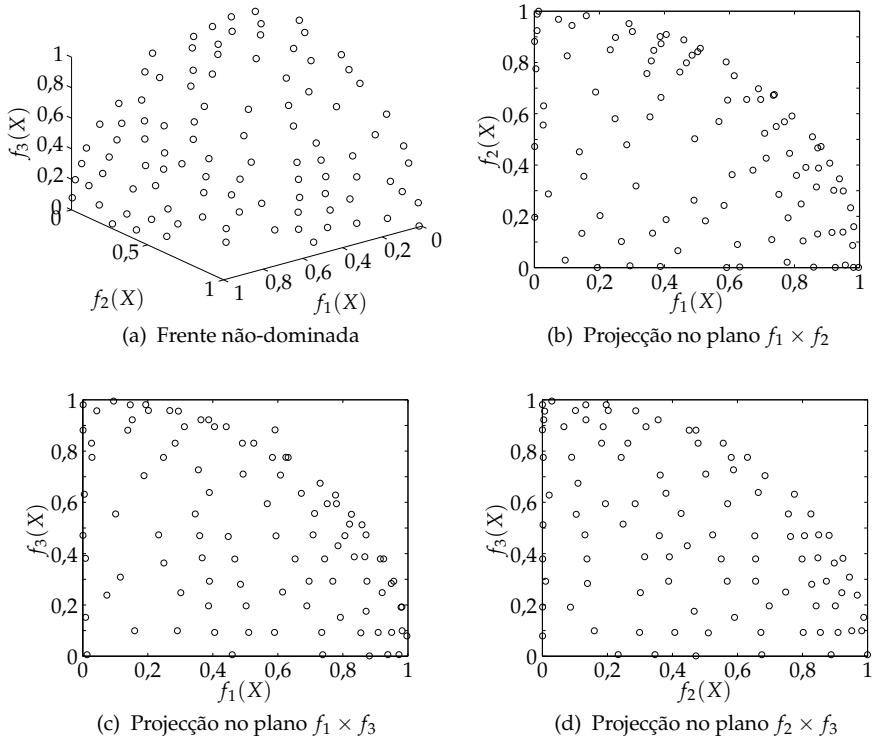


(c) Projecção no plano $f_1 \times f_3$



(d) Projecção no plano $f_2 \times f_3$

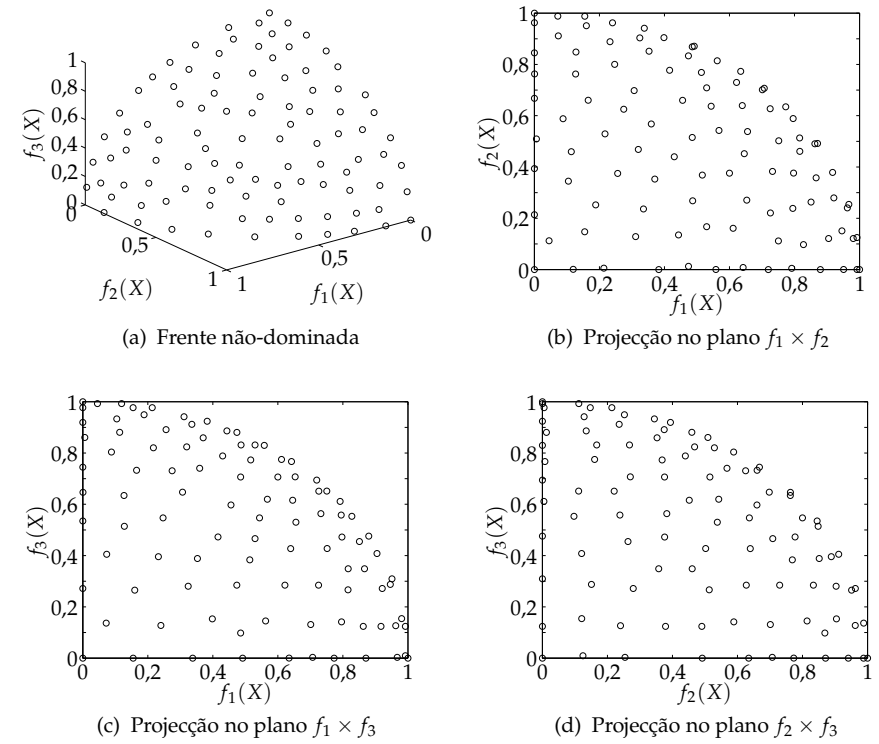
Figura 8.11. Optimização da função F_4 usando o algoritmo pombalino



(a) Frente não-dominada (b) Projecção no plano $f_1 \times f_2$

(c) Projecção no plano $f_1 \times f_3$ (d) Projecção no plano $f_2 \times f_3$

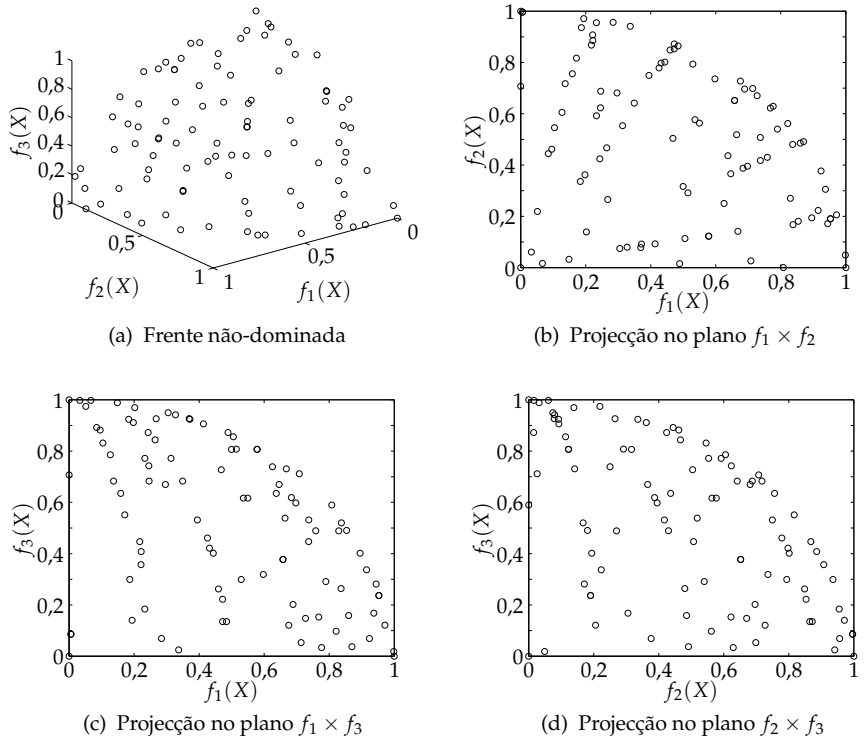
Figura 8.12. Optimização da função F_4 usando o algoritmo de agrupamento



(a) Frente não-dominada (b) Projecção no plano $f_1 \times f_2$

(c) Projecção no plano $f_1 \times f_3$ (d) Projecção no plano $f_2 \times f_3$

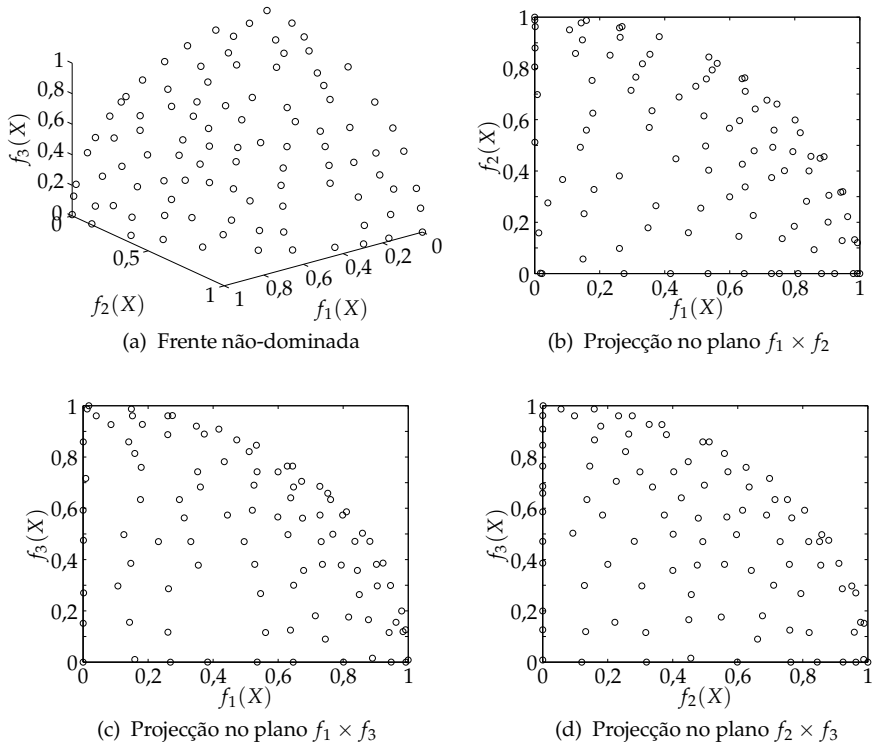
Figura 8.13. Optimização da função F_5 usando o algoritmo MaxiMin



(a) Frente não-dominada (b) Projecção no plano $f_1 \times f_2$

(c) Projecção no plano $f_1 \times f_3$ (d) Projecção no plano $f_2 \times f_3$

Figura 8.14. Optimização da função F_5 usando o algoritmo pombalino



(a) Frente não-dominada (b) Projecção no plano $f_1 \times f_2$

(c) Projecção no plano $f_1 \times f_3$ (d) Projecção no plano $f_2 \times f_3$

Figura 8.15. Optimização da função F_5 usando o algoritmo de agrupamento

Tabela 8.6. Resultados da função de teste F_5

	MaxiMin		Pombalino		Agrupamento	
	SP	MDG	SP	MDG	SP	MDG
Mediana	**0,0275**	**0,0172**	0,0598	0,0477	0,0318	0,0225
Média	**0,0278**	**0,0173**	0,0598	0,0475	0,0316	0,0229
D. padrão	**0,0033**	**0,0023**	0,0051	0,0033	0,0036	0,0035
Min	**0,0211**	**0,0126**	0,0444	0,0397	0,0236	0,0157
Max	**0,0383**	**0,0253**	0,0728	0,0553	0,0427	0,0416

apêndice B) para verificar qual dos métodos é significativamente superior, *i.e* considerando a hipótese alternativa como H_1: *os métodos são significativamente diferentes*.

O resultado do teste de Mann-Whitney entre o esquema MaxiMin e os algoritmos pombalino e de agrupamento encontram-se na tabela 8.7, onde a primeira coluna indica a função de teste e nas colunas seguintes estão, respectivamente os resultados para os índices de desempenho SP, Δ' e MDG. Da tabela 8.7(a) conclui-se que o método de selecção MaxiMin tem um desempenho superior ao do algoritmo pombalino para as funções de teste F_1, F_2, F_4 e F_5 pois os resultados da analise estatística são significativos com $p \leq 0{,}001$ (teste de Mann-Whitney, bilateral; $z < -12$). No entanto, para a função F_3 o ordenamento pombalino é significativamente melhor com uma probabilidade de $p > 0{,}01$ quando se usa o índice de desempenho MDG. Nos dois índices restantes (Δ' e SP) os resultados não distinguem significativamente o desempenho de qualquer um dos métodos. Pela tabela 8.7(b) o teste de Mann-Whitney apresenta valores contraditórios dependendo do índice usado levando a concluir que os teste são equivalentes. Contudo, comparando as figuras 8.10 e 8.13 respectivamente com as figuras 8.12 e 8.15, pode observar-se que as soluções estão espaçadas mais uniformemente no método MaxMin. Este fenómeno é mais notório na função de teste F_5. De facto, se se usar o teste direccional, *i.e.* se a hipótese alternativa for H_2: *o algoritmo MaxiMin tem um desempenho melhor que o algoritmo de agrupamento*, na função F_5 obtém-se uma significância com $p \leq 0{,}05$ (teste de Mann-Whitney, unilateral; $z = \min(|-6{,}9726|, |-1{,}6309|)$). Adicionalmente o método de selecção MaxiMin apresenta uma complexidade mais reduzida $O((pop_{\dim})^2)$ do

que o algoritmo de agrupamento $O((pop_{\dim})^3)$.

Tabela 8.7. Resultados estatísticos dos testes de Mann-Whitney

(a) Teste de Mann-Whitney entre os algoritmos MaxiMin e pombalino

	SP	Δ'	MDG
F_1	$-12{,}2758$	$-12{,}2421$	$-12{,}2758$
F_2	$-12{,}2734$	$-12{,}0254$	$-12{,}2758$
F_3	$+1{,}7922$	$+0{,}2684$	$+2{,}7311$
F_4	$-12{,}2758$	–	$-12{,}2734$
F_5	$-12{,}2758$	–	$-12{,}2758$

(b) Teste de Mann-Whitney entre os algoritmos MaxiMin e de agrupamento

	SP	Δ'	MDG
F_1	$+5{,}4127$	$-10{,}1791$	$-1{,}6791$
F_2	$+4{,}4691$	$-8{,}8719$	$-0{,}6476$
F_3	$+1{,}4889$	$+0{,}4634$	$-2{,}7142$
F_4	$-0{,}5164$	–	$+0{,}5428$
F_5	$-6{,}9726$	–	$-1{,}6309$

8.6 Planeamento de trajectórias para manipuladores planares

8.6.1 Introdução

Nesta secção são executados simulações com manipuladores planares sendo os resultados das frentes óptimas de Pareto analisados com os métodos propostos na secção 8.3 e 8.4.

8.6.2 Formulação do problema

Pretende-se mover um manipulador robótico planar iR com $i = \{2;3;4\}$ entre o ponto $A \equiv \{1{,}2;-0{,}3\}$ e o ponto $B \equiv \{-0{,}5;1{,}4\}$, optimizando o deslocamento angular, f_q, e o deslocamento do órgão terminal, f_p, (8.9). As configurações inicial e final, para o manipulador $2R$, são calculadas usando a cinemática inversa. Para

os manipuladores $3R$ e $4R$ as configurações são determinadas de modo a apresentar uma configuração o mais parecida com o manipulador $2R$.

Nas simulações usam-se os seguintes parâmetros: o número de elementos da população é de $pop_{\text{dim}} = 300$; o número de gerações é de $T_{\text{t}} = 1500$; a probabilidade de cruzamento é de $p_{\text{c}} = 0{,}6$; a probabilidade de mutação é de $p_{\text{m}} = 0{,}05$; o comprimento dos elos é de $l = 2/i$ m e a massa dos elos é de $m = 2/i$ kg.

$$f_q = \sum_{j=1}^{n} \sum_{l=1}^{i} \left(\dot{q}_l^{(j\Delta t,T)} \right)^2 \tag{8.9a}$$

$$f_p = \sum_{j=2}^{n} d\left(p_j,\ p_{j-1}\right)^2 \tag{8.9b}$$

A representação do vector, na geração T, que representa a trajectória é a seguinte:

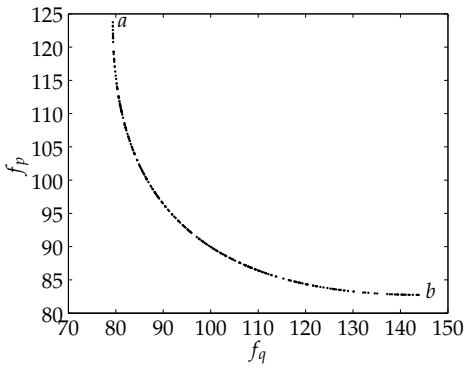
$$[\{q_1^{(\Delta t,T)}, .., q_i^{(\Delta t,T)}\}, \ldots, \{q_1^{(2\Delta t,T)}, .., q_i^{(2\Delta t,T)}\}, \ldots, \{q_1^{((n-2)\Delta t,T)}, .., q_i^{((n-2)\Delta t,T)}\}] \tag{8.10}$$

O número de configurações da trajectória é de $n = 8$. Como as configurações inicial e final se mantêm inalteradas durante toda a evolução, estas configurações não são codificadas na trajectória. O tempo considerado entre duas configurações sucessivas é de $\Delta t = 0{,}1$ s.

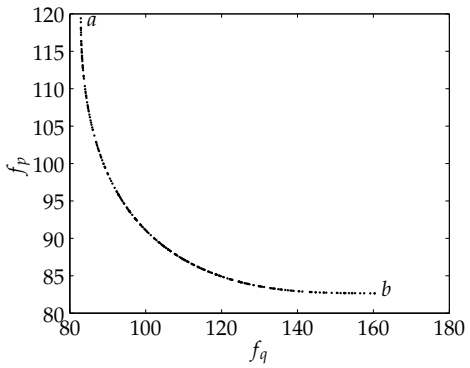
Na resolução do problema é usado um algoritmo multi-objectivo elitista $(\mu + \lambda)$, sendo a selecção efectuada pelo método do posto de Pareto (*Pareto ranking*) e o esquema de partilha com $\sigma_{\text{partilha}} = 0{,}01$ e $\alpha = 2$ no domínio dos parâmetros. No algoritmo usado, a métrica entra em linha de conta com todas as soluções da população independentemente do seu posto. Posteriormente, após ser gerada a população dos descendentes a nova população é determinada pelo método proposto na secção 8.2.

8.6.3 Análise dos resultados das experiências

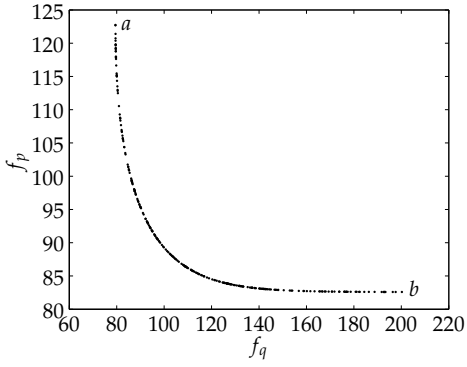
Nesta secção são apresentados os resultados de um conjunto de testes realizados. Para cada tipo de testes são executadas várias experiências até obter $n_{\text{exp}} = 21$ simulações validas[2], nomeadamente para os manipuladores $2R$, $3R$ e $4R$. Na figura 8.16 encontram-se uma das frentes óptima de Pareto obtida para cada tipo de manipulador referido. A convergência do algoritmo para a frente óptima de Pareto é de 95%, 57% e 38%, respectivamente para os manipuladores $2R$, $3R$ e $4R$. Esta diminuição na convergência deve-se ao aumento exponencial das frentes locais com o aumento do número de elos do manipulador aumentando assim a probabilidade de o algoritmo encontrar e convergir para uma dessas frentes.



(a) Manipulador $2R$



(b) Manipulador $3R$



(c) Manipulador $4R$

Figura 8.16. Frentes óptimas de Pareto

[2] *i.e.* são executadas um número de experiências até obter n_{exp} experiências que converjam para a frente óptima de Pareto

Em todas as simulações as frentes encontradas podem ser modeladas pela equação (8.11a) ($\kappa, \alpha, \beta \in \mathbb{R}$). A mediana, média e desvio padrão obtidos para os parâmetros κ, α e β encontram-se na tabela 8.8. A aproximação da frente pelas equações é válida somente entre as soluções não-dominadas extremas ('a' e 'b'). As frentes encontradas também podem ser modeladas pela função densidade de probabilidade de Pareto, equação (8.11b), com assimptotas em $q = \alpha$ e $p = \beta$, mas esta aproximação apresenta um erro ligeiramente superior à equação anterior não sendo assim considerada neste trabalho.

$$p(q) = \kappa \frac{q + \alpha}{q + \beta} \tag{8.11a}$$

$$p(q) = \frac{\kappa}{(q - \alpha)^{\gamma}} + \beta \tag{8.11b}$$

Tabela 8.8. Estatísticas dos parâmetros das frentes

	Frente 2R			Frente 3R			Frente 4R		
	κ	α	β	κ	α	β	κ	α	β
Mediana	77,90	−66,83	−71,21	78,68	−71,36	−75,12	80,28	−71,82	−74,44
Média	77,99	−66,74	−71,13	78,76	−71,33	−75,19	80,15	−71,49	−74,34
D. Padrão	0,44	0,71	0,42	0,58	1,67	1,22	0,60	2,21	1,62

Da tabela 8.8 pode verificar-se que a média e a mediana têm valores praticamente idênticos. Adicionalmente, o desvio padrão apresenta valores relativamente baixos, o que permite concluir que o algoritmo converge para a mesma frente ou seja a frente de óptima de Pareto. Por outro lado, à medida que se aumenta o número de elos o desvio padrão aumenta, *i.e.*, à medida que se aumenta a complexidade do problema o algoritmo tem mais dificuldade em obter sempre a mesma frente não-dominada.

A extensão da frente para os manipuladores iR, com $i = \{2, 3, 4\}$, têm média de $\mu_{\mathrm{P}} = \{86{,}57; 105{,}61; 200{,}49\}$ e desvio padrão de $\sigma_{\mathrm{P}} = \{2{,}00; 17{,}48; 48{,}94\}$. Também aqui se pode concluir que, à medida que aumenta a complexidade do problema, o algoritmo tem mais dificuldade em obter frentes sempre com a mesma extensão.

A diversidade estatística das soluções ao longo da frente não-dominada é apresentada nas figuras 8.17-8.19. A percentagem final média de soluções pertencentes à frente não-dominada é de $\mu_1 = \{96{,}15\%; 92{,}53\%; 88{,}25\%\}$ e com desvio padrão de $\sigma_1 = \{1{,}26; 5{,}21; 4{,}28\}$. Através das figuras pode verificar-se que as soluções estão distribuídas ao longo de todos os intervalos. No entanto, essa distribuição não é uniforme e a uniformidade diminui à medida que se aumenta o número de elos do manipulador. Isto deve-se à percentagem de soluções da frente não-dominada diminuir à medida que se aumenta o número de elos do manipulador, pois o algoritmo favorece as soluções não-dominadas, e só depois é que entra em linha de conta com a diversidade. Consequentemente, somente quando se atinge uma população com 100% de soluções não-dominadas é que se tira maior proveito do algoritmo proposto. Pode ainda verificar-se pelas figuras que os intervalos com menos soluções não-dominadas são compensados pelo algoritmo, atribuindo-lhe mais soluções dominadas de modo a tentar manter uma boa distribuição de soluções em todos os intervalos.

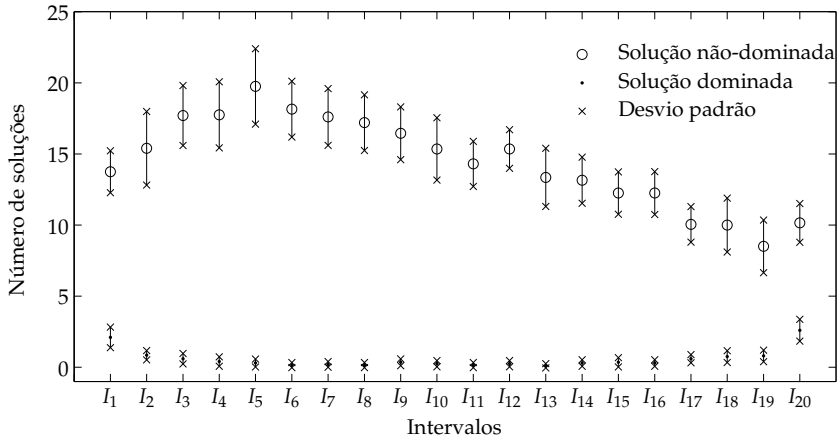


Figura 8.17. Distribuição das soluções ao longo da frente 2*R*

Na figura 8.20 encontram-se soluções extremas ('*a*' e '*b*') correspondentes às frentes óptimas de Pareto ilustradas na figura 8.16. A figura ilustra as configurações sucessivas e o deslocamento angular para os manipuladores 2*R*, 3*R* e 4*R* das soluções referidas.

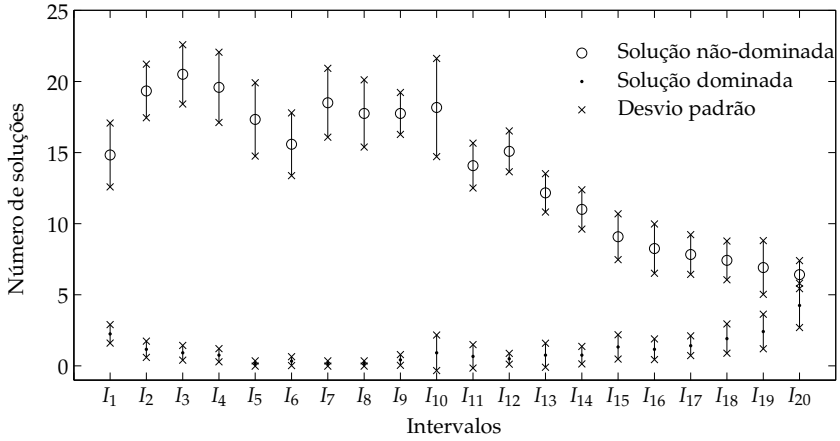


Figura 8.18. Distribuição das soluções ao longo da frente 3*R*

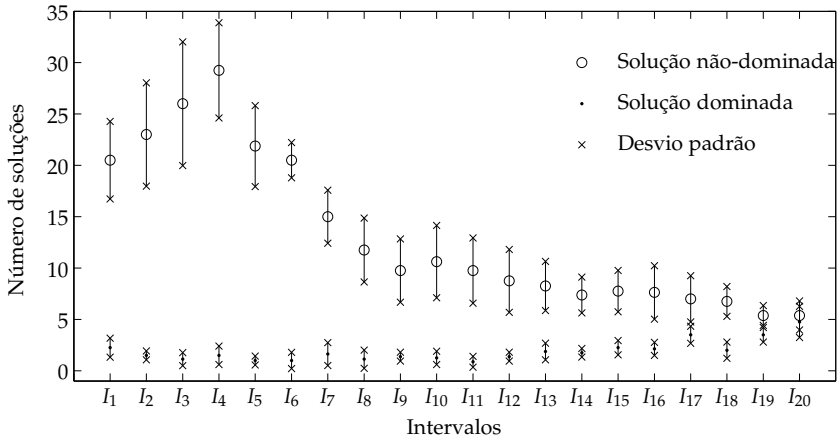
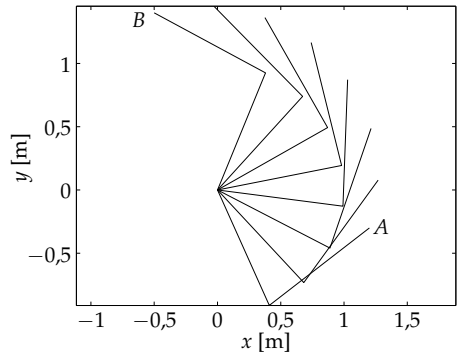


Figura 8.19. Distribuição das soluções ao longo da frente 4*R*

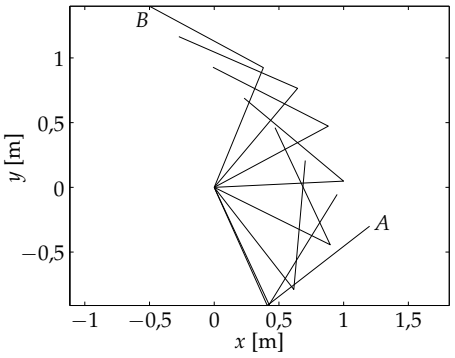
8.7 Conclusões

Neste capítulo apresentam-se as seguintes técnicas:

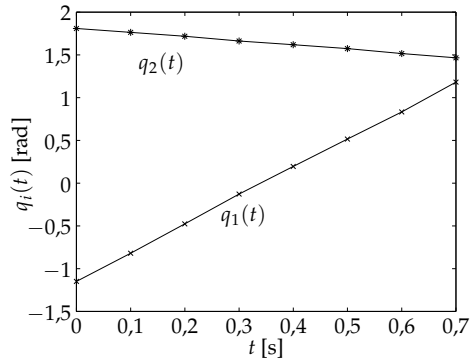
- O algoritmo MaxiMin para promover a diversidade das soluções não-dominadas ao longo da frente óptima de Pareto. O algoritmo proposto além de manter uma boa diversidade das soluções garante uma uniformidade das mesmas no espaço dos objectivos. O algoritmo foi testado em várias funções padrão e comparado com duas técnicas bastante utilizadas para o mesmo fim nos algoritmos AEMO: o algoritmo pombalino e o algoritmo de agrupamento. Os



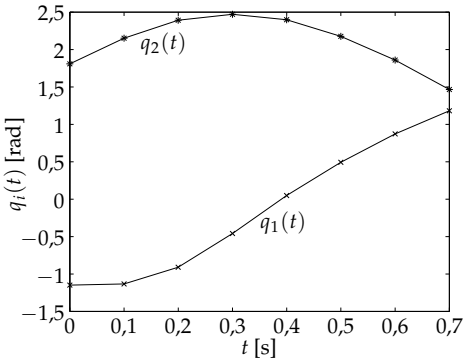
(a) Configurações sucessivas do manipulador 2*R* (solução *a*, figura 8.16(a))



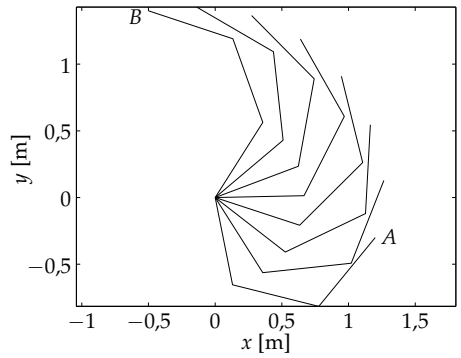
(b) Configurações sucessivas do manipulador 2*R* (solução *b*, figura 8.16(a))



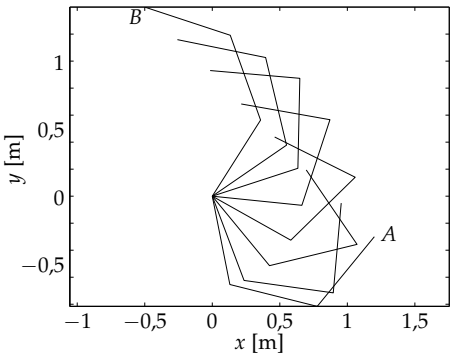
(c) Deslocamento das juntas do manipulador 2*R*, (solução *a*, figura 8.16(a))



(d) Deslocamento das juntas do manipulador 2*R* (solução *b*, figura 8.16(a))



(e) Configurações sucessivas do manipulador 3*R* (solução *a*, figura 8.16(b))

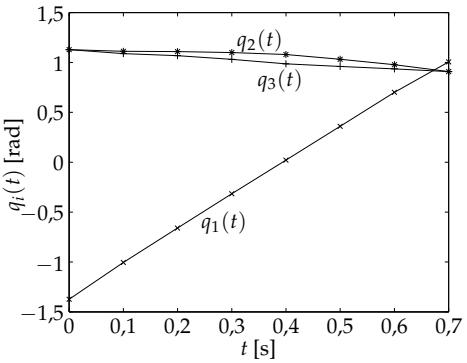


(f) Configurações sucessivas do manipulador 3*R* (solução *b*, figura 8.16(b))

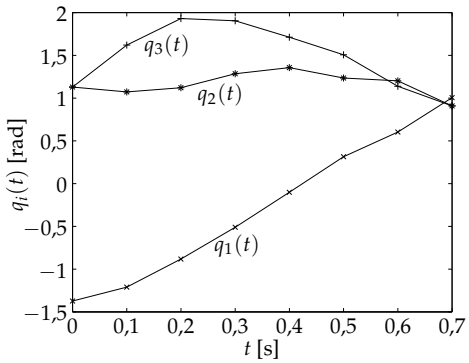
Figura 8.20. Trajectórias pertencentes a frente óptima de Pareto

resultados obtidos com o MaxiMin superam os correspondentes aos do pombalino e são semelhantes aos obtidos com o método de agrupamento.

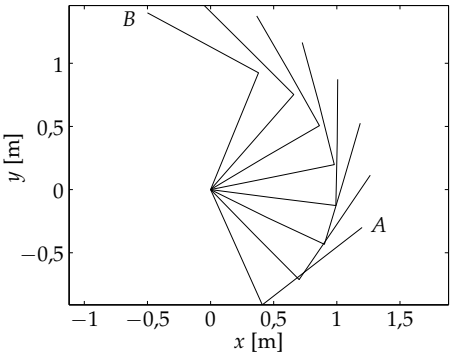
- Um método para analisar a convergência dos AEMO.



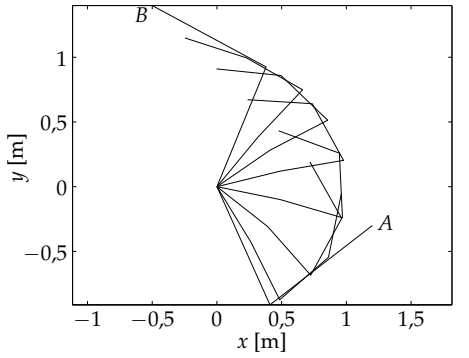
(g) Deslocamento das juntas do manipulador 3*R* (solução *a*, figura 8.16(b))



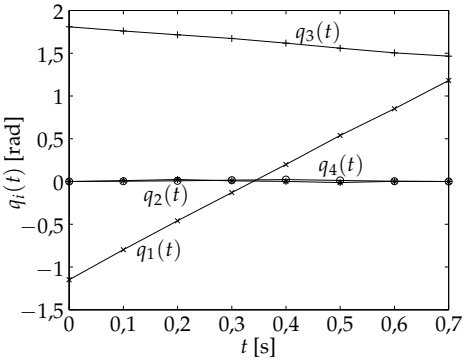
(h) Deslocamento das juntas do manipulador 3*R* (solução *b*, figura 8.16(b))



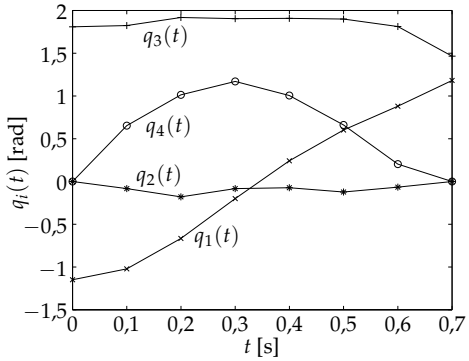
(i) Configurações sucessivas do manipulador 4*R* (solução *a*, figura 8.16(c))



(j) Configurações sucessivas do manipulador 4*R* (solução *b*, figura 8.16(c))



(k) Deslocamento das juntas do manipulador 4*R* (solução *a*, figura 8.16(c))



(l) Deslocamento das juntas do manipulador 4*R* (solução *b*, figura 8.16(c))

Figura 8.20. Trajectórias pertencentes a frente óptima de Pareto *(cont.)*

- Um método para medir a extensão da frente óptima de Pareto.
- Duas técnicas para estudar a diversidade de soluções.

As técnicas anteriores foram aplicadas ao estudo de um planeador de trajectórias obtendo-se bons resultados. O planeador consegue manter uma boa diversidade de soluções da frente, especialmente quando a população contém uma percentagem apreciável de soluções não-dominadas

9 Dinâmica dos algoritmos genéticos

9.1 Introdução

Um sistema pode ser visto como uma agregação de objectos unidos através de uma forma de interacção ou de interdependência. Quando um ou mais sinais (variáveis) do sistema variam no tempo é referido, na generalidade, como existindo uma dinâmica do sistema.

O primeiro passo na análise de um sistema consiste em estabelecer os sinais de interesse e a relação entre eles. O conceito principal da teoria dos sistemas associado ao seu comportamento, deduzido das propriedades dos subsistemas ou elementos que o compõem, é a sua interacção.

Os fenómenos (sinais) que ocorrem fora do sistema e actuam nele e não são afectados pelo que ocorre dentro do sistema são chamados de sinais de entrada. Por outro lado, os sinais que são afectados pela acção desses fenómenos externos são denominados por sinais de saída. O conceito matemático da dinâmica do sistema pode ser considerado uma "estrutura" que recebe uma entrada em cada instante t e emite uma ou mais saídas. Na maior parte dos casos, a(s) saída(s) não depende(m)

apenas dos sinais de entrada actuais mas também dos instantes anteriores. Numa visão abrangente, o conceito de sistema pode ser empregue para além dos sistemas físicos, podendo ser aplicado a fenómenos abstractos tais como sistemas biológicos, económicos *et caetera*.

Este capítulo estuda a dinâmica dos AGs. O algoritmo é visto como um sistema constituído por sinais de entrada (*e.g.*, probabilidade de mutação, cruzamento, *etc.*) e de saída (*e.g.*, valor de aptidão do melhor elemento da população). O capítulo está estruturado em duas partes principais. A primeira parte, secção 9.2, apresenta o estudo da dinâmica de um AG simples. A segunda parte, secção 9.3, desenvolve o estudo da dinâmica de um planeador de trajectórias. Na secção 9.4 são enunciadas as principais conclusões.

9.2 Dinâmica de um algoritmo genético simples

9.2.1 Introdução

Nesta secção, a dinâmica de um AG é estudada durante a sua evolução ao longo das gerações. Para investigar o fenómeno ocorrido na evolução da população do AG, a probabilidade de mutação é exposta à excitação de perturbações durante algumas gerações sendo a variação do valor de aptidão analisado. Para medir a influência da função de aptidão, são utilizadas quatro funções de aptidão diferentes.

Esta secção está organizada nas seguintes partes: na secção 9.2.2 são formuladas as funções de aptidão utilizadas para estudar a dinâmica do AG. De seguida, na secção 9.2.3 é descrita a modelação do AG, é indicado o sinal perturbador e são apresentados alguns resultados quando se optimiza uma função de aptidão. Nas secções 9.2.4 e 9.2.5 são apresentados outros resultados complementares com funções objectivo diferentes. Por último, na secção 9.2.6, são enumeradas as principais conclusões.

9.2.2 Funções de optimização

Esta secção descreve os AGs que são utilizados para estudar a dinâmica deste tipo de sistemas. Para investigar a influência da função de aptidão nos AGs, recorre-se a quatro funções de aptidão diferentes (9.1), todas elas de minimização. As funções têm todas um parâmetro correspondente ao comprimento do vector com $l = 24$ elementos binários e a representação de cada vector da população é dada por (9.2). O parâmetro real b, obtido pela descodificação do vector $\vec{b}$, pode variar no intervalo $[-50000, 50000]$.

$$f_A(b) = 1 + |b - 41| \tag{9.1a}$$

$$f_B(b) = 1 + |b - 41|^2 \tag{9.1b}$$

$$f_B(b) = 1 + |b - 41|^3 \tag{9.1c}$$

$$f_C(b) = 2 - \mathrm{sinc}(b - 41) \tag{9.1d}$$

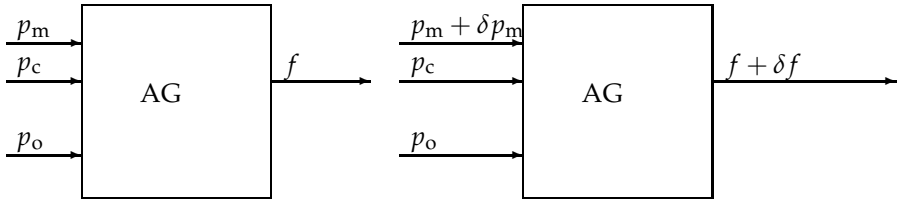
$$\vec{b} = \{b_1, b_2, b_3, \ldots, b_l\} \tag{9.2}$$

Em todas as experiências, envolvendo as quatro funções de teste, a população inicial é idêntica. O mesmo se aplica aos parâmetros genéticos, nomeadamente: uma população de 50 vectores, optimizada durante 200 gerações sob a selecção por posto, o cruzamento simples com $p_\mathrm{c} = 0{,}8$ e a mutação por lugar com $p_\mathrm{m} = 0{,}05$. A melhor solução é sempre transposta para a geração seguinte.

O único parâmetro que varia ao longo das experiências, dentro de cada função de teste, é a semente inicial de ruído que é adicionado à probabilidade de mutação. Este vector de sementes iniciais, onde cada elemento é utilizado numa experiência diferente, é igual para todas as funções de teste.

9.2.3 Modelação, propagação do sinal e dinâmica do sistema AG

Nesta secção, os AGs são modelados através do sistema representado na figura 9.1(a). As entradas consideradas são as probabilidades de mutação e de cruzamento, entre outros parâmetros, e a saída é o melhor valor de aptidão encontrado na população nessa geração. Devido à complexidade do sistema o estudo incide essencialmente na probabilidade de mutação, mantendo-se os restantes sinais de entrada constantes. Assim, o sistema AG é estimulado pela perturbação da probabilidade de mutação, p_{m}, através de um sinal de ruído branco δp_{m} de amplitude pequena, e a variação da aptidão da população correspondente, δf, é avaliada com o fim de medir a influência deste operador no AG (ver figura 9.1). Os sinais de probabilidade de cruzamento, p_{c}, e os restantes, p_{o}, permanecem inalterados. Consequentemente, a variação da probabilidade de mutação e a modificação do valor de aptidão do melhor elemento da população, durante a evolução, podem ser vistos como os valores dos sinais de entrada e de saída, que variam ao longo das sucessivas gerações. Esta análise é efectuada realizando várias experiências com sementes diferentes para o sinal de ruído com amplitude pequena. Todas as sementes dos sinais restantes permanecem inalteradas.



(a) Sistema inicial (b) Sistema perturbado

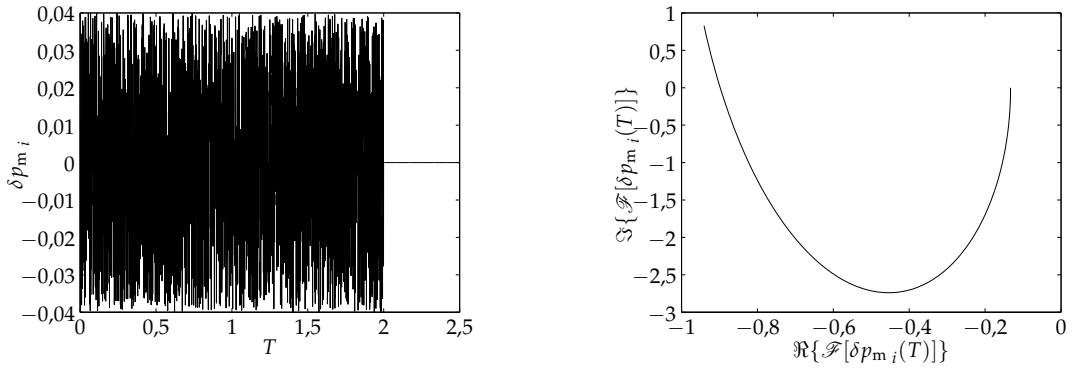
Figura 9.1. Dinâmica do sistema

Nesta perspectiva, um sinal de ruído branco $\delta p_{\mathrm{m}}(T)$ é adicionado, no inicio da evolução de cada AG, aos vectores da probabilidade de mutação p_{m} durante um período de tempo T_{exc}. A nova probabilidade de mutação p_{n} é calculada através da equação (9.3) onde η é o sinal de ruído branco com amplitude máxima $\pm\Delta p$.

$$p_{\mathrm{n}} = \begin{cases} 0 & \text{if } p_{\mathrm{m}} + \eta(\Delta p) < 0 \\ 1 & \text{if } p_{\mathrm{m}} + \eta(\Delta p) > 1 \\ p_{\mathrm{m}} + \eta(\Delta p) & \text{outros casos} \end{cases} \tag{9.3}$$

Consequentemente, o sinal de entrada, na geração T, é a diferença entre os dois casos, isto é $\delta p_{\mathrm{m}}(T) = p_{\mathrm{n}}(T) - p_{\mathrm{m}}(T)$. Por outro lado, o sinal de saída é a diferença da aptidão da população com e sem sinal de ruído, ou seja, $\delta f(T) = f_{\mathrm{n}}(T) - f(T)$.

A figura 9.2 ilustra o sinal de entrada $\delta p_{\mathrm{m}}(T)$, com semente $i = 1$, no domínio das gerações e o diagrama polar correspondente, para f_A, $\Delta p = 0{,}04$ e $T_{exc} = 2$, onde $\mathscr{F}[\delta p_{\mathrm{m}}(T)]$ representa a transformada de Fourier do sinal perturbador. A figura 9.3 apresenta a variação do sinal de saída $\delta f(T)$ correspondente.



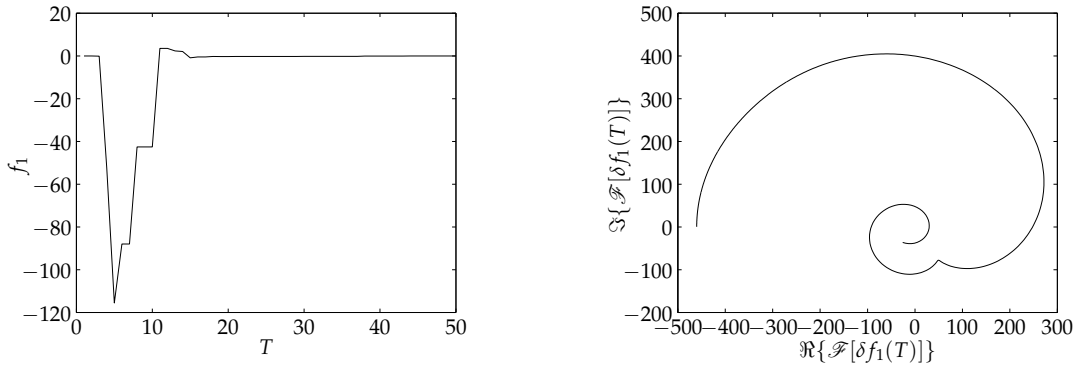
(a) Domínio das gerações (b) Diagrama polar

Figura 9.2. Sinal de entrada perturbador δp_{m} durante $T_{\mathrm{exc}} = 2$ gerações com semente $i = 1$ ($\Delta p = 0{,}04$, função f_A)

Após obter a transformada de Fourier dos sinais de entrada e de saída é possível obter a respectiva função de transferência (9.4) para a semente i.

$$H_i(\mathrm{j}w) = \frac{\mathscr{F}\{\delta f_i(T)\}}{\mathscr{F}\{\delta p_{\mathrm{m}\,i}(T)\}}, \quad i = \{1, \ldots, n\} \tag{9.4}$$

A função de transferência $H_i(\mathrm{j}w)$, com semente $i = 1$, entre os sinais de entrada e de saída, é ilustrada na figura 9.4.



(a) Domínio das gerações (b) Diagrama polar

Figura 9.3. Sinal de saída $\delta f(T)$ para o sinal de entrada perturbador durante $T_{\text{exc}} = 2$ gerações com semente $i = 1$ ($\Delta p = 0{,}04$; aptidão f_A)

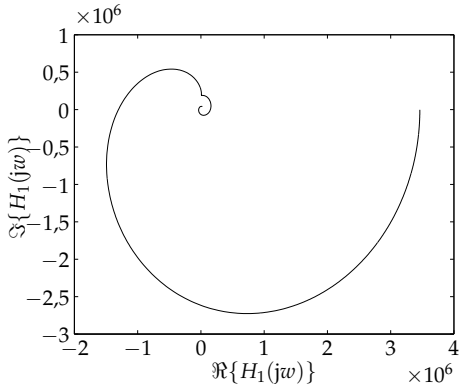


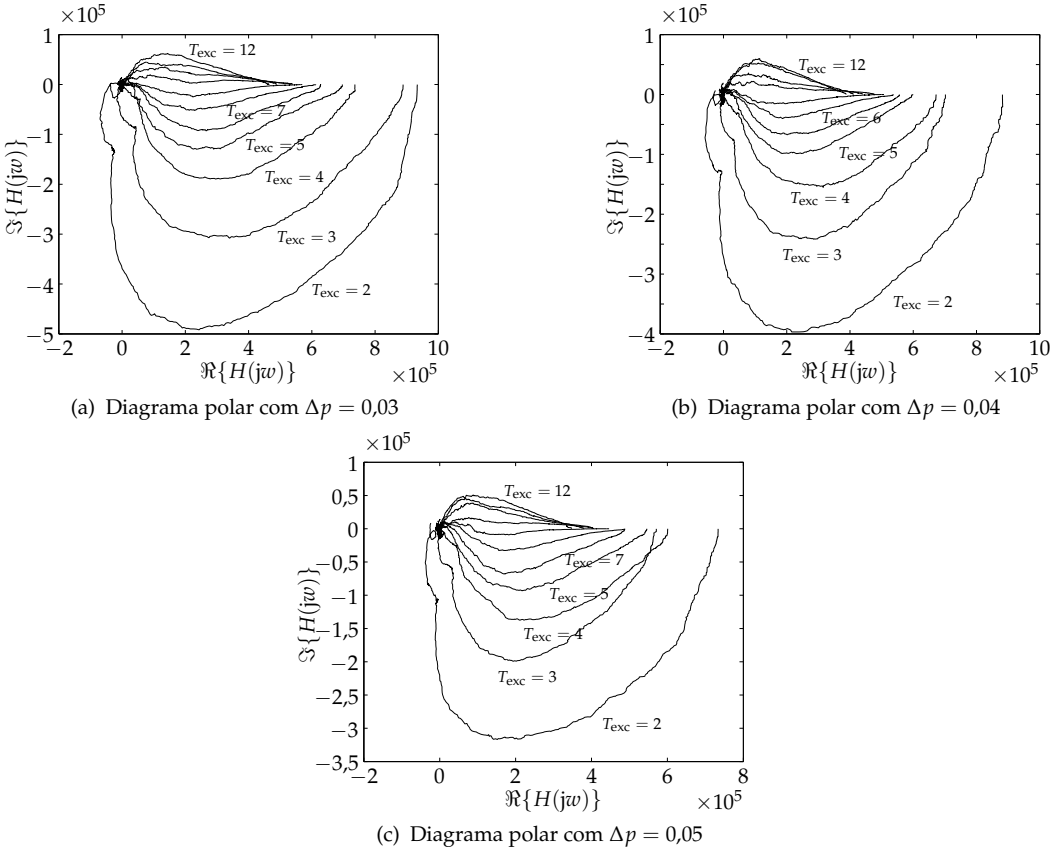
Figura 9.4. Função de transferência $H_1(\mathrm{j}w)$ usando a semente $i = 1$ ($T_{\text{exc}} = 2$, $\Delta p = 0{,}04$, aptidão f_A)

Após se obterem todas as funções de transferência, uma para cada semente, a função de transferência "representativa" é calculada através da mediana da amostra estatística [139, 140].

Identificação da função de transferência

Nesta secção a mediana numérica da função de transferência do sistema, figura 9.5, é aproximada por expressões analíticas com ganho $\kappa \in \mathbb{R}^+$ e dois pólos $(a, b) \in \mathbb{R}^+$ de ordem fraccionária, respectivamente $(\alpha, \beta) \in \mathbb{R}^+$, dada pela equação (9.5) onde $l = A$ para a função f_A. A equação (9.5) que aproxima as funções de transferência é escolhida de forma heurística.

$$G_l(\mathrm{j}w) = \frac{\kappa}{\left(\frac{\mathrm{j}w}{a}+1\right)^{\alpha}\left(-\frac{\mathrm{j}w}{b}+1\right)^{\beta}} \qquad \{\kappa, a, b, \alpha, \beta\} \in \mathbb{R}^{+} \tag{9.5}$$



(a) Diagrama polar com $\Delta p = 0{,}03$

(b) Diagrama polar com $\Delta p = 0{,}04$

(c) Diagrama polar com $\Delta p = 0{,}05$

Figura 9.5. Diagrama polar de $H_A(\mathrm{j}w)$ com $\Delta p = \{0{,}03; 0{,}04; 0{,}05\}$

É de notar que a expressão da função de transferência (9.5) é de fase não-mínima adoptando pólos dos semi-planos esquerdo e direito, respectivamente 'a' e 'b'.

Com o fim de estimar os parâmetros da função de aptidão, um AG adoptando um vector de valores reais é executado com a seguinte representação $\{k,\ a,\ b,\ \alpha,\ \beta\}$. O AG de identificação é simulado durante $T_{ide} = 600$ gerações com uma população de 100 elementos. É usado o cruzamento binário simulado e o operador de mutação altera o vector $\{x_1,\ldots,x_5\} \equiv \{k,\ a,\ b,\ \alpha,\ \beta\}$ segundo a equação (9.6) onde u_i é um número aleatório que segue uma distribuição uniforme U e ε_i é fixado de acordo com o intervalo de estimação.

$$x_{i+1} = 10^{u_i} x_i \tag{9.6a}$$

$$u_i \sim U[-\varepsilon_i, +\varepsilon_i] \tag{9.6b}$$

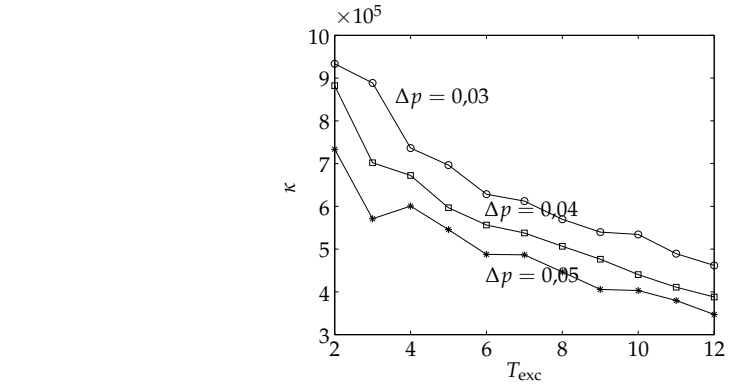
A função de aptidão f_{ide} mede a distância cartesiana entre a mediana $H(\mathrm{j}w_k)$ e a função de transferência analítica $G(\mathrm{j}w_k)$:

$$f_{\mathrm{ide}} = \sum_{k=1}^{nf} \parallel H(\mathrm{j}w_k) - G(\mathrm{j}w_k) \parallel \tag{9.7}$$

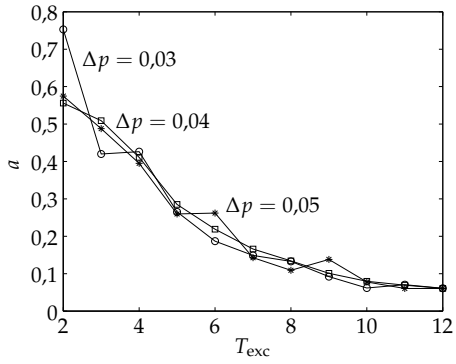
onde: H representa a mediana de n funções de transferência que resultam das experiências com semente diferente, nf é o número total de pontos de amostragem e w_k, $k = \{1, ..., nf\}$ é o vector de frequências correspondente.

Uma vez que o AG de optimização tem uma dinâmica estocástica, cada execução do AG, com uma semente de ruído distinta, conduz a uma função de transferência diferente. Consequentemente, de modo a obter uma convergência numérica [139] são realizadas $n = 1701$ experiências com sementes diferentes para o sinal de ruído perturbador $\delta p_{\mathrm{m}}(T)$ (todas as restantes sementes dos vectores dos sinais de p_{m}, p_{c} e probabilidade de selecção permanecem inalteradas). Deste modo, a função de transferência do AG de optimização é calculada a partir da transformada de Fourier (TF) para cada par de sinais de entrada e de saída. Posteriormente, a mediana das funções de transferência calculadas previamente (*i.e.*, a mediana da parte real e a mediana da parte imaginária, para cada frequência) é tomada como o valor final da função de transferência numérica $H(\mathrm{j}w)$ (ver figura 9.5).

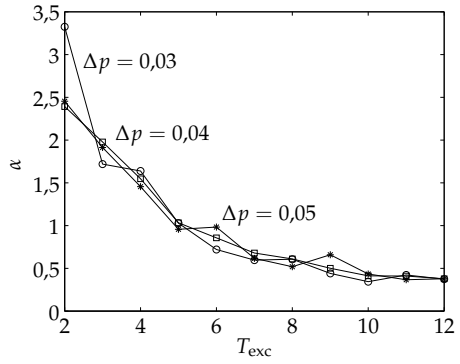
Para estudar a influência do período de excitação T_{exc} várias simulações são executadas entre $T_{\mathrm{exc}} = 2$ e $T_{\mathrm{exc}} = 12$ gerações. A relação entre os parâmetros da função de transferência (9.5) e T_{exc} estão ilustrados na figura 9.6.



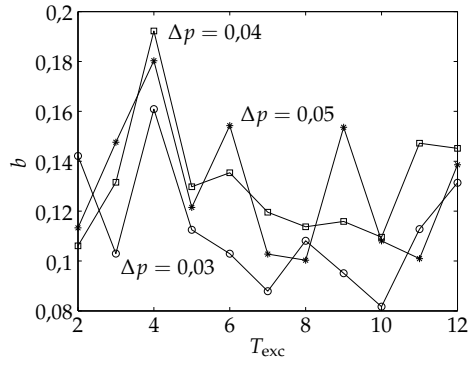
(a) Ganho κ *vs.* T_{exc}



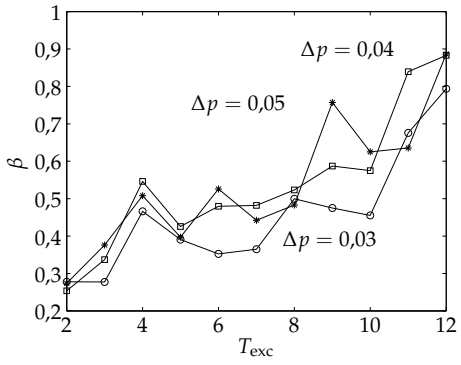
(b) Pólo a *vs.* T_{exc}



(c) Pólo de ordem fraccionária α *vs.* T_{exc}



(d) Pólo b *vs.* T_{exc}



(e) Pólo de ordem fraccionária β *vs.* T_{exc}

Figura 9.6. Parâmetros estimados $\{\kappa, a, \alpha, b, \beta\}$ *vs.* T_{exc} com a função f_A

Os gráficos de $\{\kappa, a, b, \alpha, \beta\}$ podem ser aproximados através da equação (9.8) resultando nos parâmetros da tabela 9.1.

$$\{\kappa, a, \alpha, b, \beta\} \simeq \gamma_1 \, (T_{exc})^{\gamma_2} \tag{9.8}$$

Tabela 9.1. Parâmetros γ_i, $i = \{1;2\}$ da aproximação de $\{\kappa, a, \alpha, b, \beta\}$ com a função de optimização f_A

	κ		a		α		b		β	
Δp	$\gamma_1(10^5)$	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2
0,03	10,0	$-0{,}40$	2,46	$-1{,}48$	7,78	$-1{,}27$	0,15	$-0{,}16$	0,18	0,48
0,04	10,0	$-0{,}43$	2,13	$-1{,}37$	6,30	$-1{,}14$	0,13	$+0{,}01$	0,18	0,57
0,05	9,6	$-0{,}38$	2,12	$-1{,}38$	6,13	$-1{,}12$	0,15	$-0{,}09$	0,20	0,52

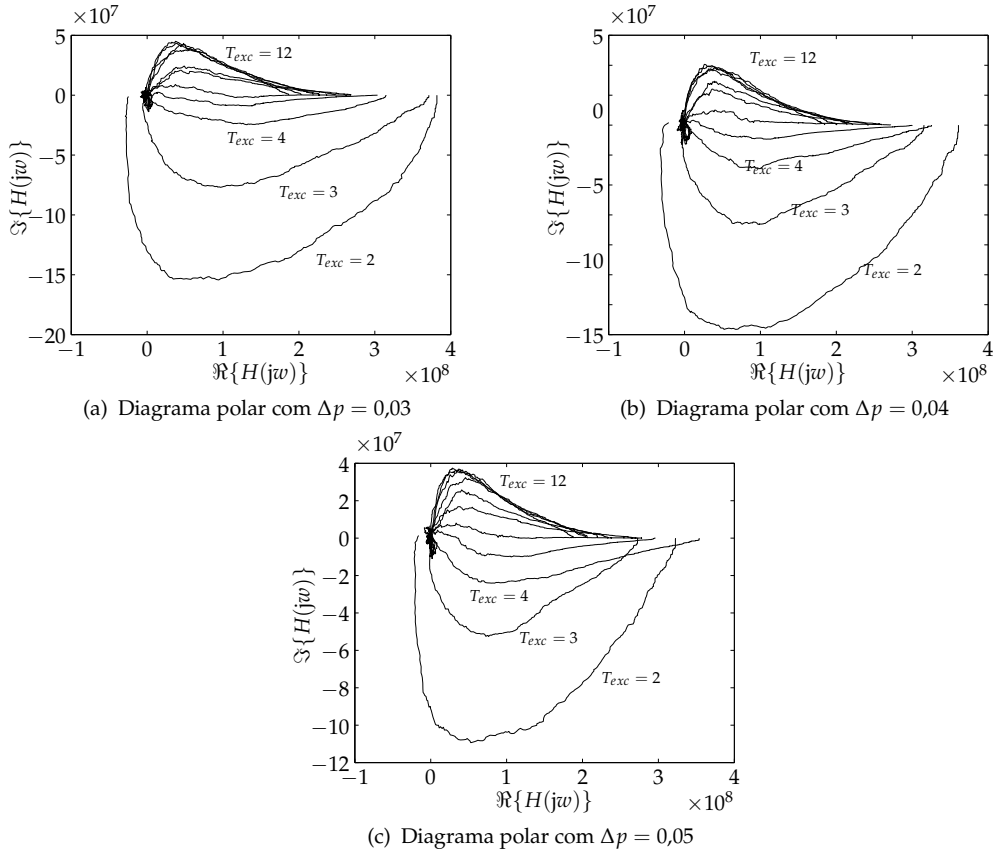
Estes resultados revelam que os parâmetros da função de transferência $\{\kappa, a, \alpha, b, \beta\}$ variam de acordo com uma exponencial em função do tempo T_{exc}. O pólo do semi-plano direito tem uma dependência menor com T_{exc} e, consequentemente, a adopção de um valor peculiar de T_{exc} não é particularmente relevante, mas por outro lado, o pólo do semi-esquerdo 'a' tem uma variação assinalável. Adicionalmente, κ varia significativamente com Δp enquanto que a, α, b, β tem uma dependência menor relativamente a Δp.

Permitindo variar livremente a ordem dos pólos, são obtidos valores não-inteiros para α e β. De facto, a adopção de uma função de transferência de ordem-inteira levaria à necessidade de um número elevado de pólos para obter a mesma "qualidade" na função analítica, identificada a partir dos valores numéricos.

9.2.4 Experiências com as funções de aptidão f_B e f_C

Nesta secção são apresentados os diagrama polares e a aproximação dos parâmetros correspondentes quando são usadas as funções de optimizações f_B e f_C.

Os diagramas polares obtidos para a função f_B podem ser vistos na figura 9.7 e os correspondentes parâmetros da função de transferência encontram-se na figura 9.8. Quando se aproximam os parâmetros através de expressões exponenciais, do mesmo tipo que as utilizadas na função de optimização f_A, resultam os valores apresentados na tabela 9.2.



(a) Diagrama polar com $\Delta p = 0{,}03$

(b) Diagrama polar com $\Delta p = 0{,}04$

(c) Diagrama polar com $\Delta p = 0{,}05$

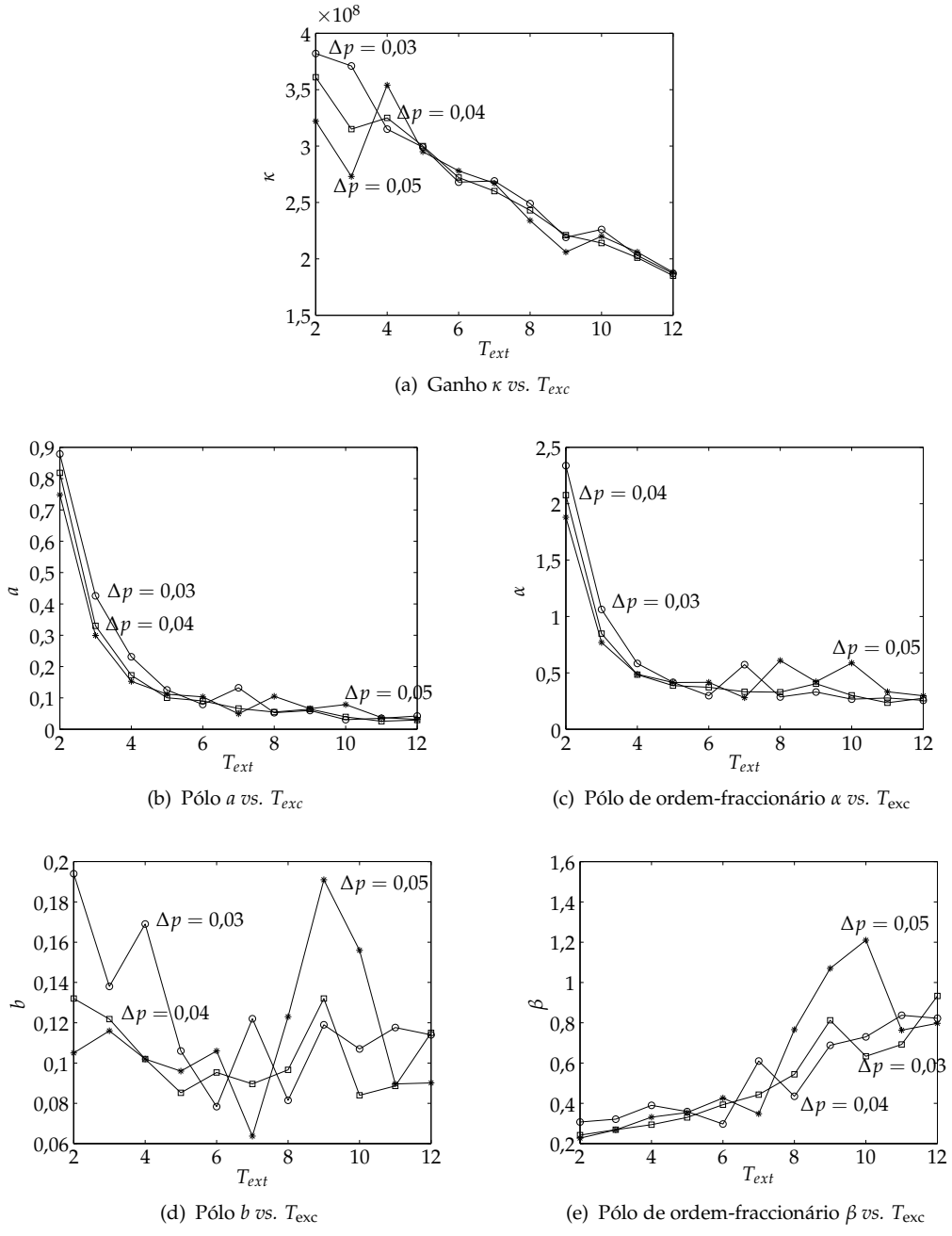
Figura 9.7. Diagramas polares $H_B(\mathrm{j}w)$ com $\Delta p = \{0{,}03; 0{,}04; 0{,}05\}$

Os resultados obtidos com a função f_B não diferem significativamente dos resultados anteriores, com excepção do parâmetro κ que revela uma variação menos significativa com Δp, particularmente para valores elevados de T_{exc}.

Tabela 9.2. Parâmetros γ_i, $i = \{1;2\}$ da aproximação de $\{\kappa, a, \alpha, b, \beta\}$ com a função de optimização f_B

	κ		a		α		b		β	
Δp	$\gamma_1(10^8)$	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2
0,03	5,0	−0,41	2,93	−1,83	0,20	−0,29	3,55	−1,13	0,16	0,61
0,04	5,0	−0,36	2,40	−1,81	0,13	−0,13	2,64	−0,99	0,11	0,77
0,05	4,0	−0,30	1,62	−1,52	0,10	+0,05	1,85	−0,72	0,10	0,90

Para a função f_C os diagramas polares obtidos encontram-se na figura 9.9 e a estimação dos parâmetros da função de transferência pode ser vista na figura 9.10. Com estes resultados os parâmetros são também aproximados através da expressão



(a) Ganho κ *vs.* T_{exc}

(b) Pólo a *vs.* T_{exc}

(c) Pólo de ordem-fraccionário α *vs.* T_{exc}

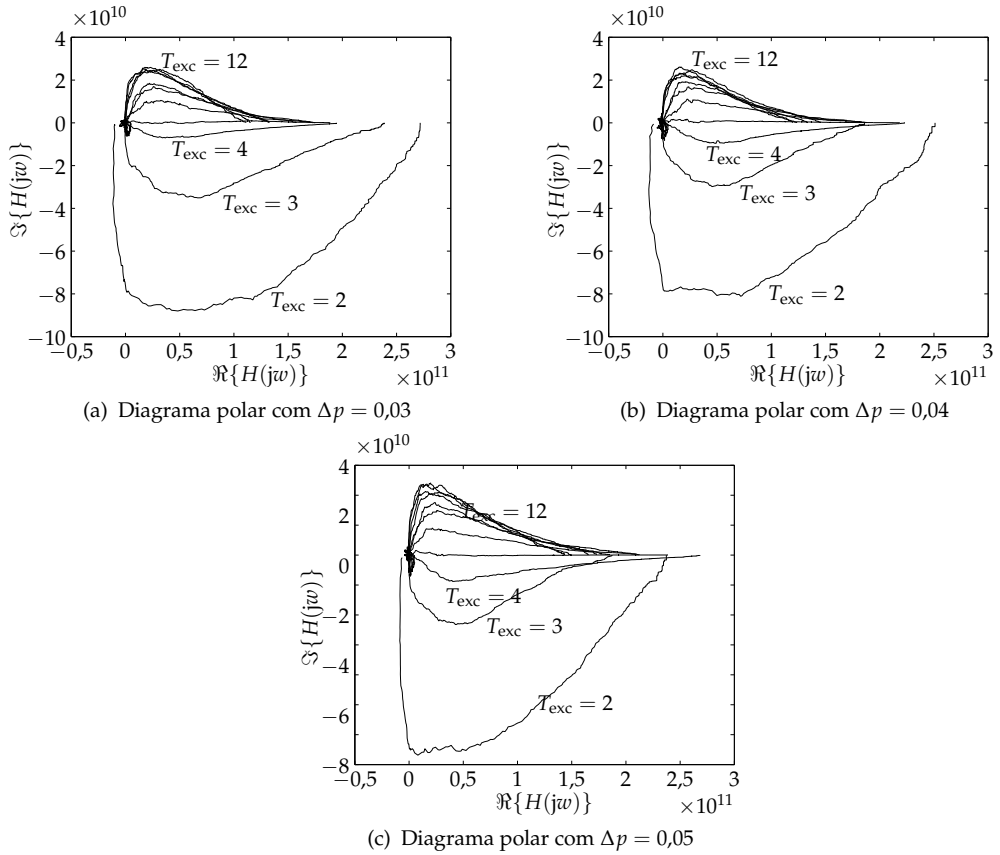
(d) Pólo b *vs.* T_{exc}

(e) Pólo de ordem-fraccionário β *vs.* T_{exc}

Figura 9.8. Parâmetros estimados $\{\kappa, a, \alpha, b, \beta\}$ *vs.* T_{exc} com a função f_B

exponencial usada anteriormente (9.3).

Os resultados obtidos com a função de aptidão f_C não diferem significativamente dos resultados anteriores, com excepção para o parâmetro κ que revela um comportamento inicial ($T_{\text{exc}} \leq 3$) idêntico aos obtidos para as funções f_A e f_B. Para $T_{\text{exc}} \geq 4$



(a) Diagrama polar com $\Delta p = 0{,}03$

(b) Diagrama polar com $\Delta p = 0{,}04$

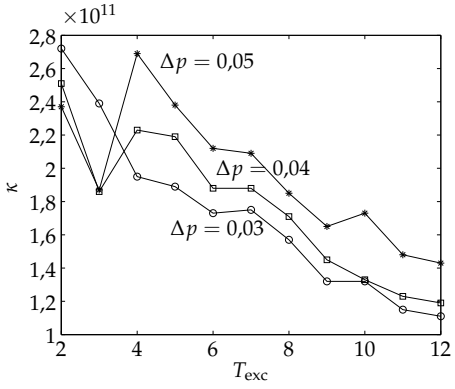
(c) Diagrama polar com $\Delta p = 0{,}05$

Figura 9.9. Diagramas polares $H_C(\mathrm{j}w)$ com $\Delta p = \{0{,}03; 0{,}04; 0{,}05\}$

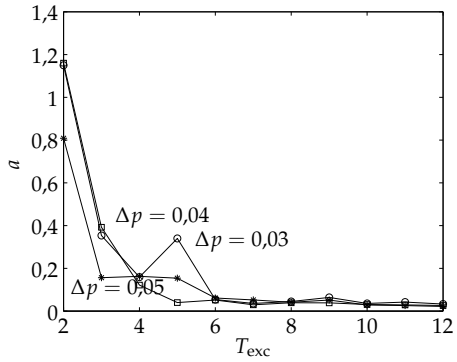
o parâmetro κ varia de forma visível com Δp, inversamente proporcional a Δp. Com o aumento do expoente de $|b - 41|$, da função (9.1), o valor do ganho aumenta significativamente. Por outro lado, a gama dos parâmetros de 'a' e 'α' é idêntica, apenas a concavidade de acentua com o aumento da exponencial da função de optimização.

Tabela 9.3. Parâmetros γ_i, $i = \{1;2\}$ da aproximação de $\{\kappa, a, \alpha, b, \beta\}$ com a função de optimização f_C

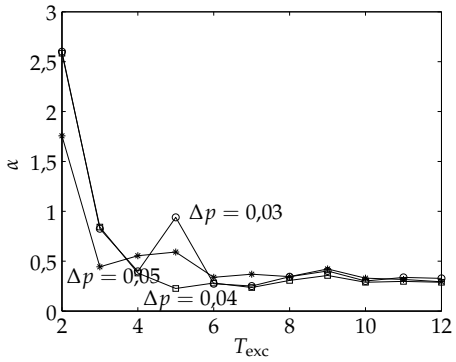
	κ		a		α		b		β	
Δp	$\gamma_1(10^11)$	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2
0,03	4,0	−0,49	3,30	−1,95	0,17	−0,18	2,90	−1,00	0,16	0,73
0,04	3,0	−0,39	2,61	−2,02	0,14	−0,25	2,35	−0,99	0,14	0,67
0,05	3,0	+0,56	2,03	−1,83	0,15	−0,29	1,72	−0,74	0,16	0,64



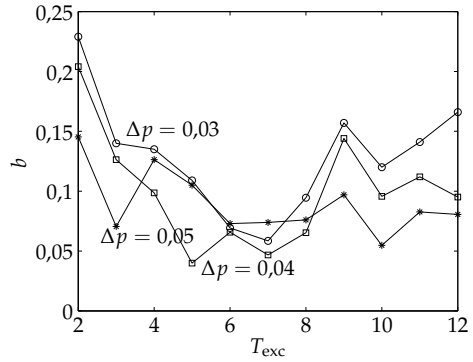
(a) Ganho κ *vs.* T_{exc}



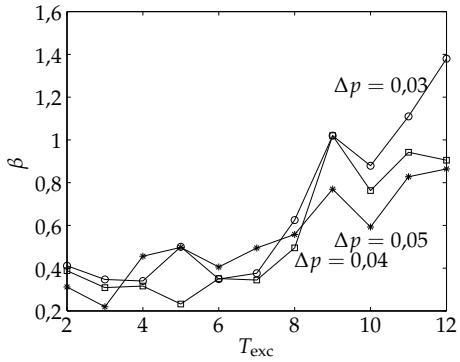
(b) Pólo a *vs.* T_{exc}



(c) Pólo de ordem-fraccionário α *vs.* T_{exc}



(d) Pólo b *vs.* T_{exc}



(e) Pólo de ordem-fraccionário β *vs.* T_{exc}

Figura 9.10. Parâmetros estimados $\{\kappa, a, \alpha, b, \beta\}$ *vs.* T_{exc} com a função f_C

9.2.5 Experiências com a função de optimização f_D

O estudo da função de optimização f_D segue uma estratégia idêntica à que foi adoptada nas secções anteriores. Para esta equação, e para um sinal perturbador do tipo ruído branco δp_{m}, é obtido um sinal de saída e uma função de transferência como

é ilustrado nas figuras 9.11 e 9.12. Neste caso, a saída é menos sensível e responde apenas algumas gerações depois de ser aplicado o sinal de ruído. Consequentemente, a optimização deste AG conduz a um tipo de diagrama polar diferente para $H(\mathrm{j}w)$ (ver figura 9.13). Neste caso a mediana $H(\mathrm{j}w)$ pode ser aproximada por:

$$G_C(\mathrm{j}w) = \frac{\kappa \mathrm{e}^{-\mathrm{j}wt}}{\left(\frac{\mathrm{j}w}{a}+1\right)^{\alpha}\left(\frac{\mathrm{j}w}{b}+1\right)^{\beta}} \qquad \{\kappa, T, a, b, \alpha, \beta\} \in \mathbb{R}^{+} \tag{9.9}$$

levando a um sistema de fase mínima com atraso no tempo em contraste com os casos anteriores.

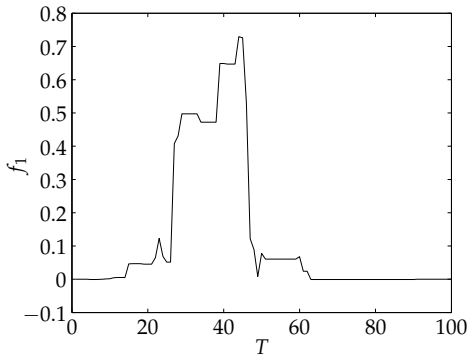


Figura 9.11. Variação da saída $\delta f(T)$ com a função de aptidão f_D e para o sinal de excitação durante $T_{\mathrm{exc}} = 2$ gerações com semente $i = 1$ ($\Delta p = 0{,}04$)

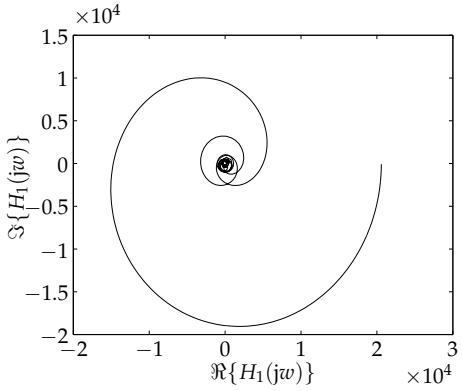
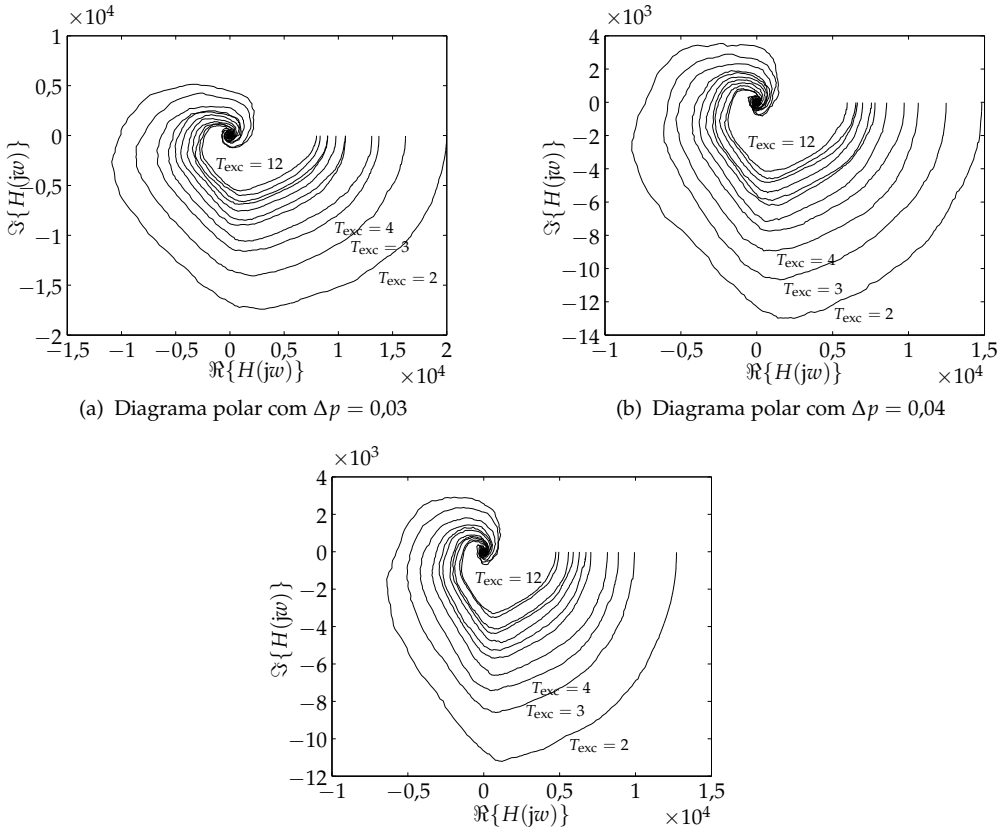


Figura 9.12. Função de transferência $H_1(\mathrm{j}w)$ com a função f_D usando a semente $i = 1$ ($T_{\mathrm{exc}} = 2$; $\Delta p = 0{,}04$)

Nesta simulação apenas o ganho κ (figura 9.14) segue claramente a equação (9.8), resultando os valores estimados encontrados na tabela 9.4. Os restantes parâmetros revelam um comportamento idêntico ao ruído branco não se tendo uma ideia clara para uma aproximação. No entanto, para simplificar a sua comparação com os casos anteriores decidiu-se manter a estimação dos parâmetros através da equação (9.8) obtendo-se exponenciais γ_2 próximas de zero.



(a) Diagrama polar com $\Delta p = 0{,}03$

(b) Diagrama polar com $\Delta p = 0{,}04$

(c) Diagrama polar com $\Delta p = 0{,}05$

Figura 9.13. Diagrama polar de $H(\mathrm{j}w)$ para a função $f_D(b)$ e $\Delta p = \{0{,}03; 0{,}04; 0{,}05\}$

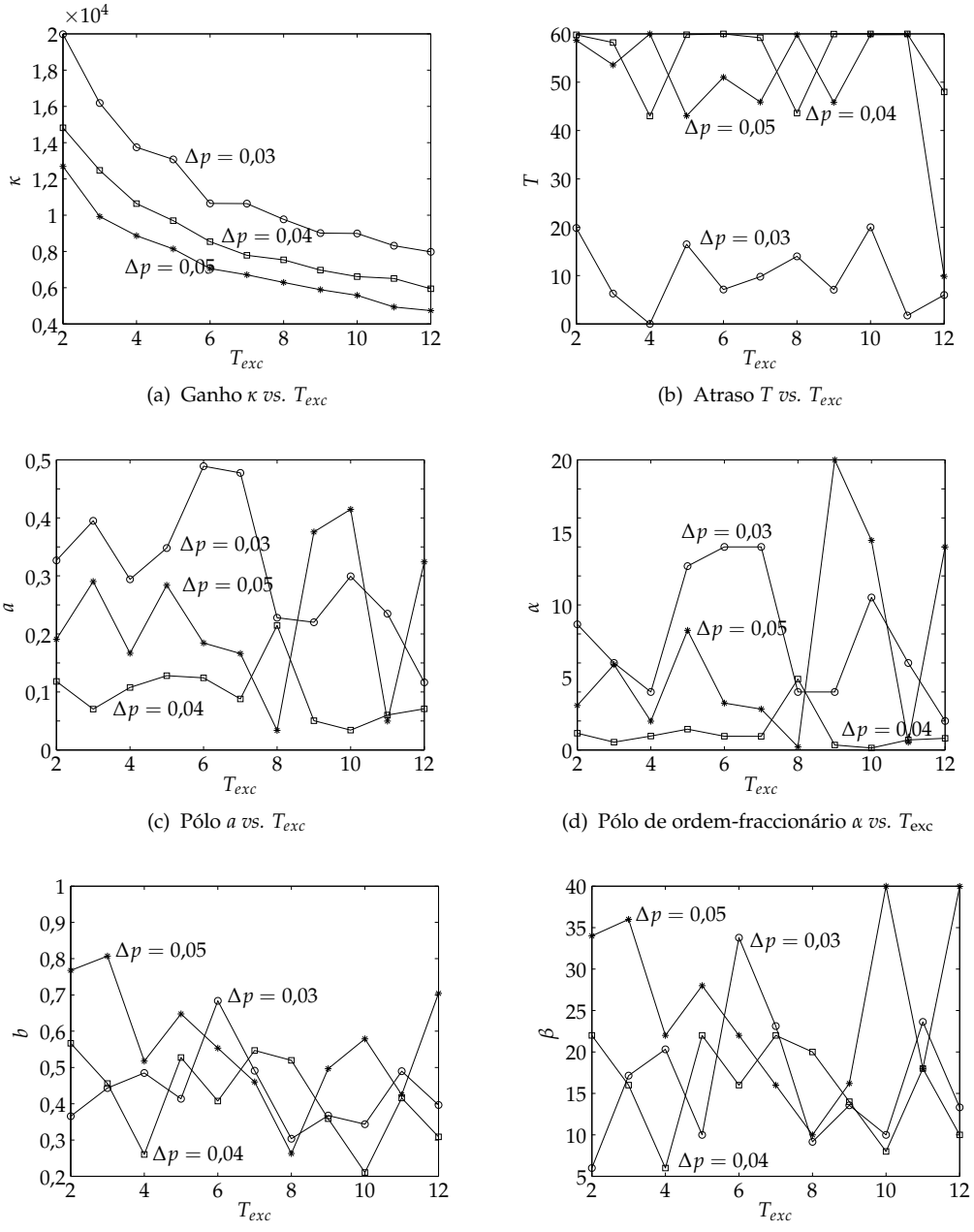
Tabela 9.4. Parâmetros γ_i, $i = \{1;2\}$ da aproximação de $\{\kappa, a, \alpha, b, \beta\}$ com a função de optimização f_D

(a) Parâmetros κ, T *vs.* Δp

	κ		T	
Δp	γ_1	γ_2	γ_1	γ_2
0,03	28328	−0,51	0,34	1,17
0,04	21467	−0,51	57,22	0,02
0,05	18418	−0,53	87,80	−0,36

(b) Parâmetros a, α, b, β *vs.* Δp

	a		α		b		β	
Δp	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2
0,03	0,58	−0,38	11,71	−0,32	0,45	−0,04	9,65	0,23
0,04	0,16	−0,35	1,50	−0,35	0,61	−0,23	19,72	−0,16
0,05	0,25	−0,18	34,51	−0,21	0,88	−0,27	3,09	−0,08



(a) Ganho κ *vs.* T_{exc}

(b) Atraso T *vs.* T_{exc}

(c) Pólo a *vs.* T_{exc}

(d) Pólo de ordem-fraccionário α *vs.* T_{exc}

(e) Pólo b *vs.* T_{exc}

(f) Pólo de ordem-fraccionário β *vs.* T_{exc}

Figura 9.14. Parâmetros estimados *vs.* T_{exc} com a função f_D

9.2.6 Análise dos resultados

Esta secção analisou a propagação e o fenómeno da dinâmica inerente à evolução dos indivíduos de uma população de um AG. O estudo foi concebido usando um AG simples com quatro funções de aptidão diferentes. Enquanto que o estudo dos AGs, usando esquemas, é bastante conhecido [29], a influência de sinais perturbadores sobre as condições de operação é pouco estudada.

A partir destes resultados pode-se concluir que as funções de optimização e a duração do tempo de perturbação tem uma influência assinalável sobre o AG enquanto que o valor adoptado para a probabilidade de mutação parece ser menos significativo. Adicionalmente, verifica-se que funções de aptidão "parecidas" revelam funções de transferência "parecidas".

9.3 Dinâmica de um planeador de trajectórias

9.3.1 Introdução

Nesta secção é estudada a dinâmica de um planeador genético com o fim de gerar uma trajectória para um manipulador robótico $2R$. O AG tem como objectivo minimizar o deslocamento da trajectória e a oscilação residual sem exceder um determinado binário. Com o fim de investigar o fenómeno envolvido no AG, como na secção anterior, a mutação é exposta a excitação de perturbações e a correspondente variação é calculada. O ruído caótico e os sinais de entrada/saida são estudados revelando uma dinâmica de ordem fraccionária, característicos de sistemas com termos de memória longos.

Nesta ordem de ideias, analisa-se a evolução dos sinais do sistema e a ordem fraccionária da dinâmica da população do AG com vista a determinar a trajectória para o manipulador robótico. Assim, a secção está organizada da seguinte forma: na

secção 9.3.2 introduz-se o problema a resolver, o AG usado na resolução e o resultado de uma experiência. Na secção 9.3.3 são apresentados alguns resultados para várias experiências envolvendo diferentes condições de excitação e é feito o estudo dos sinais resultantes e do fenómeno da dinâmica. Por último, na secção 9.3.4 são enunciadas as principais conclusões.

9.3.2 Planeador genético

Esta secção apresenta o esquema do AG adoptado para fornecer uma trajectória óptima, com uma oscilação residual reduzida no deslocamento e nas velocidades, sem ultrapassar os binários máximos impostos. Neste estudo é considerado um manipulador planar $2R$, que se pretende deslocar entre dois pontos do espaço de trabalho. O algoritmo usa a cinemática directa de modo a evitar os problemas singulares.

Representação da trajectória

O manipulador pode mover-se entre dois pontos do espaço de trabalho. Consequentemente, as configurações inicial e final são dadas pela cinemática inversa. A trajectória é codificada directamente, usando codificação real, num vector no espaço das juntas para ser usado pelo AG:

$$\left[\Delta t, (q_1^{(1,T)}, q_2^{(1,T)}), \ldots, (q_1^{(j,T)}, q_2^{(j,T)}), \ldots, (q_1^{(m-2,T)}, q_2^{(m-2,T)})\right] \tag{9.10}$$

A junta i para a posição intermédia j, na geração T, é $q_i^{(j,T)}$, o vector é constituído por $m-2$ configurações e cada configuração é formada pelos dois valores relativos às posições das juntas. Os valores, $q_i^{(j,0)}$, são inicializados no intervalo $]-\pi, +\pi]$. Como as configurações inicial e final se mantêm inalteradas durante a pesquisa do AG não são colocadas no vector. O parâmetro Δt é introduzido no vector para especificar o tempo entre duas configurações consecutivas.

Operadores genéticos

A população inicial é gerada aleatoriamente. A pesquisa é então efectuada através dessa população. Os três operadores usados no planeador genético são: reprodução, cruzamento e mutação, descritos de seguida. No que diz respeito ao operador de reprodução é baseado no seu valor de aptidão. Neste caso, é usada uma selecção por torneio-5 de modo a escolher as soluções que darão origem a dois descendentes. Para determinar quais os elementos da população que estes descendentes substituem é usado um outro torneio de 5 elementos para escolher as trajectórias menos aptas. Para o operador de cruzamento, os elementos são agrupados em pares. De seguida o operador de cruzamento de ponto simples é executado sobre cada par. O ponto de cruzamento só pode ocorrer entre configurações, (*i.e.*, o operador de cruzamento não pode "cortar" uma configuração ao meio). O operador de mutação pode modificar vários parâmetros, nomeadamente o tempo entre duas configurações, Δt, e os ângulos das juntas. Assim, o operador de cruzamento substitui um valor da junta, $q_i^{(j,T)}$ com uma probabilidade de mutação p_{m}. O novo valor $q_i^{(j,T+1)}$ é obtido através da expressão $q_i^{(j,T+1)} = q_i^{(j,T)} + N(0; 1/\sqrt{2\pi})$, onde N é a função de distribuição normal.

Critério de avaliação

Para avaliar o desempenho das trajectórias são usados cinco índices. Todos estes índices são inseridos na função de aptidão como funções a minimizar. Cada índice é calculado separadamente e, de seguida, é integrado na função de aptidão. A função de aptidão f, adoptada para determinar o desempenho de cada trajectória, é definida como:

$$f = \beta_1 f_q + \beta_2 f_{\dot{q}} + \beta_3 f_p + \beta_4 f_{\dot{p}} + \beta_5 f_{\mathrm{sc}} \tag{9.11}$$

onde os índices f_q, $f_{\dot{q}}$, f_p, $f_{\dot{p}}$ e f_{sc} são formulados de seguida. A optimização consiste em encontrar um conjunto de parâmetros que minimiza f de acordo com as prioridades indicadas pelos factores de pesos β_i ($i = 1, .., 5$).

O critério f_q, é usado para minimizar a distância angular percorrida pelo manipulador robótico dando origem à seguinte fórmula:

$$f_q = \sum_{j=1}^{m-1} \sum_{i=1}^{2} \left(\dot{q}_i^j \right)^2 \tag{9.12}$$

A função $f_{\dot{q}}$ é adoptada para minimizar a oscilação residual da velocidade do manipulador através do critério:

$$f_{\dot{q}} = \sum_{j=1}^{m} \sum_{i=1}^{2} \left(\ddot{q}_i^j \right)^2 \tag{9.13}$$

O critério f_p é introduzido na função de aptidão f para minimizar o comprimento total da trajectória. Este índice é definido como:

$$f_p = \sum_{j=1}^{m-1} d\left(p^j,\ p^{j-1} \right)^2 \tag{9.14}$$

onde p^w indica a posição cartesiana w do órgão terminal e $d(.,.)$ é a função que mede a distância entre os dois argumentos.

O índice $f_{\dot{p}}$ na função de aptidão é responsável pela redução da oscilação residual da velocidade cartesiana do órgão terminal. Este critério é formulado da seguinte maneira:

$$f_{\dot{p}} = \sum_{j=2}^{m-1} \left| d\left(p^j,\ p^{j-1} \right) - d\left(p^{j-1},\ p^{j-2} \right) \right|^2 \tag{9.15}$$

O índice f_{sc} representa a actuação em excesso relativamente ao binário máximo

$\tau_{i\ \max}$, que o motor i consegue fornecer para a trajectória em consideração.

$$f_{sc} = \sum_{j=0}^{m-1} \left(f_1^j + f_2^j \right) \tag{9.16a}$$

$$f_i^j = \begin{cases} 0, & \text{se } \left|\tau_i^j\right| < \tau_{i\ \max} \\ \left|\tau_i^j\right| - \tau_{i\ \max}, & \text{outros casos} \end{cases} \tag{9.16b}$$

As equações da dinâmica para um manipulador de dois elos (figura 9.15) pode ser facilmente obtido pelo lagrangeano [134] levando às seguintes equações:

$$\tau_1 = d_1\ddot{q}_1 + d_c\ddot{q}_2 - c\dot{q}_2^2 - 2c\dot{q}_1\dot{q}_2 + g_1 \tag{9.17a}$$

$$\tau_2 = d_c\ddot{q}_1 + d_2\ddot{q}_2 + c\dot{q}_1^2 + g_2 \tag{9.17b}$$

$$d_1 = m_1 l_1^2 + m_2\left[l_1^2 + l_2^2 + 2l_1 l_2 \cos\left(q_2\right)\right] \tag{9.17c}$$

$$d_2 = m_2 l_2^2 \tag{9.17d}$$

$$d_c = m_2\left[l_2^2 + 2l_1 l_2 \cos\left(q_2\right)\right] \tag{9.17e}$$

$$c = m_2 l_1 l_2 \sin\left(q_2\right) \tag{9.17f}$$

$$g_1 = g\left(m_1 + m_2\right) l_1 \cos\left(q_1\right) + g_2 \tag{9.17g}$$

$$g_2 = g m_2 l_2 \cos\left(q_1 + q_2\right) \tag{9.17h}$$

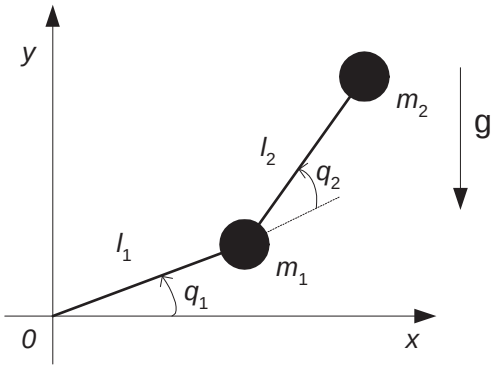


Figura 9.15. Manipulador de dois eixos rotacionais (2R)

Resultados da simulação

Esta secção descreve uma experiência simples que consiste em mover o manipulador robótico desde o ponto inicial $A \equiv \{1{,}25; -0{,}30\}$ até ao ponto final $B \equiv \{-0{,}50; 1{,}40\}$. O AG adopta uma probabilidade de cruzamento e de mutação, respectivamente de $p_{\mathrm{c}} = 0{,}8$ e de $p_{\mathrm{m}} = 0{,}05$ por lugar, uma população de 200 elementos para as configurações intermédias, $m = 8$ e um selecção por torneio-5. O comprimento e o peso de cada elo do robô, é respectivamente de $l_i = 1$ m e de $m_i = 1$ kg ($i = 1; 2$). Adicionalmente, as juntas 1 e 2 são livres de rodar em torno dos seus eixos e o binário máximo permitido é, respectivamente de $\tau_{1\ max} = 16$ Nm e de $\tau_{2\ max} = 5$ Nm.

Neste estudo existem duas variáveis distintas nomeadamente: o tempo da trajectória t, durante o qual o manipulador se move, e o tempo de evolução T, que corresponde às gerações sucessivas do AG. No primeiro caso, o tempo entre duas configurações sucessivas é restringido ao intervalo $0{,}05 \leq \Delta t \leq 1{,}60$ s. No segundo caso, o tempo normalizado entre duas gerações consecutivas do AG é considerado $T = 1$ s.

As figuras 9.16–9.19 apresentam: a trajectória no plano $\{x, y\}$, o deslocamento, a velocidade e o binário do manipulador nas juntas. A figura 9.20 contém os percentis para P_r, $r = \{0; 30; 70; 100\}\%$, da aptidão da população do AG ao longo da evolução.

A trajectória apresenta um comportamento suave, tanto nos deslocamentos como nas velocidades e o binário máximo requerido não excede os limites impostos.

9.3.3 Evolução e dinâmica de ordem fraccionária

Esta secção estuda a dinâmica envolvida na propagação do sinal na população do AG. Nesta perspectiva, perturbações de amplitude pequena são injectadas nos sinais do sistema e a sua influência na população é avaliada.

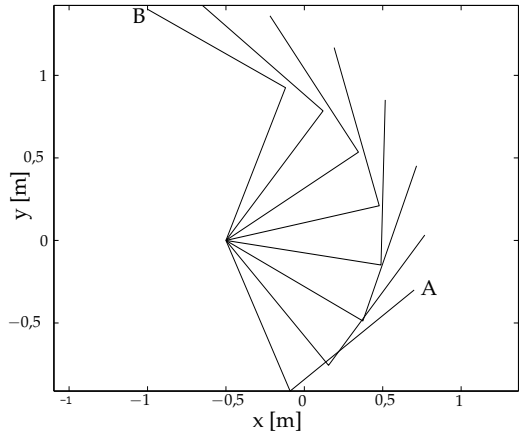


Figura 9.16. Trajectória robótica no plano $\{x, y\}$

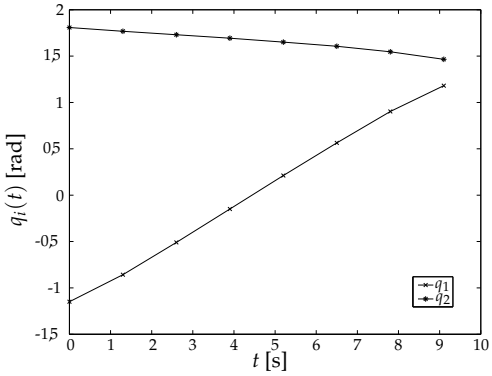


Figura 9.17. Posição das juntas *versus* tempo t

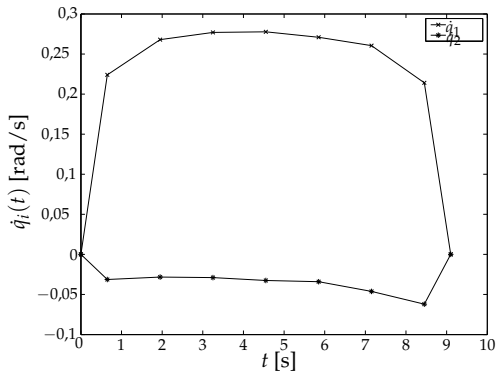


Figura 9.18. Velocidade das juntas *versus* tempo t

Simulações

Nesta secção o sistema AG é estimulado perturbando a probabilidade de mutação através de um sinal de ruído branco e a variação dos percentis da aptidão da população correspondente é estudada. Assim, a variação do sinal da probabilidade de mutação e a variação dos percentis da aptidão resultante da população genética, durante a evolução, podem ser vistos, respectivamente como os sinais de entrada e de saída em função do tempo (figura 9.21).

As figuras 9.22 e 9.23 mostram o sinal de entrada δp_m, nos domínios do tempo (das gerações) e da frequência, para perturbações na probabilidade de mutação de $\Delta p = 0{,}12 p_{\mathrm{m}}$ e para um período de excitação de $T_{\mathrm{exc}} = 100$ gerações. As figuras 9.24 e

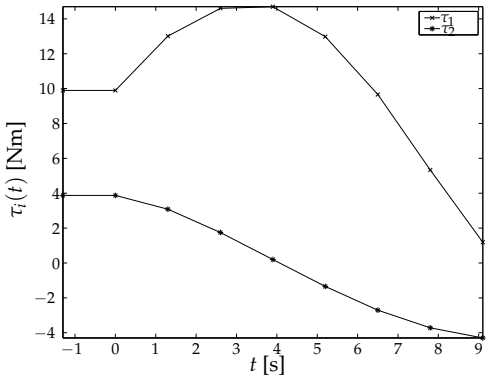


Figura 9.19. Binário *vs*. tempo t

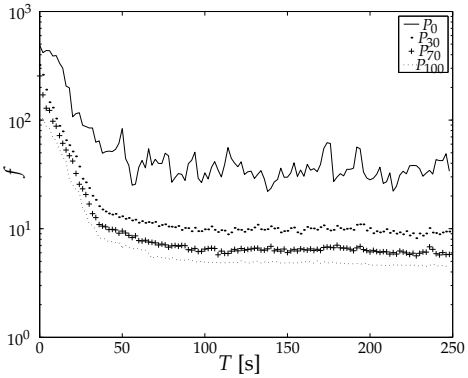
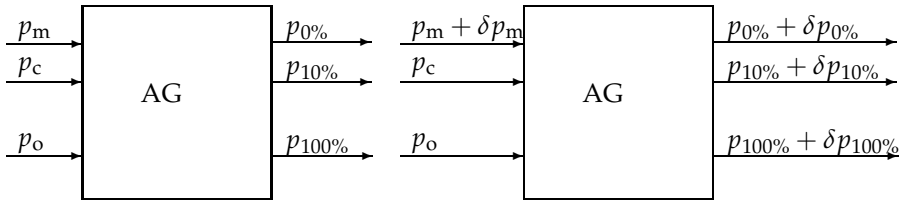


Figura 9.20. Percentis da aptidão da população *vs.* geração T



(a) Sistema inicial (b) Sistema perturbado

Figura 9.21. Perturbação do sistema através de um sinal de ruído branco

9.25 apresentam a variações correspondente do sinal de saída δP_r para o percentil de $r = 50\%$ da função de aptidão. A função de transferência $H_r(j\omega)$, entre os sinais de entrada e de saída, e a aproximação analítica de ordem fraccionária $G_r(j\omega)$ é descrita na figura 9.26.

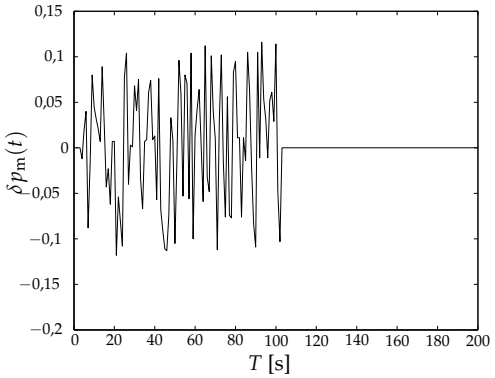


Figura 9.22. Perturbação de entrada $\delta p_\mathrm{m}(T)$ injectada na probabilidade de mutação durante $T_\mathrm{exc} = 100$ gerações

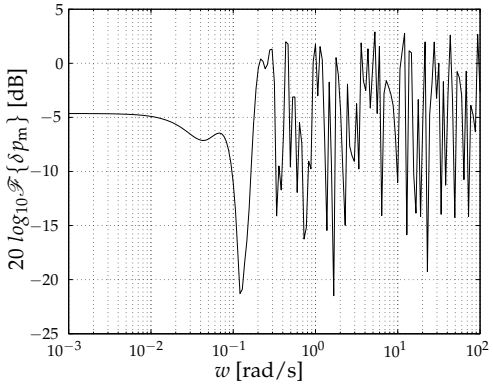


Figura 9.23. Espectro de Fourier $\mathscr{F}\{\delta p_\mathrm{m}(T)\}$ da variação da probabilidade de mutação

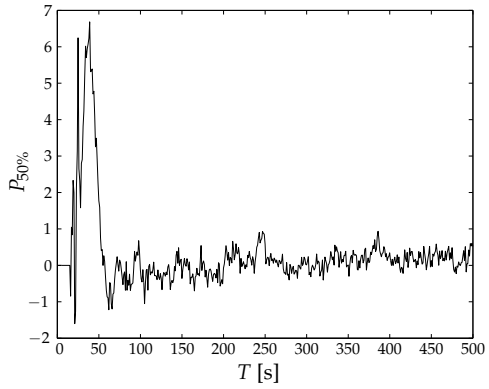


Figura 9.24. Variação do percentil de saída $\delta P_{50}(T)$ para a excitação do sinal de entrada de $T_{\text{exc}} = 100$ gerações

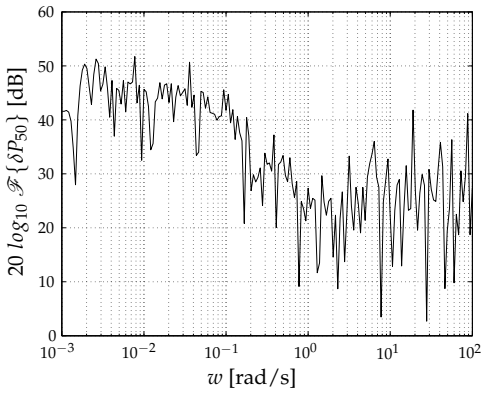


Figura 9.25. Espectro de Fourier $\mathscr{F}\{\delta P_{50}(T)\}$ da função de aptidão do percentil $r = 50\%$

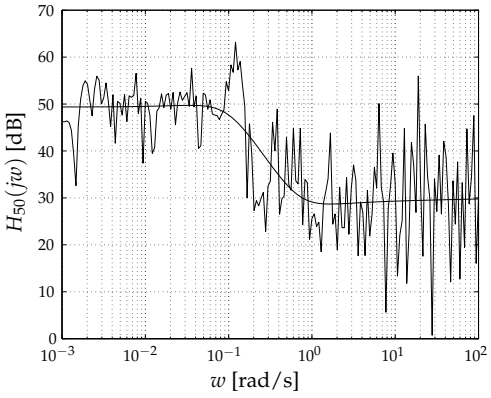


Figura 9.26. Função de transferência $H_{50}(j\omega) = \mathscr{F}\{\delta P_{50}(T)\}/\mathscr{F}\{\delta p_{\text{m}}(T)\}$ e a aproximação analítica $G_{50}(j\omega)$ para o percentil $r = 50\%$

Identificação

Nesta secção, os dados numéricos das funções de transferência do sistema são aproximados por expressões analíticas com ganho $k \in \mathbb{R}^+$, um zero e um pólo $(a, b) \in \mathbb{R}^+$, respectivamente, dados por:

$$G_r(s) = k\frac{\left(\frac{s}{a}\right)^{\alpha} + 1}{\left(\frac{s}{b}\right)^{\beta} + 1} \tag{9.18}$$

O AG utilizado para identificar a função analítica é idêntico ao da secção 9.2. No entanto, a função de identificação $f_{r,ide}$ mede, logaritmicamente, a distância entre a

função de transferência numérica H_r e a analítica G_r (9.19) devido à estimação ser levada a cabo numa escala logarítmica.

$$f_{r,ide} = \sum_{i=1}^{nf} \left[\log_{10} \frac{H_r(\omega_i)}{G_r(\omega_i)} \right]^2 \tag{9.19}$$

$$G_r(\omega_i) = k \left\{ \frac{\left[\left(\frac{\omega_i}{a}\right)^{\alpha} \cos\left(\frac{\pi}{2}\alpha\right) + 1 \right]^2 + \left[\left(\frac{\omega_i}{a}\right)^{\alpha} \sin\left(\frac{\pi}{2}\alpha\right) \right]^2}{\left[\left(\frac{\omega_i}{b}\right)^{\beta} \cos\left(\frac{\pi}{2}\beta\right) + 1 \right]^2 + \left[\left(\frac{\omega_i}{b}\right)^{\beta} \sin\left(\frac{\pi}{2}\beta\right) \right]^2} \right\}^{1/2} \tag{9.20}$$

Com o fim de obter os parâmetros da expressão (9.18) são executadas 21 identificações através de um AG e os resultados medianos são escolhidos como os parâmetros de estimação final. Adicionalmente, para estudar a influência do período de excitação T_{exc} são realizadas várias simulações desde $T_{\text{exc}} = 20$ até $T_{\text{exc}} = 1000$ gerações. A relação entre os parâmetros da função de transferência e (T_{exc}, P_r) encontram-se nas figuras 9.27–9.31.

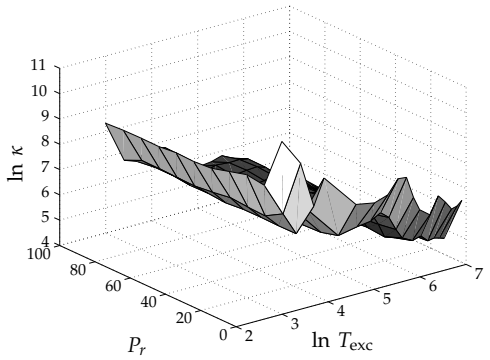


Figura 9.27. Ganho estimado $\ln(\kappa)$ *vs.* (T_{exc}, P_r)

Os gráficos de $\{\kappa, a, b, \alpha, \beta\}$ podem ser aproximados através da técnica dos mínimos quadrados levando as equações:

$$\kappa = 62069\ {T_{\text{exc}}}^{0,852} \mathrm{e}^{-0,011\ P_r} \tag{9.21a}$$

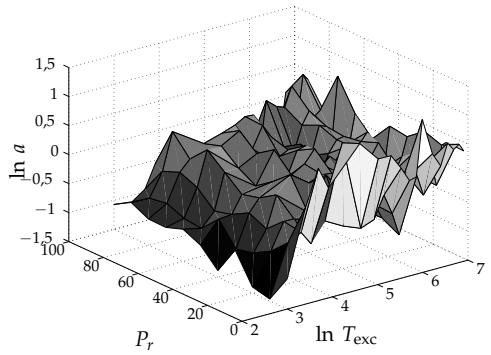


Figura 9.28. Zero estimado $\ln(a)$ *versus* $(T_{\text{exc}},\, P_r)$

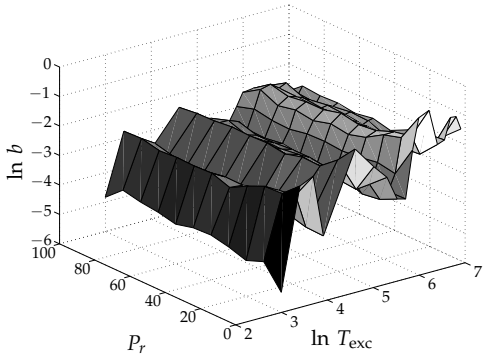


Figura 9.29. Pólo estimado $\ln(b)$ *versus* $(T_{\text{exc}},\, P_r)$

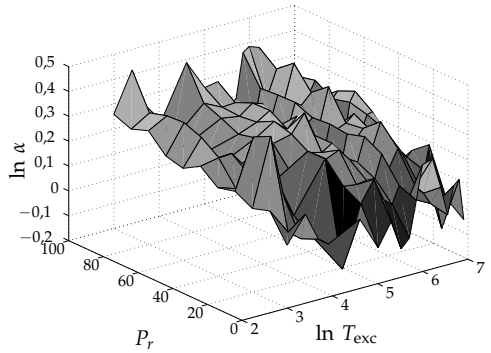


Figura 9.30. Pólo estimado de ordem fracionária $\ln(\alpha)$ *versus* $(T_{\text{exc}},\, P_r)$

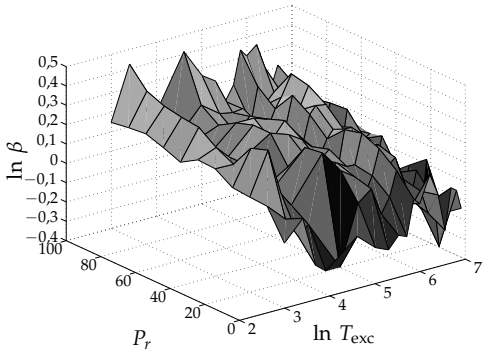


Figura 9.31. Pólo estimado de ordem fracionária $\ln(\beta)$ *versus* $(T_{\text{exc}},\, P_r)$

$$a = 0{,}317\ T_{exc}^{0{,}204} \mathrm{e}^{-0{,}0044\ P_r} \tag{9.21b}$$

$$b = 0{,}012\ {T_{\text{exc}}}^{0{,}362} \mathrm{e}^{-0{,}0060\ P_r} \tag{9.21c}$$

$$\alpha = 1{,}121\ {T_{\text{exc}}}^{-0{,}005} \mathrm{e}^{0{,}0019\ P_r} \tag{9.21d}$$

$$\beta = 1{,}093\ {T_{\text{exc}}}^{-0{,}028} \mathrm{e}^{0{,}0025\ P_r} \tag{9.21e}$$

9.3.4 Análise dos resultados

Os resultados obtidos revelam que os parâmetros da função de transferência têm uma dependência pequena relativamente aos percentis P_r da função de aptidão e, consequentemente, a escolha de um valor para r não tem qualquer importância para o estudo em questão. Por outro lado, o período de excitação T_{exc} tem uma influência maior na variação dos parâmetros.

Permitindo que a ordem do pólo/zero varie livremente, são obtidos valores não-inteiros para α e para β, ao passo que a escolha de valores inteiros para a função de transferência levaria a um número grande de pólos e zeros de modo a obter uma boa aproximação na estimação analítica dos valores numéricos. O "requisito" de modelos de ordem fraccionária, em oposição aos modelos clássicos de ordem inteira, é uma discussão bem conhecida e mesmo hoje em dia não existem conclusões claras, uma vez que é sempre possível aproximar a resposta em frequência fraccionária por uma de ordem inteira com um número grande de pólos e de zeros. Contudo, nas experiências presentes existe um ponto de vista complementar na direcção do cálculo fraccionário. De facto, analisando o sinal de saída, observa-se um sinal de ruído branco com semelhanças a sinais que aparecem em sistemas naturais, que é auto sustentado, mesmo para períodos de tempo afastados do período de excitação através do ruído branco. Esta característica é típica de sistemas caóticos e sugere uma investigação futura na dinâmica que resulta na perturbação de outros sinais de entrada, isto é, para outras variáveis do AG e outro sinal de perturbação com diferente espectro.

9.4 Conclusões

Neste capítulo, modelaram-se vários sistemas AGs e analisou-se a sua dinâmica. As entradas consideradas são os parâmetros de sintonia dos AGs, nomeadamente as probabilidades de mutação, cruzamento, entre outras. Devido à complexidade do

problema considerou-se o vector da probabilidade de mutação como sinal variável, mantendo os restantes vectores dos sinais de entrada fixos ao longo das experiências.

Concluiu-se que a dinâmica dos AGs é adequadamente modelada por funções de transferência de ordem-fraccionária. Este capítulo estudou a dinâmica de um AG simples e de um planeador de trajectórias:

- No AG simples a modelação foi efectuada utilizando o método dos mínimos quadrados, no domínio complexo. Depois de modelado foi analisado o seu comportamento no domínio das frequências, particularmente no que se refere a estabilidade do mesmo. Considerou-a introdução do sinal perturbador na probabilidade de mutação durante um certo período evolutivo inicial variável. Concluiu-se que para certos tipos de AGs, a exposição do sinal de mutação ao ruído durante longos períodos pode alterar significativamente o comportamento dinâmico do sistema.
- No planeador genético, o estudo foi desenvolvido baseado num AG para o planeamento de uma trajectória de um manipulador robótico. Para o caso em estudo a evolução do sinal tem semelhanças com os revelados pelos sistemas caóticos, o que confirma a necessidade de uma ferramenta matemática adequada ao fenómeno sob investigação onde o cálculo fraccionário encaixa perfeitamente.

10 Conclusões

10.1 Introdução

Neste capítulo é feita uma síntese do trabalho desenvolvido ao longo da tese e são formuladas as principais conclusões. São também realçados alguns aspectos que poderão merecer uma investigação mais profunda.

Nesta ordem de ideias, na secção 10.2 são apontados os aspectos principais do trabalho e, na secção 10.3 são referidas algumas perspectivas para desenvolvimento futuro.

10.2 Síntese conclusiva do trabalho realizado

Nesta dissertação:

- Foram revistos os tópicos principais sobre algoritmos evolutivos, especialmente sobre algoritmos genéticos. De seguida foram apresentados alguns algoritmos multi-objectivo e um conjunto de índices de desempenho que permitem medir a sua eficiência;

- Foi apresentado o problema do planeamento de trajectórias para manipuladores robóticos e foi descrito um resumo de aplicações de sistemas robóticos que utilizam algoritmos evolutivos, com particular incidência nos AGs. Estas aplicações incidem essencialmente na geração de trajectórias para robôs móveis, no planeamento de trajectórias para manipuladores robóticos, na selecção e síntese do robô que melhor se adapte a determinadas tarefas, na locomoção de robôs e na estimação de parâmetros de robôs;

- Foram apresentados os resultados do planeamento de trajectórias para manipuladores robóticos, do tipo $2R$, em tempo real. Para esse fim, foi desenvolvido, numa primeira fase um programa baseado em AGs que calcula todas as trajectórias possíveis. Posteriormente, quando solicitado pelo utilizador, fornece as trajectórias entre duas posições;

- Foi desenvolvido um AG para gerar a estrutura mecânica que melhor se adeqúe à execução de determinadas trajectórias. O problema é resolvido tendo em conta o ponto de vista mono e multi objectivo;

- Foi apresentada uma aplicação que optimiza trajectórias, para manipuladores robóticos, usando simultaneamente vários objectivos fornecendo um conjunto alternativo de resultados;

- Foi proposto um novo algoritmo com o intuito de promover a diversidade das soluções no espaço dos objectivos, bem como um conjunto de técnicas que permitem medir o desempenho dos algoritmos multi-objectivo;

- Na teoria dos AGs, foi estudada a dinâmica existente num AG onde se obtiveram resultados promissores. A dinâmica do sistema foi aproximada por funções de transferência de ordem fraccionária.

Os resultados obtidos demonstram, claramente, que o planeamento de trajectórias através de AGs é viável e que a complexidade do algoritmo é mantida, independentemente da existência, ou não, de obstáculos, ou do número destes. Conclui-se que

os AGs têm uma enorme capacidade em aplicações deste tipo.

Os resultados obtidos, no planeamento de trajectórias, podem ser melhorados com a introdução de novas técnicas que aumentem o desempenho dos algoritmos genéticos, como os que foram apresentados nesta tese. Por outro lado, o conhecimento da dinâmica dos AGs pode ser útil na formulação e resolução dos algoritmos propostos.

10.3 Perspectivas para desenvolvimentos futuros

Com base no trabalho desenvolvido surgiram os seguintes aspectos que podem ser alvo de uma possível investigação no futuro:

- Planeamento de trajectórias no espaço das juntas (em alternativa ao espaço operacional) utilizando a cinemática inversa e tendo em atenção as singularidades;
- Estudo de novas funções de aptidão nomeadamente com base na manipulabilidade do manipulador robótico;
- Aplicação dos AGs em sistema com vários braços cooperantes.
- Modificação da função de aptidão de forma a diminuir o tempo de cálculo com vista à sua aplicação em sistemas de tempo real;
- Aplicação de outros tipos de AEs no planeamento de trajectórias para manipuladores robóticos de forma a encontrar o que melhor se adapte a este tipo de problema;
- No operador de selecção, ter em atenção as diferentes soluções que se sobrepõem no espaço dos objectivos, de modo a escolher as mais aptas do ponto de vista do espaço dos parâmetros;

- Comparar o algoritmo MaxMin com outro tipos de algoritmos, nomeadamente o SPEA, SPEA-II, $\epsilon-$SPEA, $\epsilon-$MOEA entre outros;
- Estudar a dinâmica dos AGs, alterando outros parâmetros tais como a probabilidade de cruzamento, comprimentos do vector, taxa de substituição dos vectores, dimensão da população, o número de parâmetros, de objectivos, com técnicas de nicho entre outras.

Em conclusão, pode afirmar-se que existe um largo espectro de investigação que, no futuro poderá trazer novos formalismos e novos conceitos, permitir aplicações mais sofisticadas com desempenho ainda mais relevante.

A Técnicas de pesquisa

A.1 Técnicas de pesquisa

Neste apêndice são descritas sucintamente algumas técnicas de pesquisa. Na figura 2.1 está ilustrada um modo de classificar as técnicas de pesquisa. No entanto, estas podem ser agrupadas de acordo com um vasto número de características [141, 142] nomeadamente: se usam população; se usam memória; o número de vizinhos; a medida de desempenho da solução ser estática ou dinâmica; o tipo de inspiração; se segue uma trajectória ou se executa saltos durante a pesquisa; tipo de nicho que usam; regras determinística *vs.* regras estocásticas.

Métodos Indirectos: os métodos indirectos baseiam-se na obtenção analítica de valores nulos das derivada. Normalmente são usados para funções suaves (não descontínuas) e onde o número de zeros das derivadas das funções é bastante pequeno.

Técnica Exaustivas: em princípio, pesquisam todos os pontos possíveis, um de cada vez. Esta técnica normalmente é simples de implementar mas o número de pontos possíveis pode ser muito grande para a pesquisa directa. Em alguns

casos, *e.g.*, teoria dos jogos, é possível reduzir a pesquisa de modo que as regiões que não contenham a solução não sejam examinados.

Técnicas de Cálculo Directo: tais como Fibonacci ou Newton usam valores de funções ou de gradiente para estimar a localização de um extremo próximo. Estas técnicas, e outras, são conhecidas por técnicas de Hill-climbing [143] porque estimam onde um máximo se encontra, movendo-se para esse ponto, faz uma nova estimativa, e assim sucessivamente até alcançar o topo. Estas técnicas podem ser usadas em problemas "bem comportados" ou problemas que se podem transformar em problemas bem comportados.

Em Profundidade Primeiro: (*Depth-First Search*) este tipo de técnica [144] é usado quando a informação a pesquisar é guardada em árvores. A pesquisa é efectuada dando sempre prioridade aos ramos de um lado, em profundidade.

Em Largura Primeiro: (*Width-First Search*) este tipo de técnica é usado quando a informação a pesquisar é guardada em árvores. A pesquisa é efectuada de acordo com o nível dos nós da árvore, em largura.

Programação Dinâmica: funciona partindo do princípio que encontra a solução global seleccionando um ponto intermédio que incida entre o ponto actual e o ponto de chegada [144]. Este processo é recursivo, uma vez que os pontos intermédios seguintes são determinados em função dos pontos previamente seleccionados.

Ramifica e Limita: (*Branch and Bound*) este método baseia-se na sucessiva partição do espaço de pesquisa. Inicialmente é determinada um custo limite para o problema. O passo seguinte é tentar determinar uma solução com custo inferior ao actual.

Algoritmo *Greedy*: a solução do problema é construída a partir de uma série de passos. A solução é construída considerando que o melhor deslocamento em cada passo leva a solução óptima (o que nem sempre acontece).

Algoritmo A*: O algoritmo A* é baseado nos métodos *greedy* mas usa uma função de desempenho com informação adequada de modo a evitar óptimos que não interessam.

Algoritmos Evolutivos: baseados na teoria da selecção natural Darwiana evolutiva [29], onde a selecção é progressivamente melhorada pela remoção selectiva dos piores elementos e pela criação de novos elementos a partir do melhores elementos. Enquanto que os algoritmos de pesquisa podem ser bem adaptados para um problema em particular, o teorema *no-free-lunch* mostra que em média para todos os problemas os métodos são todos equivalentes, *i.e.*, se têm um bom desempenho para um problema em particular, existem outros problemas onde o seu desempenho é bastante baixo. Por outras palavras, podem existir problemas onde o método de pesquisa tem um desempenho tão mau como a pesquisa aleatória.

Estratégias de Evolução: Este tipo de algoritmo evolutivo foi desenvolvido por I. Recheberg e H. P. Schwefel na universidade técnica de Berlim na década de 1960. A Estratégia de evolução utiliza tipicamente parâmetros de valor real, contudo também tem sido aplicados em problemas discretos. A característica básica é: a distinção entre a população de progenitores (de μ elementos) e a população de descendentes ($\lambda \geq \mu$ elementos); o uso de mutações através de funções com distribuição normal; o uso de tipos de cruzamento próprios; e a incorporação da auto-adaptação para os parâmetros estratégicos. Os parâmetros estratégicos, que descrevem a função de densidade de probabilidades do operador de mutação, são "envolvidos" ao longo das gerações, pelo menos princípios que as variáveis objectivo.

Algoritmo Genético: um algoritmo evolutivo desenvolvido por J. H. Holland e pelos seus alunos em Ann Arbor, MIT, na década de 1960 [1]. Procedimentos equivalentes foram mais cedo apresentados por H. J. Bremermann na UC Berkeley e A. S. Fraser na universidade de Canberra, Austrália nas décadas de

1960 e 1950, respectivamente. Originalmente, os algoritmos genéticos foram desenhados para sistemas formais com fins adaptativos em vez de sistemas de optimização. As características básicas são uma forte ênfase na recombinação (cruzamento), uso de um operador estocástico na selecção (selecção proporcional), e a interpretação da mutação com um papel menor no decorrer do algoritmo. Na versão original do algoritmo genético são usadas representações vectoriais binárias.

Programação Genética: é um tipo de algoritmo derivado dos algoritmos genéticos desenvolvidos por John Koza [19, 30]. A programação genética caracteriza uma classe de algoritmos evolutivos com atenção à geração automática de programas de computador. Para evidenciar este facto, cada indivíduo da população representa um programa completo de computador numa linguagem de programação adequada. Geralmente, as expressões simbólicas representam árvores de análise gramatical e normalmente são implementadas na linguagem LISP.

Sistemas de Classificação: este tipo de sistemas são baseados em regras de aprendizagem por exemplos e por indução [29]. Os sistemas de classificação envolvem uma população que produz regras (também denominada por aproximação de Michigan, onde um indivíduo corresponde a uma regra simples) ou uma população que produz regras básicas (também denominada por aproximação de Pittsburgh, onde um indivíduo representa uma regra básica completa) por meio de um algoritmo evolutivo. As regras são normalmente codificadas através de um alfabeto ternário, que inclui um símbolo "*don't care*" facilitando a capacidade de generalização de condições ou partes de acção de uma regra, permitindo assim uma aprendizagem indutiva de conceitos. Na aproximação de Michigan, a regra de aptidão é actualizada de modo incremental em cada geração pelo algoritmo de afectação por um critério baseado na recompensa que o sistema obtém do ambiente. Enquanto que a aproximação Pittsburgh, a aptidão de uma regra base completa é calculada pelo teste do comportamento

do indivíduo no seu ambiente.

Programação Evolutiva: Um algoritmo evolutivo desenvolvido por L. J. Fogel em San Diego, CA-USA, na década de 1960 e depois redefinido por D. B. Fogel e outros investigadores na década de 1990. A programação evolutiva foi originalmente desenvolvida como um método para envolver máquinas de estado finito com o fim de resolver tarefas de predição de série temporal e posteriormente foi extendida para problemas de optimização. A programação evolutiva incide tipicamente na variação de um progenitor através de operadores criados para o problema em questão. Contudo, as versões mais recentes da programação evolutiva consideram a possibilidade da recombinação de três ou mais progenitores. A selecção usada é a selecção por torneio estocástica que determina os μ progenitores e os λ descendentes gerados pela mutação. A programação evolutiva permite a auto-adaptação dos parâmetros estratégicos durante a pesquisa.

Colónia de Formigas: A optimização por colónia de formigas [145] é um sistema multi-agente onde iterações simples entre formigas artificiais resultam num comportamento complexo da colónia. Este tipo de algoritmos foram inspirados em colónias de formigas reais que depositam uma substância química (feromona) no chão. Estas substâncias influenciam o comportamento individual das formigas. Quanto maior for a quantidade de feromona existente num caminho maior é a probabilidade de uma formiga o escolher.

Redes Neuronais: As redes neuronais artificiais tentam implementar num computador a capacidade de processamento de dados do cérebro. Para implementar este sistema (pelo menos num modo bastante simplificado tendo em atenção o número de unidades e a sua interligação), unidades simples (correspondem a neurónios) são organizados num número de camadas e a comunicação é realizada através de ligações ponderadas (correspondem a *axónios* e *dendrites*). A rede funciona em paralelo, cada unidade da rede calcula tipicamente uma

soma ponderada de entradas, executa alguns mapas internos de resultados, e eventualmente propaga um vector diferente de zero para a sua ligação de saída.

***Simulated Annealing*:** É uma estratégia de optimização inspirada num modelo de evolução termodinâmica, modelando o processo de aquecimento e arrefecimento lento com vista a encontrar um estado de energia mínimo. A estratégia funciona com uma possível solução e gera soluções através de um operador de variação (mutação). A solução criada é aceite sempre que apresente uma diminuição da energia ou então aceite com uma probabilidade pequena se apresentar um valor de energia superior. O parâmetro de controlo (probabilidade pequena) é normalmente denominado por temperatura.

Algoritmo Cultural: Os algoritmos culturais [146] têm uma implementação semelhante aos algoritmos evolutivos que suportam os modelos de herança. Um é implementada ao nível da micro-evolução em termos de características e o outro ao nível da macro-evolução em termos de crenças. Os dois modelos interagem por um canal de comunicação que permite que o comportamento dos indivíduos altere a estrutura das crenças e que a estrutura das crenças restringem o modo como os indivíduos se comportam. A estrutura das crenças representa o conhecimento "cultural" relativos a certos problemas e consequentemente ajuda na solução de problemas ao nível das características.

Optimização de bandos de partículas (PSO): Este tipo de algoritmo, proposto por Kennedy e Eberhart [147], é baseado no comportamento social de bando de aves ou cardumes em movimento coordenado e sincronizado tendo em vista um objectivo como a forma de procurar alimentação ou como mecanismo de autodefesa. Cada indivíduo desloca-se com uma velocidade ajustada de acordo com a sua experiência e com a da população restante.

Técnicas de Pesquisa Estocásticas: usam a pesquisa para conduzir a escolha probabilística dos próximos pontos a pesquisar. Elas são gerais no seu domínio,

sendo capazes de resolver problemas bastante complexos que estão para além das capacidades tanto das técnicas exaustivas como das de cálculo.

Aprendizagem Incremental Baseada na População (PBIL): Captura explicitamente a dependência de primeira ordem entre os parâmetros de uma solução individual e a qualidade da solução [143]. O algoritmo PBIL usa um vector de probabilidades de primeira ordem para modelar uma distribuição de probabilidades simples sobre todo o conjunto de vectores. O vector de probabilidades é ajustado para aumentar a verosimilhança de gerar soluções de grande qualidade. Para este efeito, ao longo da pesquisa, o vector de probabilidades é influenciado no sentido da solução com melhor desempenho em cada geração.

BOA: (*Bayesian optimization algorithm*) foi desenvolvido por Pelikan, Goldberg, e Cantú-Paz [148] baseado nos conceitos dos AGs. O algoritmo usa técnicas para modelar os dados através de redes bayesianas para estimar a distribuição das soluções promissoras. As novas soluções são geradas usando a distribuição bayesiana.

Pesquisa Tabu: A pesquisa tabu usa um mecanismo de "memória" que força o algoritmo a explorar novas áreas do espaço de pesquisa. Este processo é conseguido memorizando algumas soluções examinadas recentemente. Tornando-se assim soluções tabu (proibidas), não as considerando na escolha da solução seguinte a pesquisar.

Algoritmos Genéticos Híbridos: São algoritmos de pesquisa heurísticos baseados numa população. Basicamente combinam uma pesquisa heurística local com os operadores genéticos. Este tipo de algoritmo também é designado por *memetic algorithms*

B Teste de Mann-Whitney

B.1 Introdução

Quando se pretende comparar duas amostras estatísticas relativamente ao seu valor médio, para uma certa variável, pode ser usado o teste-t de Student quando as amostras são independentes. Alternativamente podem ser adoptadas outros testes não-paramétricos[1] tais como os testes de Wald-Wolfowitz, de Mann-Whitney [149, 150] e de Kolmogorov-Smirnov [151] quando se consideram duas amostras independentes. Por outro lado, quando se pretendem comparar mais que duas amostras deve-se usar a análise da variância [152, 153] como, por exemplo, o teste de Kruscal-Wallis ou o teste da mediana.

Os resultados das experiência são expressos em função da sua significância. Neste contexto, a significância refere-se essencialmente aos resultados da análise estatística dos dados. A significância, expressa através de um valor probabilístico, indica quão diferentes são as amostras analisadas. Assegura também que o resultado da experiência realizada é válido e que não se deve a um enviesamento (*e.g.*, uma amostra

[1] Os testes não-paramétricos são usados quando não se conhecem os parâmetros da variável de interesse da população. Assim, os testes não incidem na estimação de parâmetros que descrevem a função de distribuição estatística da variável.

não representativa da população ou a influência de uma variável desprezada).

O nível mais baixo de confiança normalmente aceite é de 5%, isto é de 0,05 de nível de probabilidade ou de significância, p. Nos casos em que se verifica $p < 0{,}05$ é aceite a hipótese alternativa[2] e rejeitada a hipótese nula[3]. Quanto mais baixo é este valor maior é o nível de confiança da experiência realizada.

Neste apêndice é descrito o método de Mann-Whitney utilizado na tese.

B.2 Método de Mann-Whitney

B.2.1 Introdução

O teste não-paramétrico U de Mann-Whitney é usado para determinar a significância, ou o grau de diferença, entre duas populações. Deste modo, através de duas amostras da população, o teste U de Mann-Whitney é adoptado para testar a hipótese nula H_0: "As duas populações seguem a mesma função de distribuição" ou "Não existe significância entre as duas populações".

As condições necessárias para aplicar o método de Mann-Whitney são as seguintes:

1. As duas amostras são escolhidas aleatoriamente de duas populações;

2. A variável dependente é contínua, capaz em princípio, se não na prática, de produzir medidas em todo o domínio;

3. As medidas das duas amostram devem ter as propriedades de uma medida ordinal escalar, de modo a que tenha sentido falar em "maior que", "menor que" e "igual a".

[2]Hipótese alternativa é o oposto da hipótese nula, ver nota seguinte.
[3]A hipótese nula é usada para indicar a inexistência de diferença.

Valor bruto	Amostra	Posição	Posto
3,5	A	1	1
3,7	A	2	2
4,0	A	3	3
4,1	A	4	4
4,2	A	5	5,5
4,2	B	6	5,5
4,3	B	7	7
4,4	B	8	8
4,5	A	9	9
4,6	B	10	10
4,8	A	11	11
5,1	A	12	12
5,2	B	13	13
5,5	B	14	15,5
5,5	A	15	15,5
5,5	A	16	15,5
5,5	B	17	15,5
6,2	A	18	18
6,8	B	19	19
6,9	B	20	20
7,1	B	21	21
Soma (T_A)	A		96,5
Média μ_A	A		8,8
Soma (T_B)	B		134,5
Média μ_B	B		13,5

Tabela B.1. Conjunto formado pelas amostras A e B

O método de Mann-Whitney pode ser determinado da forma apresentada de seguida.

B.2.2 Descrição do modo de funcionamento

Considere-se duas amostras A e B retiradas aleatoriamente, respectivamente das populações P_A e P_B. Inicialmente começa-se por unir as duas amostras num único conjunto de $N = n_a + n_b$ elementos. De seguida ordena-se o conjunto de forma ascendente. O posto atribuído a cada elemento da amostra é obtido pela médias das posições de todos os elementos cujo "valor bruto" seja igual (ver tabela B.1).

Calculam-se as somas (T_A, T_B) e T_{AB} e as médias (μ_A, μ_B) e μ_{AB} dos postos das duas amostras:

$T_A = 96{,}5$; $\mu_A = 8{,}8$; $T_B = 134{,}5$; $\mu_B = 13{,}5$; $T_{AB} = T_A + T_B = 231$ e $\mu_{AB} = 11$.

O efeito de substituir os "valores brutos" por "postos" tem duas vantagens:

- Transforma o problema para domínios com relações ordinais ($<, >, =$) ignorando que esses "valores brutos" não derivem de uma escala com intervalos iguais;
- Transforma o vector de dados num tipo de sistema fechado onde se pode tirar partido das propriedades deste tipo de conjunto.

Em geral, a soma dos postos de qualquer conjunto com N elementos pode ser calculado através da expressão:

$$T_{AB} = \frac{N(N+1)}{2} \tag{B.1}$$

e a média será obviamente dada por:

$$\mu_{AB} = \frac{T_{AB}}{N} = \frac{N+1}{2} \tag{B.2}$$

Considere-se a hipótese nula H_0: "os conjuntos A e B não diferem significativamente". Se esta hipótese for verdadeira, então os "valores brutos" das amostras serão parecidos e os "postos" provenientes delas ficam ordenados de forma aleatória. Assim, se a hipótese nula for verdadeira, será de esperar que a média da soma de cada posto T_i se aproxime da média geral μ_{AB}. Consequentemente, a soma dos postos T_i de cada grupo deve-se aproximar dos valores:

$$\tilde{T}_A = n_a \frac{N+1}{2} = 121$$

$$\tilde{T}_B = n_b \frac{N+1}{2} = 110$$

Por outras palavras, se a hipótese nula for verdadeira pode dizer-se que:

- $T_A = 96{,}5$ pertence a uma distribuição amostrada cuja média é igual a 121;
- $T_B = 134{,}5$ pertence a uma distribuição amostrada cuja média é igual a 110;

Assim, para qualquer situação as distribuições têm a mesma variância e desvio padrão. Sendo o desvio padrão dado pela fórmula:

$$\sigma_T = \sqrt{\frac{n_a n_b (N+1)}{12}} \tag{B.3}$$

No exemplo indicado obtém-se $\sigma_T = 14{,}2$.

Num conjunto onde N postos estão bem distribuídos pelas duas amostras de dimensão n_a e n_b ($n_a \geq 5$ e $n_b \geq 5$), pode afirmar-se que as distribuições T_A e T_B tendem aproximar-se de uma função de distribuição normal.

Conhecida a média e desvio padrão de uma amostra que segue uma distribuição normal, podem-se transformar os valores observados de T_A e T_B numa versão apropriada da relação–z gaussiana e indexar o resultado à distribuição normal unitária. Na relação–z deve-se incorporar uma correcção de 0,5 devido à continuidade para acomodar o facto de T ser intrinsecamente discreto.

Assim, pode-se calcular a significância considerando $T_{\text{obs}} = T_i$, $\mu_T = \mu_i$ e $\sigma_T = \sigma_i$ com $i = \{A \text{ ou } B\}$.

Sendo a estrutura geral da relação dado pela seguinte fórmula:

$$z = \frac{(T_{\text{obs}} - \mu_T) + 0{,}5\gamma}{\sigma_T} \tag{B.4}$$

	Nível de significância, p				
Teste direccional	0,05	0,025	0,01	0,005	0,0005
Teste bidireccional	$--$	0,05	0,02	0,01	0,001
$z_{\text{crítico}}$	**1,645**	**1,960**	**2,326**	**2,576**	**3,291**

Tabela B.2. Valores críticos de z

$$\gamma = \begin{cases} -1; & T_{\text{obs}} > \mu_T \\ 1; & T_{\text{obs}} < \mu_T \end{cases} \tag{B.5}$$

No exemplo considerado os valores críticos das amostras são os seguintes:

$$z_A = \frac{96{,}5 - 121 + 0{,}5}{14{,}2} = -1{,}69 \tag{B.6}$$

$$z_B = \frac{134{,}5 - 110 - 0{,}5}{14{,}2} = 1{,}69 \tag{B.7}$$

A simetria observada entre os valores de z_A e de z_B não acontece por coincidência. De facto, em todos os casos, z_A e z_B devem ser simétricos. Na tabela B.2 pode-se ver os valores críticos de z. Assim, se se considerar a hipótese alternativa H_1 como "A população A tem tendência a fornecer valores mais baixos" pela tabela pode concluir-se que os resultados são significativos com $p \leq 0{,}05$ (teste de Mann-Whitney, unilateral, $z = -1{,}69$). Conclui-se assim, que a população P_A tem tendência a apresentar valores mais baixos. Por outro lado, se a hipótese alternativa é "Qual a população que tem tendência a obter valores mais baixos", os resultado não se distinguem significativamente (teste de Mann-Whitney, bidireccional, $z = -1{,}69$).

Se a hipótese nula for verdadeira, a probabilidade dos resultados da amostra A incidirem na parte direita de $z = -1{,}69$ com o valor de distribuição normal, no caso do teste bidireccional, de $p = 0{,}0455$, ver figura B.1.

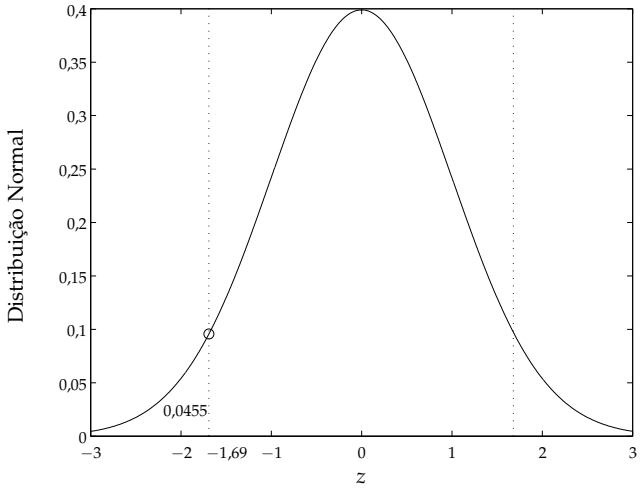


Figura B.1. Distribuição normal com média $\mu = 0$ e desvio padrão $\sigma = 14{,}2$

B.3 Resumo

Neste apêndice foi apresentado os conceitos fundamentais do teste não-paramétrico *U* de Mann-Whitney. Este teste é utilizado para comparar o desempenho do método multi-objectivo MaxiMin.

C Cálculo fraccionário

C.1 Introdução

O cálculo fraccionário (CF) é uma extensão natural da matemática clássica. Apesar do CF ser um tema de pesquisa contemporâneo do cálculo diferencial e integral, muitos investigadores desconhecem a sua existência. O primeiro documento sobre CF data de 1695 quando L'Hopital escreveu uma carta a Leibniz onde o questiona sobre o significado de $D^n y$ quando $n = 1/2$. Desde então os termos CF e o "cálculo diferencial e integral de ordem arbitrária" são usados para denominarem esta área.

A partir das ideias iniciadas por Leibniz muitos matemáticos importantes tais como Euler (1930), Laplace (1812), Fourier (1822), Abel (1823), Lioville (1823) e Riemann (1847), entre outros realizaram investigação nesta área. Apesar do trabalho desenvolvido, muitos aspectos do CF estão longe de estarem clarificados pois a sua complexidade acrescida cria uma barreira à sua aplicação. Por exemplo, ao contrário do cálculo diferencial de ordem-inteira, existem várias definições alternativas para a derivada fraccionaria (DF). As expressões propostas conduzem a pontos de vista complementares mas não existe, até ao momento, uma interpretação geométrica simples para as DFs.

Contudo, o desenvolvimento da teoria do caos e dos fractais, revelam a existência de uma relação profunda com as derivadas e integrais fraccionários (DIFs) o que motivou interesse renascido no CF. Um aspecto importante do CF é o facto inquestionável de que as DFs revelam propriedades que não são suportadas pelas derivadas de ordem inteira. Por exemplo, a característica de memória de um modelo de ordem fraccionária parece estar relacionado com a irreversibilidade dos processos físicos, enquanto que os modelos de ordem inteira só fornecem descrições bidireccionais no tempo.

Os aspectos fundamentais da teoria do CF podem ser encontrados em [154–157]. No que concerne a aplicações do CF podem ser mencionadas investigações em visco-elasticidade/amortecimento, caos/fractais, biologia, electrónica, processamento de sinal, difusão e propagação de ondas, infiltração, modelação, controlo e irreversibilidade [158–171].

Na secção C.2 é feita uma introdução sucinta ao cálculo fraccionário.

C.2 Introdução ao cálculo fraccionário

Desde a fundação do cálculo fraccionário a generalização do conceito de derivada e integral em ordem não-inteira α (α real ou complexo) tem sido sujeita a aproximações distintas (tabela C.1). A partir das definições apresentadas na tabela é possível calcular as DIFs de várias funções. Algumas funções elementares encontram-se na tabela C.2.

Apesar de existirem várias definições alternativas de DIFs que, na realidade, são equivalentes, existem algumas definições que se adequam mais em certo tipo de problemas. Por exemplo, a definição de Laplace/Fourier de uma derivada de ordem fraccionária $\alpha \in \mathbb{C}$ do sinal $x(t)$, $D^{\alpha}[x(t)]$ é uma generalização "directa" da expressão clássica de ordem inteira (C.1) sendo a transformada do sinal multiplicada pelo operador $s/\mathrm{j}w$. Assim, no âmbito da teoria do controlo automático significa

Tabela C.1. Definições de derivadas e integrais fraccionários

Liouville	$(I_c^\alpha \varphi)(x)$	$=$	$\frac{1}{\Gamma(\alpha)} \int_{-\infty}^{x} \frac{\varphi(t)}{(x-t)^{1-\alpha}} \mathrm{d}t,$	$-\infty < x < +\infty$
	$(D_c^\alpha f)(x)$	$=$	$\frac{1}{\Gamma(1-\alpha)} \frac{\mathrm{d}}{\mathrm{d}x} \int_{-\infty}^{x} \frac{f(t)}{(x-t)^{\alpha}} \mathrm{d}t,$	$-\infty < x < +\infty$
Riemann-	$(I_{a+}^\alpha \varphi)(x)$	$=$	$\frac{1}{\Gamma(\alpha)} \int_{a}^{x} \frac{\varphi(t)}{(x-t)^{1-\alpha}} \mathrm{d}t,$	$a < x$
Liouville	$(D_{a+}^\alpha f)(x)$	$=$	$\frac{1}{\Gamma(1-\alpha)} \frac{\mathrm{d}}{\mathrm{d}x} \int_{a}^{x} \frac{f(t)}{(x-t)^{\alpha}} \mathrm{d}t,$	$a < x$
Hadamard	$(I_{+}^\alpha \varphi)(x)$	$=$	$\frac{1}{\Gamma(\alpha)} \int_{0}^{x} \frac{\varphi(t)}{t[\ln(t/x)]^{1-\alpha}} \mathrm{d}t,$	$x > 0, a > 0$
	$(D_{a+}^\alpha f)(x)$	$=$	$\frac{\alpha}{\Gamma(1-\alpha)} \int_{a}^{x} \frac{f(x)-f(t)}{t[\ln(x/t)]^{1+\alpha}} \mathrm{d}t$	
Grünwald-	$(I_{a+}^\alpha \varphi)(x)$	$=$	$\frac{1}{\Gamma(\alpha)} \lim_{h \to +0} \left[h^\alpha \sum_{j=0}^{(x-a)/h} \frac{\Gamma(\alpha+j)}{\Gamma(j+1)} \varphi(x-jh) \right]$	
Letnikov	$(D_{a+}^\alpha f)(x)$	$=$	$\lim_{h \to 0} \left[\frac{1}{h^\alpha} \sum_{k=0}^{+\infty} (-1)^k \frac{\Gamma(\alpha+1)}{\Gamma(k+1)(\alpha-k+1)} x(t-kh) \right]$	
Chen	$(I_c^\alpha \varphi)(x)$	$=$	$\frac{1}{\Gamma(\alpha)} \int_{c}^{x} \varphi(t)(x-t)^{\alpha-1} \mathrm{d}t,$	$x > c$
	$(D_c^\alpha f)(x)$	$=$	$\frac{1}{\Gamma(1-\alpha)} \frac{\mathrm{d}}{\mathrm{d}x} \int_{c}^{x} f(t)(x-t)^{-\alpha} \mathrm{d}t,$	$x > c$
Marchaud	$(D_{+}^\alpha \varphi)(x)$	$=$	$\frac{\alpha}{\Gamma(1-\alpha)} \int_{-\infty}^{x} \frac{f(x)-f(t)}{(x-t)^{1+\alpha}} \mathrm{d}t$	
Fourier	$\mathscr{F}\{I_{\pm}^\alpha \varphi\}$	$=$	$\frac{\mathscr{F}\{\varphi\}}{(\pm \mathrm{j}w)^\alpha},$	$0 < \Re(\alpha) < 1$
	$\mathscr{F}\{D_{\pm}^\alpha \varphi\}$	$=$	$(\pm \mathrm{j}w)^\alpha \mathscr{F}\{\varphi\},$	$\Re(\alpha) \geq 0$
Laplace	$\mathscr{L}\{I_{0+}^\alpha \varphi\}$	$=$	$\frac{\mathscr{L}\{\varphi\}}{s^\alpha},$	$\Re(\alpha) > 0$
	$\mathscr{L}\{D_{0+}^\alpha \varphi\}$	$=$	$s^\alpha \mathscr{L}\{\varphi\},$	$\Re(\alpha) \geq 0$

Tabela C.2. DIFs de algumas funções elementares

$\varphi(x), x \in \mathbb{R}$	$(I_{+}^\alpha \varphi)(x), x \in \mathbb{R}, \alpha \in \mathbb{C}$	
$(x-a)^{\beta-1}$	$\frac{\Gamma(\beta)}{\Gamma(\alpha+\beta)}(x-a)^{\alpha+\beta-1},$	$\Re(\beta) > 0$
$\mathrm{e}^{\lambda x}$	$\lambda^{-\alpha}\mathrm{e}^{\lambda x},$	$\lambda^{-\alpha}\mathrm{e}^{\lambda x} > 0, \Re(\lambda) > 0$
$\sin(\lambda x)$	$\lambda^{-\alpha}\sin(\lambda x - 0{,}5\alpha\pi),$	$\lambda > 0, \Re(\alpha) > 1$
$\cos(\lambda x)$	$\lambda^{-\alpha}\cos(\lambda x - 0{,}5\alpha\pi),$	$\lambda > 0, \Re(\alpha) > 1$
$\mathrm{e}^{\lambda x}\sin(\gamma x)$	$\frac{\mathrm{e}^{\lambda x}}{\lambda^2+\gamma^2}\sin(\gamma x - \alpha\phi),$	$\phi = \arctan(\gamma/\lambda), \gamma > 0, \Re(\lambda) > 1$
$\mathrm{e}^{\lambda x}\cos(\gamma x)$	$\frac{\mathrm{e}^{\lambda x}}{\lambda^2+\gamma^2}\cos(\gamma x - \alpha\phi),$	$\phi = \arctan(\gamma/\lambda), \gamma > 0, \Re(\lambda) > 1$

que a análise dos métodos baseados na frequência têm uma adaptação imediata.

$$\mathscr{L}\{D^{\alpha}[x(t)]\} = s^{\alpha}X(s) \tag{C.1}$$

Se se considerar o sistema de controlo representado na figura C.1, com $1 < \alpha < 2$, então o diagrama de Bode em malha aberta (figura C.2) apresenta um declive de -20α dB/dec e uma fase, constante, de $-\alpha\pi/2$ rad. Consequentemente, o sistema em malha fechada tem uma margem de fase constante de $\pi(1 - \alpha/2)$ rad que é independente do ganho do sistema. Esta importante propriedade também se pode observar através do lugar das raízes ilustrado na figura C.3.

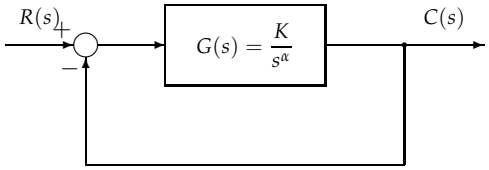
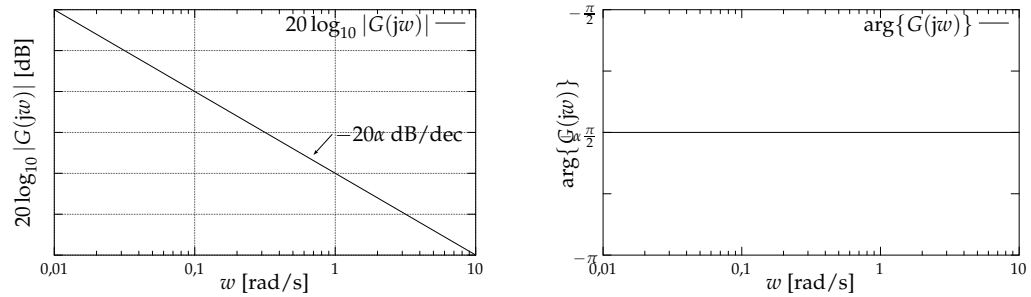


Figura C.1. Diagrama de blocos



(a) Diagrama de módulo (b) Diagrama de fase

Figura C.2. Diagrama de Bode em malha aberta do sistema de ordem fraccionária ilustrado na figura C.1, com $1 < \alpha < 2$

Por exemplo, quando $1 < \alpha < 2$ o lugar das raízes segue a relação $\pi - \alpha\pi/2 = \arccos(\zeta)$, onde ζ é o coeficiente de amortização, que é independente do ganho K do sistema. A implementação de DIFs baseado na definição de Laplace/Fourier adopta o domínio da frequência e requer um número infinito de pólos e zeros obedecendo a uma relação recursiva. Por exemplo, considere o circuito implementado na figura C.4. A partir da figura pode ver que:

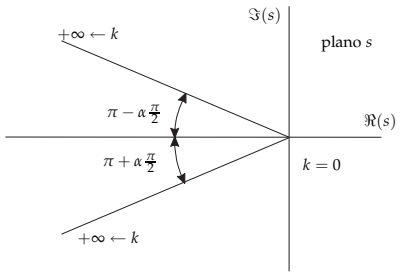


Figura C.3. Lugar das raízes para o sistema de controlo ilustrado na figura C.1, com $1 < \alpha < 2$

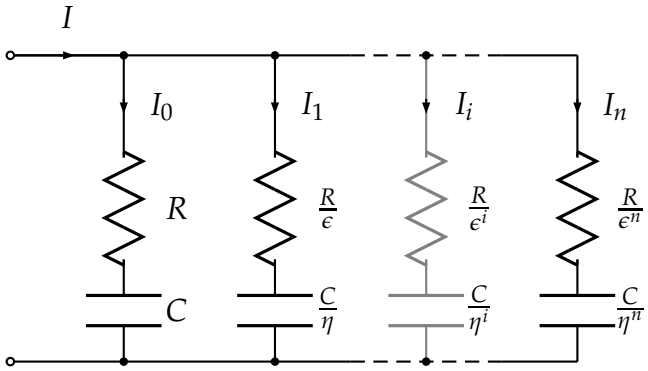


Figura C.4. Circuito eléctrico recursivo com elementos resistivos e capacitivos, $1 < i < n$

$$I = \sum_{i=0}^{n} I_i \tag{C.2a}$$

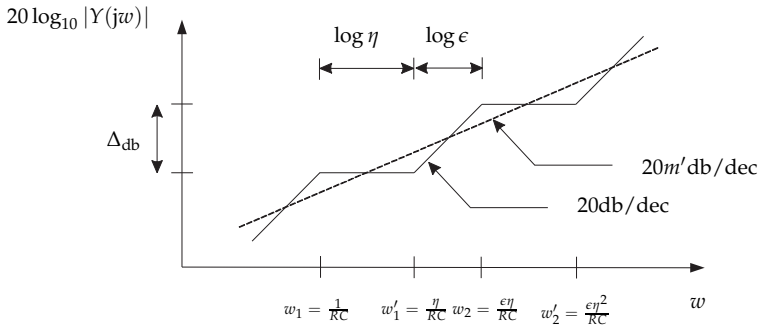
$$R_{i+1} = \frac{R_i}{\epsilon} \tag{C.2b}$$

$$C_{i+1} = \frac{C_i}{\eta} \tag{C.2c}$$

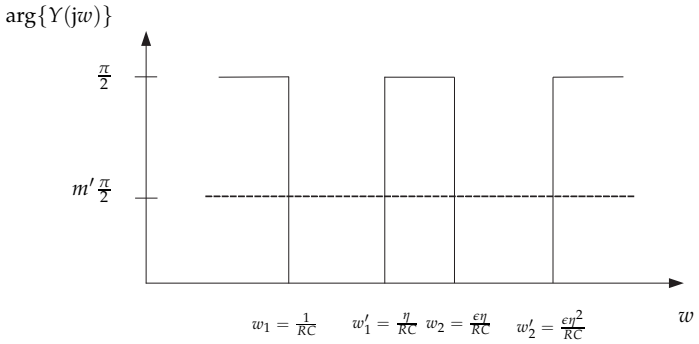
onde ϵ e η são factores escalares, I_i é a corrente resultante da tensão aplicada e R_i e C_i são, respectivamente os elementos resistivos e capacitivos do ramo i. A admitância $Y(\mathrm{j}w)$ é dado por:

$$Y(\mathrm{j}w) = \frac{I(\mathrm{j}w)}{V(\mathrm{j}w)} = \sum_{i=0}^{n} \frac{\mathrm{j}wC\epsilon^i}{\mathrm{j}CR + (\eta\epsilon)^i} \tag{C.3}$$

A figura C.5 ilustra o diagrama assimptotico de Bode de $Y(\mathrm{j}w)$. As frequências dos pólos e zeros (w_i e w_i') obedecem a relação recursiva:



(a) Amplitude



(b) Fase

Figura C.5. Diagrama de Bode do circuito eléctrico recursivo

$$\frac{w'_{i+1}}{w'_i}=\frac{w_{i+1}}{w_i}=\epsilon\eta;\ \frac{w_i}{w'_i}=\epsilon;\ \frac{w'_{i+1}}{w_i}=\eta$$

A partir da amplitude ou da fase do diagrama de Bode, a inclinação média m' pode ser determinada através da expressão:

$$m'=\frac{\log\epsilon}{\log\epsilon+\log\eta} \tag{C.4}$$

No entanto, esta aplicação tem algumas desvantagens. Numa aproximação real o número finitos de pólos e zeros introduz uma ondulação na resposta em frequência encontra-se limitada a uma certa faixa de frequências. Por outro lado, a conversão para tempo discreto requer um número adicional de cálculos e as aproximações tornam-se difíceis de analisar. O método é restringido aos casos onde a resposta em frequência é bem conhecida, pelo que ocorrem circunstâncias onde é frequente

surgirem problemas na sua implementação.

Uma aproximação alternativa, baseada no conceito de diferencial fraccionário, consiste na definição de Grünwald-Letnikov dada pela equação (C.5):

$$D^{\alpha}\left[x(t)\right] = \lim_{h \to 0} \left[\frac{1}{h^{\alpha}} \sum_{k=0}^{+\infty} (-1)^k \frac{\Gamma(\alpha+1)}{\Gamma(k+1)(\alpha-k+1)} x(t-kh) \right] \tag{C.5}$$

$$\Gamma(z) = \int_0^{+\infty} e^{-t} t^{z-1} dt, \ \Re(z) > 0 \tag{C.6}$$

onde Γ é a função gama de Euler (C.6), que generaliza o conceito de factorial para valores complexos, e h é o incremento no tempo. Uma propriedade importante revelada pela equação (C.5) é que, enquanto uma derivada de ordem inteira implica uma série de termos finitos, a derivada de ordem fraccionária requer uma série de termos infinitos. Isto significa que as derivadas inteiras são operadores "locais" em oposição com as derivadas fraccionais que tem, implicitamente, uma memória de todos os eventos passados.

Esta fórmula inspira um algoritmo para o cálculo de DIFs em tempo discreto [172], baseado na aproximação do incremento do tempo h através do tempo de amostragem T, levando a equação no domínio dos tempos (C.7) onde $X(z) = \mathscr{Z}\{x(t)\}$.

$$\mathscr{Z}\{D^{\alpha}[x(t)]\} \approx \left[\frac{1}{T^{\alpha}} \sum_{i=0}^{\infty} \frac{(-1)^k \Gamma(\alpha+1)}{k!\Gamma(\alpha-k+1)} z^{-k} \right] X(z) = \left(\frac{1-z^{-1}}{T} \right)^{\alpha} X(z) \tag{C.7}$$

Uma implementação real da equação (C.7) corresponde a truncar a série no termo n resultando a equação (C.8).

$$\mathscr{Z}\{D^{\alpha}[x(t)]\} \approx \left[\frac{1}{T^{\alpha}} \sum_{i=0}^{n} \frac{(-1)^k \Gamma(\alpha+1)}{k!\Gamma(\alpha-k+1)} z^{-k} \right] X(z) \tag{C.8}$$

Consequentemente, para obter uma boa aproximação, deve-se usar um valor elevado para n e um valor de amostragem, T, pequeno.

C.3 Resumo

Neste apêndice foram apresentados alguns conceitos básicos fundamentais do cálculo de derivadas e integrais de ordem fraccionárias.

Referências

[1] Holland, J.H.: A Adaptation in Natural and Artificial Systems: An Introduction analysis with Applications to Biology , Control, and Artificial Intelligence. MIT Press (1992)

[2] Solteiro Pires, E., Tenreiro Machado, J.: Trajectory optimization for redundant robots using genetic algorithms. In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.G., eds.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), Las Vegas, Nevada, USA, Morgan Kaufmann (2000) 967

[3] Solteiro Pires, E., Tenreiro Machado, J.: Trajectory optimization for redundant robots using genetic algorithms with heuristic operators. In Whitley, D., ed.: Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, Las Vegas, Nevada, USA (2000) 290–296

[4] Solteiro Pires, E.J., Tenreiro Machado, J.A.: A GA perspective of the energy requirements for manipulators maneuvering in a workspace with obstacles, San Diego, California, USA, CEC 2000 – Congress on Evolutionary Computation (2000) 1110–1116

[5] Solteiro Pires, E., Tenreiro Machado, J., de Moura Oliveira, P.B.: An evolutionary optimization for robotic manipulators. In: INES'2002 - 6th International Conference on Intelligent Engineering Systems, Opatija, Croatia (2002) 131–136

[6] Solteiro Pires, E., Tenreiro Machado, J., de Moura Oliveira, P.B.: A real time trajectory planner for 2*R* robotic manipulators. In: Controlo'2002 - 5th International Portuguese Conference on Automatic Control, Aveiro, Portugal (2002) 471–476

[7] Solteiro Pires, E., Tenreiro Machado, J., de Moura Oliveira, P.B.: A Real Time Optimization for 2*R* Manipulators. In: Intelligent Systems at the Service of Mankind. Ubooks (2004) 109–119

[8] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: An evolutionary approach to robot structure and trajectory optimization, Budapest, Hungary, ICAR'01-10th International Conference on Advanced Robotics (2001) 333–338

[9] Solteiro Pires, E., Tenreiro Machado, J., de Moura Oliveira, P.B.: Structure and trajectory optimization for redundant manipulators. In: EUNITE 2001 - European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems, Tenerife, Spain (2001) 85–90

[10] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: Robotic manipulator synthesis using a hierarchical multi-objective genetic algorithm. In: 7º Congresso Interamericano, CAIP'2005 - Computation Aplicada a La Industria de Processos, Vila Real, Portugal (2005)

[11] Solteiro Pires, E.J., de Moura Oliveira, P.B., Tenreiro Machado, J.A.: Multi-objective genetic manipulator trajectory planner. In Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G., eds.: Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT,

EvoIASP, EvoMUSART, EvoSTOC. Volume 3005 of LNCS., Coimbra, Portugal, Springer Verlag (2004) 219–229

[12] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: Robot Trajectory Planning Using Multiobjective Genetic Algorithm Optimization. In et al., K.D., ed.: Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I, Seattle, Washington, USA, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102 (2004) 615–626

[13] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: Manipulator trajectory planning using a MOEA. Applied Soft Computing (Aceite para publicação)

[14] Solteiro Pires, E.J., de Moura Oliveira, P.B., Tenreiro Machado, J.A.: Multiobjective maximin sorting scheme. In: Conference on Evolutionary Multicriterion Optimization – EMO 2005, Guanajuanto, México, Springer-Verlag, Lecture Notes in Computer Science Vol. 3410 (2005) 165–175

[15] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: Fractional order dynamics in a genetic algorithm. In: ICAR'03-11th International Conference on Advanced Robotics, Coimbra, Portugal (2003) 264–269

[16] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: Fractional order dynamics in a GA planner. Signal Process. **83** (2003) 2377–2386

[17] Solteiro Pires, E.J., Tenreiro Machado, J.A., de Moura Oliveira, P.B.: Fractional order dynamical phenomena in a GA. In Cantú-Paz, E., Foster, J.A., Deb, K., Davis, D., Roy, R., O'Reilly, U.M., Beyer, H.G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A.C., Dowsland, K., Jonoska, N., Miller, J., eds.: Genetic and Evolutionary Computation – GECCO-2003. Volume 2723 of LNCS., Chicago, Springer-Verlag (2003) 510–511

[18] Thoma, Y., Sanchez, E.: A Teconfigurable Chip for Envolvable Hardware. In

et al., K.D., ed.: Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I, Seattle, Washington, USA, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102 (2004) 816–827

[19] Koza, J., Bennett, F., Andre, D., Keane, M.: Genetic Programming III: Darwinian Invention and Problem Solving. (1999)

[20] Peterson, M.R., Doom, T.E., Raymer, M.L.: GA – facilitated knowledge discovery and pattern recognition applied to the biochemistry of protein solvation. In: Genetic and Evolutionary Computation – GECCO-2004. Volume 1 of LNCS., Berlin, Springer-Verlay (2004) 426–437

[21] Fogel, D.B.: System Identification through Simulated Evolution: A Machine Learning Approach to Modeling. Ginn Press (1991)

[22] Paz Ramos, M.A., Torres Jimenez, J., Quintero Marmol Marquez, E., Estrada Esquivel, H.: Pid controller tuning for stable and unstable processes applying ga. In: Genetic and Evolutionary Computation – GECCO-2004. Volume 2 of LNCS., Berlin, Springer-Verlay (2004) 1–10

[23] de Moura Oliveira, P., Jones, A.: Co-evolutionary design pid control structures. In: IFAC workshop on digital control: Past, Present and Future of PID Control PID'00, Terrassa, Spain (2000) 325–330

[24] Balling, R.: The maximin fitness function; multi-objective city and regional planning. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L., eds.: Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings. Volume 2632 of Lecture Notes in Computer Science., Springer (2003) 1–15

[25] Koza, J., Bennett, F., Andre, D., Keane, M.: Chapter 16. In: Genetic Programming III: Darwinian Invention and Problem Solving. (1999)

[26] Ritchie, M.D., Coffey, C.S., Moore, J.H.: Gennetic programming neuronal

networks as a bioinformatics tool for human genetics. In: Genetic and Evolutionary Computation – GECCO-2004. Volume 1 of LNCS., Berlin, Springer-Verlay (2004) 438–460

[27] Lai, L.L.: Intelligent System Applications in Power Engineering: Evolutionary Programming and Neural Networks. John Wiley & Sons (1998)

[28] Kingdon, J.: Intelligent Systems and Financial Forecasting. Springer-Verlag Berlin and Heidelberg GmbH & Co. K (1997)

[29] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison – Wesley (1989)

[30] Koza, J.R.: Genetic Programming On the Programming of Computers by Means of Natural Selection. A Bradford Book. The MIT Press, (Cambridge, Massachusetts, London, England)

[31] Kinnear, K.E.J.: Advances in Genetic Programming. The MIT Press, (Cambridge, Massachusetts, London, England)

[32] Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer (2001)

[33] Langdon, W.B.: Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming! Kluwer, Boston (1998)

[34] Yokose, Y., Cingoski, V., Kaneda, K., Yamashita, H.: Performance comparison between gray coded and binary coded genetic algorithms for inverse shape optimization of magnetic devices. Applied Electromagnetics (2000) 115–120

[35] Fogel, D.B.: Evolutionary computation: A new transations. IEEE Transactions on Evolutionary Computation **1** (1997) 1–2 Editor in Chief Natural Selection, Inc, La Jolla, CA 92037 USA.

[36] Man, K.F., Tang, K.S., Kwong, S.: Genetic algorithms: Concepts and aplications. IEEE Transactions on Industrial, Electronics **43** (1996)

[37] Beasley, D., Bull, D.R., Martin, R.R.: An overview of genetic algorithms: part 1, fundamentals. In: Inter Committee on Computing, University Computing (1993) 58–69

[38] Beasley, D., Bull, D.R., Martin, R.R.: An overview of genetic algorithms: part 2, fundamentals. In: Inter Committee on Computing, University Computing (1993) 170–180

[39] Grefenstette, J.J.: Reducing bias and ineficienty in the selection algorithm. In Ed., L.E.A., ed.: Proc. 2nd Int. Conf. Genetic Algorithms. (1995) 14–21

[40] Davidor, Y.: Analogous crossover. In: Procedings of the third International Conference on Genetic Algorithms, George Manson University (1989) 98–103

[41] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. third edn. Springer–Verlag (1996)

[42] Bäck, T.: Evolutionary Algorithms in theory and Practice, Evolutionary Strategies – Evolutionary Programming – Genetic Algorithms. Oxford University Press, New York, Oxford (1996)

[43] Q.Zhu, K., Liu, Z.: Empirical study of population diversity in permutation-based genetic algorithm. In et al., K.D., ed.: Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part II, Seattle, Washington, USA, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102 (2004) 420–421

[44] Baudet, P., Azzaro, C., Pibouleau, L., Domenech, S.: A genetic algorithm for batch chemical plant scheduling. In: CHISA'96 – International Congress of Chemical and Process Engineering, Prague (1996)

[45] Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, LTD (2001)

[46] Bäck, T., Fogel, D.B., Michalewicz, Z.: Evolutionary Computation 2, Advanced Algorithms and Operators. IOP – Institute of Physics Publishing, Bristol and Philadelphia (2000)

[47] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley-Interscience Series in Systems and Optimization. (2001)

[48] Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Archiving with guaranteed convergence and diversity in multi-objective optimization. (2002)

[49] Knowles, J., Corne, D.: Bounded Pareto Archiving: Theory and Practice. In: Metaheuristics for Multiobjective Optimisation. Volume 535 of Lecture Notes in Economics and Mathematical Systems. Springer (2004) 39–64

[50] Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In Erlbaum, L., ed.: Proceedings of the First International Conference on Genetic Algorithms. (1985) 93–100

[51] Hajela, P., Lee, E., Lin, C.: Genetic algorithms in structural topology optimization. In: Proceedinggs of the NATO Advanced Research Workshop on Topology Design of structures, Sesimbra, Portugal (1993) 117–133

[52] Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Fifth International Conference on Genetic Algorithms. (1993) 416–423

[53] Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation **2** (1994) 221–248

[54] Horn, J., Nafploitis, N., Goldberg, D.: A niched pareto genetic algorithm for multi-objective optimization, Proceedings of the First IEEE Conference on Evolutionary Computation (1994) 82–87

[55] Laumanns, M., Rudolph, G., Schwefel, H.P.: A spatial predator-prey approach to multi-objective optimization: a preliminary study. In Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P., eds.: Proceedings of theFifthConference on Parallel Problem Solving from Nature(PPSN V). Volume 1498., Amsterdam, Springer (1998) 241–249

[56] Rudolph, G.: Evolutionary search under partially ordered finite sets. In Sebaaly, M.F., ed.: Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001), Dubai, U. A. E., ICSC Academic Press (2001) 818–822

[57] Osyczka, A., Kundu, S.: A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Structural Optimization **10** (1995) 94–99

[58] Zitzler, E., Thiele, L.: ;multiobjective algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation **3** (1999) 257–271

[59] Knowles, J.D., Corne, D.: Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation **8** (2000) 149–172

[60] Deb, K., Mohan, M., Mishra, S.: A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. KanGal Report 2003002, Indian Institute of Technology Kanpur (2003)

[61] Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, Massachusetts (1995)

[62] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.P., eds.: Parallel Problem Solving from Nature – PPSN VI. Volume 1917 of LNCS., Berlin, Springer (2000) 849–858

[63] Zitzler, E., D.K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8** (2000) 173–195

[64] Tan, K.C., Lee, T.H., Khor, E.F.: Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. Artificial Intelligence Review **17** (2002) 251–290

[65] Okabe, T., Jin, Y., Sendhoff, B.: A critical survey of performance indices for multi-objective optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Volume 2. (2003) 878–885

[66] Wu, J., Azarm, S.: Metrics for quality assessment of a multiobjective design optimization solution set. Transactions of the ASME, Journal of Mechanical Design **123** (2001)

[67] Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective Evolutionary Algorithm Test Suites. In Carroll, J., Haddad, H., Oppenheim, D., Bryant, B., Lamont, G.B., eds.: Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, Texas, ACM (1999) 351–357

[68] Fleischer, M.: The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L., eds.: Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Faro, Portugal, Springer. Lecture Notes in Computer Science. Volume 2632 (2003) 519–533

[69] While, R.L.: A new analysis of the lebmeasure algorithm for calculating hypervolume. [173] 326–340

[70] Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. [173] 62–76

[71] Duarte, F.B.M.: Análise de Robots Redundantes. Phd, Faculdade de Engenharia da Universidade do Porto (2002)

[72] Gill, M.A., Zomaya, A.Y.: Obstacle Avoidance in Multi-Robot Systems: Experiments in Parallel Genetic Algorithms. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1998)

[73] Xiao, J., Michalewicz, Z., Zhang, L., Trojanowski, K.: Adaptative evolutionary planner / navigator for mobile robots. IEEE Transactions on Evolutionary Computation **1** (1997) 18–28

[74] Han, W.G., Min Baek, S., Kuc, T.Y.: Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots. In: Conference Proceedings IEEE International Conference on Systems, Man, and Cybernetics, Hyatt, Orlando, Florida, USA (1997) 2747–2751

[75] Ramírez, D.R., Limón, D., Gómez Ortega, J., Camacho, E.: Nonlinear MBPC for robot navigation using genetic algorithms. In: Proceedings of the 1999 IEEE In-ternational Conference on Robotics & Automation, Detroit, Michigan (1999) 2452–2457

[76] Gemeinder, M., Gerke, M.: GA-based path planning for mobile robot systems employing an active search algorithm. Applied Soft Computing **3** (2003) 149–158

[77] Dozier, G.V., McCullough, S., Homaifar, A., Moore, L.: Multiobjective Evolutionary Path Planning via Fuzzy Tournament Selection. In: IEEE International Conference on Evolutionary Computation (ICEC'98), Piscataway, New Jersey, IEEE Press (1998) 684–689

[78] Gacôgne, L.: Multiple objective optimization of fuzzy rules for obstacles avoiding by an evolution algorithm with adaptative operators. In: In Proceedings of the Fifth International Mendel Conference on Soft Computing (Mendel'99), Brno, Czech Republic (1999) 236–242

[79] Hocaoğlu, C., Sanderson, A.C.: Evolutionary multiple path planner for assembly. In: Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning, Porto, Portugal (1999) 81–87

[80] Hocaoğlu, C., Sanderson, A.: Planning multiple paths with evolutionary specification. IEEE Transations on Evolutionary Computation **5** (2001) 169–191

[81] Hocaoğlu, C., Sanderson, A.C.: Evolutionary path planning using multiresolution path representation. In: Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Leuven, Belgium (1998) 318–323

[82] Cai, Z., Peng, Z.: Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. Journal of Intelligent and Robots Systems **33** (2002) 61–71

[83] Chen, M., Zalzala, A.M.S.: A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. Journal Robotic Systems **14** (1997) 529–544

[84] Chen, M.W., Zalzala, A.M.S.: Dynamic modelling and genetic-based trajectory generation for non-holonomic mobile manipulators. Control Eng. Practice, Pergamon **5** (1997) 39–48

[85] Wu, K.H., Chen, C.H., Lee, J.D.: A ring cache genetic agorithm for tunning reliable fuzzy logic controller. In: Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Orlando, Florida, USA (1997) 2847–2852

[86] Davidor, Y.: Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization. Number 1 in Series in Robotics and Automated Systems. World Scientific Publishing Co. Pte Ltd (1991)

[87] Nearchou, A.C., Aspragathos, N.A.: Application of genetic algorithms to pointo-to-point motion of redundant manipulators. Mecha. Mach. Theory, Pergamon **31** (1995) 261–270

[88] Lavoie, M.H., Boudreau, R.: Obstacle avoidance for redundant manipulators using a genetic algorithm. In: Proc. of the 2001 CCToMM Symposium on Mechanisms, Machines, and Mechatronics, Montréal (2001)

[89] Doyle, A.B., Jones, D.: Robot path planning with genetic algorithms. In: 2nd Portuguese Conf. on Automatic Control, Porto, Potugal (1996) 312–318

[90] Dae Lee, Y., Hee Lee, B., Gyoo Kim, H.: An evolutionary approach for time optimal trajectory planning of a robotic manipulator. Information Sciences **113** (1999) 245–260

[91] Kubota, N., Arakawa, T., Fukuda, T.: Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm. In: IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico (1997) 205–210

[92] Kubota, N., Fukuda, T., Shimojima, K.: Trajectory planning of cellular manipulator system using virus-evolutionary genetic algorithm. Robotics and Autonomous systems **19** (1996) 85–94

[93] Luo, X., Wei, W.: A new immune genetic algorithm and its application in redundant manipulator path planning. Journal of Robotic Systems **21** (2004) 141–151

[94] Wei-Min, Y., Yu-Geng, X.: Optimum motion planning in joint space for robots using genetic algorithms. Robotics and Autonomous Systems **18** (1996) 373–393

[95] Wang, Q., Zalzala, A.M.S.: Genetic control of near time-optimal motion for an industrial robot arm. In: IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota (1996) 2592–2597

[96] Tian, L., Collins, C.: An effective robot trajectory planning method using a genetic algorithm, mechatronics. Mechatronics, In Press (2003)

[97] Rana, A., Zalzala, A.: An evolutionary planner for near time-optimal collision-free motion of multi-arm robotic manipulators. In: UKACC International Conference on Control. Volume 1. (1996) 29–35

[98] Ridao, M.A., Camacho, E.F., Riquelme, J., Toro, M.: An evolutionary and local search algorithm for motion planning of two manipulators. Journal of Robotic Systems **18** (2001) 463–476

[99] Ali, A.D.M.S., Babu, N.R., Varghese, K.: Offline path planning of cooperative manipulators using co-evolutionary genetic algorithm. In: Proceedings of International Symposium on Automation and Robotics in Construction, 19th (ISARC), National Institute of Standards and Technology, Gaithersburg, Maryland (2002) 415–424

[100] Garg, D.P., Kumar, M.: Optimal path planning and torque minimization via genetic algorithm applied to cooperating robotic manipulators. In: IMECE – Congress of American Society of Mechanical Engineers, Ney York (2001)

[101] Garg, D.P., Kumar, M.: Optimization techinques applied to multiple manipulators for path planning and torque minimization. Engineering Applications of Artificial Intelligence (2002) 241–252

[102] Ortmann, M.: Multi-criterion optimization of robot trajectories with evolutionary strategies. FACTA UNIVERSITATIS, Electronics and Energetics **14** (2001) 19–32

[103] Chedmail, P., Ramstein, E.: Robot mechanism synthesis and genetic algorithms. In: IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota (1996) 3466–3471

[104] Kim, J.O., Khosla, P.K.: A multi-population genetic algorithm and its application to design of manipulators. In: IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems, Raleight, North Caroline (1992) 279–286

[105] Gallant, M., Boudreau, R.: The synthesis of planar parallel manipulators with prismatic joints for an optimal, singularity-free space. Journal of Robotic Systems **19** (2000) 13–24

[106] Sobh, T.M., Wang, B., Patel, S.H.: Web enabled robot design and dynamic control simulation software solutions from task points description. In: IECON '03 – Industrial Electronics Society. Volume 2. (2003) 1221–1227

[107] Kosinska, A., Galicki, M., Kedzior, K.: Designing and optimization of parameters of delta-4 parallel manipulador for a given workspace. Journal of Robotic Systems **20** (2003) 539–548

[108] Chocron, O., Bidaud, P.: Evolutionary algorithms in kinematic design of robotic system. In: IEEE/RSJ International Conference on Intelligent Robotics and Systems, Grenoble, France (1997) 279–286

[109] Han, J., Chung, W.K., Youm, Y., Kim, S.H.: Task based design of modular robotic manipulator using efficient genetic algorithm. In: IEEE Int. Conf. on Robotics and Automation, Albuquerque, New Mexico (1997) 507–512

[110] Bi, Z.M., Zhang, W.J.: Concurrent optimal design of modular robotic configuration. Journal of Robotic systems **18** (2001) 77–87

[111] Zhu, Y., Qiu, J., Tani, J.: Simultaneous optimization of a two-link flexible robot arm. Journal of Robotic Systems **18** (2001) 29–38

[112] B. Parker, G., Braun, D.W., Cyliax, I.: Learning gaits for the stiquito. In: 8th International Conference on Advanced Robotics, Monterey, California, USA (1997) 285–290

[113] Cabodevila, G., Abba, G.: Quasi optimal gait for a biped robot using genetic algoritm. In: IEEE International Conference on Systems Man, and Cybernetics Computational Cybernetics and Simulation, Orlando, Florida (1997) 3960–3965

[114] Arakawa, T., Fukuda, T.: Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP layers. In: IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico (1997) 211–216

[115] Luk, B.L., Galt, S., Chen, S.: Using genetic algorithms to establish efficient walking gaits for an eight-legged robot. International Journal of Systems Science **32** (2001) 703–713

[116] Lewis, M.A., Fagg, A.H.: Genetic programming approach to the construction of a neural network for control of a walking robot. In: IEEE International Conference on Robotics and Automation. (1992) 2618–2623

[117] Fukuda, T., Komata, Y., Arakawa, T.: Stabilization control of biped locomotion robot based learning with gas having self adaptive mutation and recurrent neural networks. In: IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico (1997) 217–220

[118] Farritor, S., Dubowsky, S.: A self planning methodology for planetary robotic explorers. In: 8th International Conference on Advanced Robotics, Hyatt Regency Monterey, California, USA (1997) 449–504

[119] Erkmen, A.M., Durna, M.: Genetic algorithm-based optimal regrasping with the anthrobot 5-fingered robot hand. In: Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Leuven, Belgium (1998) 3329–3334

[120] Udawatta, L., Watanabe, K., Izumi, K., Kiguchi, K.: Control of underactuated robot manipulators using switching computed torque method: GA based approach. Soft Computing **8** (2003) 51–60

[121] Yue, S., Henrich, D.: Manipulating deformable linear objects: Attachable adjustement-motions for vibration reduction. Journal of Robotic Systems **18** (2001) 375–389

[122] Zhuang, H., Wu, J., Huang, W.: Optimal planning calibration experiments by genetic algorithms. In: IEEE International Conference on Robotics and Automation, Minneaplois, Minnesota (1996) 981–986

[123] Zhuang, H., Wu, J., Huang, W.: Optimal planning calibration experiments by genetic algorithms. Journal of Robotic Systems **14** (1997) 741–752

[124] Calafiore, A.G., Indri, A.M., Bona, A.B.: Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation. Journal of Robotic Systems **18** (2001) 55–68

[125] Coello, C.A.C., Christiansen, A.D., Aguirre, A.H.: Use of genetic algorithms for multiobjective optimization of counterweight balancing of robot arms. In: EXPERSYS-95 Expert Systems Applications and Artificial Intelligence. I. I. T. T. International, Technology Transfer Series, San Francisco, California, In Jacob J. G. Chen, editor (1995) 243–248

[126] Coello, C.A.C., Christiansen, A.D., Aguirre, A.H.: Multiobjective design optimization of counterweight balancing of a robot arm using genetic algorithms. In: TAI '95: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, IEEE Computer Society (1995) 20–23

[127] Berlanga, A., Sanchis, A., Isasi, P., Molina, J.: Generalization capabilities of co-evolution in learning robot behavior. Journal of robotic systems **19** (2002) 455–467

[128] Barberá, H.M., Gómez-Skarmeta, A.F.: A framework for defining and learning fuzzy behaviors for autonomous mobile robots. Int. Journal of Intelligent Systems **17** (2002) 1–20

[129] Nakashima, M., Maruyama, Y., Umeda, N., Hyura, N., Hasegawa, T.: Basic experiments on robot-base vibration control of the hot-line work robot system using genetic algorithm. Electrical Engeneering in Japan **123** (1998)

[130] Chappele, F., Bidaud, P.: Closed form solutions for inverse kinematics approximation of general *6*r manipulators. Mechanism and Machine Theory (2004) 323–338

[131] Karla, P., Mahapatra, P.B., Aggraewal, D.K.: On the solution of multimodal robot inverse kinematic functions using real-coded genetic algorithm. (2003)

[132] Karla, P., Prakash, N.R.: A neuro-genetic algorithm approach for solving the inverse kinematics of robotic manipulators. (2003)

[133] Silva, F., Machado, J.A.T.: Energy analysis during biped walking, Detroit, Michigan, USA, Proc. IEEE Int. Conf. Robotics and Automation (1999) 59–64

[134] Tenreiro Machado, J.A., Martins de Carvalho, J.L., Galhano, A.M.S.: Analysis of robot dynamics and compensation using classical and computed torque techniques. IEEE Transations on Educational **36** (1993) 372–379

[135] Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolutionary Strategies, Evolutionay Programming, Genetic Algorithms. Oxford University Press, Oxford, New York (1996)

[136] Kalyanmoy Deb, Manikanth Mohan, S.M.: Towards a quick computation of well-spread pareto-optimal solutions. In Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L., eds.: Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings. Volume 2632 of Lecture Notes in Computer Science., Springer (2003) 222–236

[137] Knowles, J.D., Corne, D.W., Fleischer, M.: Bounded archiving using the lebesgue measure. In Press, I., ed.: CEC – Congress on Evolutionary Computation. Volume 4., Canberra, Australia (2003)

[138] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In Fogel, D.B., El-Sharkawi, M.A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M., eds.: Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, IEEE Press (2002) 825–830

[139] Tenreiro Machado, J.A., Galhano, A.M.S.F.: A statistical perspective to the fourier analysis of mechanical manipulators. Journal Systems Analysis-Modelling-Simulation **33** (1998) 373–384

[140] Figueiredo, L., Tenreiro Machado, J.A., Ferreira, J.R.: Dynamical analysis of freeway traffic. IEEE Transactions on Intelligent Transportation Systems **5** (2004) 259–266

[141] Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., Trubian, M.: Heuristics from nature for hard combinatorial optimization problems. International Transactions on Operational Research **3** (1996) 1–21

[142] Birattari, M., Paquete, L., Stutzle, T., Varrentrapp, K.: Classification of metaheuristics and design of experiments for the analysis of components. Technical Report AIDA-2001-05, Fachgebiet Intellektik, Fachbereich Informatik, Technische UniversitatDarmstadt, Darmstadt, Germany (2001)

[143] Baluja, S., Davies, S.: Fast probabilistic modeling for combinatorial optimization. In: AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (1998) 469–476

[144] Michalewicz, Z., Fogel, D.B.: How to solve it: modern heuristics. Springer-Verlag New York, Inc., New York, NY, USA (2000)

[145] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT (2004)

[146] G., R.R., Zannoni, E., Posner, R.M.: Learning to understand software using cultural algorithms. In Sebald, A.V., Fogel, L.J., eds.: Proceedings of the Third Annual Conference on Evolutionary Programming, Singapore, World Scientific Press (1994) 150–157

[147] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, IV, Perth, Australia (1995) 1942–1948

[148] Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: Boa: The bayesian optimization algorithm. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 1., Orlando, Florida, USA, Morgan Kaufmann (1999) 525–532

[149] Zwillinger, D., Kokoska, S.: Standard Probability and Statistics Tables and Formulae. (Chapman & Hall CRC)

[150] Sandy, R.: Statistics for Business and Economics. Statistics Series. New York: McGraw-Hill Publishing Company (1990)

[151] Shao, J.: Mathematical Statistics. Springer-Verlag, New Yourk (1999)

[152] Wellek, S.: Testing Statistical Hypotheses of Equivalence. (A CRC Press company)

[153] Siegel, S., Castellan, N.: Nonparametric statistics for the behavioral sciences. 2nd ed. edn. McGraw-Hill, New York (1988)

[154] Miller, K.S., Ross, B.: An Introduction to the Fractional Calculus and Fractional Differential Equations. John Wiley and Sons (1993)

[155] Ross, B.: Fractional Calculus and its Applications, Lecture Notes in Mathematics 457. Springer-Verlag (1974)

[156] Oldham, K.B., Spanier, J.: The Fractional Calculus: Theory and Application of Differentiation and Integration to Arbitrary Order. Academic Press (1974)

[157] Samko, S.G., Kilbas, A.A., Marichev, O.I.: Fractional Integrals and Derivatives:Theory and Applications. Gordon and Breach Science Publishers (1993)

[158] Koh, C.G., Kelly, J.M.: Application of fractional derivatives to seismic analysis of base-isolated models. Earthquake Engineering and Structural Dynamics **19** (1990) 229–241

[159] Oustaloup, A.: La Commande CRONE: Commande Robuste d'Ordre Non Intier. Hermes (1991)

[160] Méhauté, A.L.: Fractal Geometries: Theory and Applications. Penton Press (1991)

[161] Gement, A.: On fractional differentials. Proc. Philosophical Magazine **25** (1938) 540–549

[162] Oustaloup, A.: La Dérivation Non Entier: Théorie, Synthèse et Applications. Editions Hermes (1995)

[163] Tenreiro Machado, J.A.: Analysis and design of fractional-order digital control systems. Journal System Analysis-Modelling-Simulation **27** (1997) 107–122

[164] Tenreiro Machado, J.A.: System modeling and control through fractional-order algorithms. FCAA – J. of Fractional Calculus & Ap. Analysis **4** (2001) 47–66

[165] Podlubny, I.: Fractional Diferential Equations. Academic Press, San Diego (1999)

[166] Vinagre, B.M., Petras, I., Podlubny, I., Chen, Y.Q.: Using fractional order adjustment rules and fractional order reference models in model-reference adaptive control. Nonlinear Dynamics **1-4** (2002) 269–279

[167] Torvik, P.J., Bagley, R.L.: On the appearance of the fractional derivative in the behaviour of real materials. ASME Journal of Applied Mechanics **51** (1984) 294–298

[168] Agrawal, O.P.: Solution for a fractional diffusion-wave equation in a bounded domain. Nonlinear Dynamics **29** (2002) 145–155

[169] Westerlund, S.: Dead Matter Has Memory! *Causal Consulting*. Kalmar, Sweden (2002)

[170] Anastasio, T.J.: The fractional-order dynamics of brainstem vestibulo-oculomotor neurons. Biological Cybernetics **72** (1994) 69–74

[171] Chen, Y., Moore, K.L.: Discretization schemes for fractional-order differentiators and integrators. IEEE Trans. On Circuits and Systems **49** (2002) 363–367

[172] Tenreiro Machado, J.A.: Analysis and design of fractional order digital control systems. SAMS – Journal Systems Analysis-Modelling-Simulation **27** (1997) 107–122

[173] Coello, C.A.C., Aguirre, A.H., Zitzler, E., eds.: Evolutionary Multi-Criterion Optimization, Third International Conference,EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings. In Coello, C.A.C., Aguirre, A.H., Zitzler, E., eds.: EMO. Volume 3410 of Lecture Notes in Computer Science., Springer (2005)

Índice remissivo