ALGORITMOS GENÉTICOS: APLICAÇÃO NA SÍNTESE DE ALGUNS ALGORITMOS DE CONTROLO

Anabela Maria Azevedo Oliveira Lopes

Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização de Automação e Sistemas

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2009

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Anabela Maria Azevedo Oliveira Lopes, Nº 1860183, 1860183@isep.ipp.pt
Orientação científica: Professor Doutor José António Tenreiro Machado, jtm@isep.ipp.pt

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto

1 de Julho de 2009

Dedico este trabalho às minhas filhas,
Catarina e Patrícia.

Agradecimentos

Ao longo deste trabalho, várias pessoas, de formas diferentes, contribuíram para a realização desta dissertação. Infelizmente, nenhum tipo de agradecimento é suficientemente expressivo para demonstrar o meu reconhecimento. A todas as pessoas que me ajudaram, fica aqui o meu muito obrigado. No entanto, algumas merecem um pouco mais de destaque.

Em primeiro lugar quero agradecer ao Professor Tenreiro Machado, não só pelo seu apoio, incentivo, e disponibilidade constantemente manifestados, mas também pela confiança em mim depositada guiando-me com o seu saber, experiência e competência profissional, ao longo da realização do trabalho, sempre na direcção correcta. Sem ele, a realização deste trabalho nunca teria sido possível.

Agradeço também aos colegas do ISEP que me apoiaram durante este trabalho, em particular à Prof. Alexandra Galhano que sempre teve uma palavra de incentivo, ao Luís Lima, e à Bertil Marques pelo apoio e disponibilidade que sempre demonstraram a me ajudar, no que fosse necessário, para eu ultrapassar mais esta etapa da minha vida académica.

Por último às pessoas mais importantes da minha vida, a minha família. Agradeço o carinho e a paciência das minhas filhas em abdicarem de muita da minha companhia e de compreenderem a minha falta de tempo para lhes dedicar toda a atenção que necessitam, mas que sempre me deram ânimo para levar este trabalho até ao fim. Ao meu marido Carlos que suportou a minha indisponibilidade para a família, mas que sempre me apoiou e incentivou na realização deste trabalho.

Resumo

Esta dissertação fornece uma visão global da computação evolutiva, nomeadamente dos algoritmos evolutivos e da inteligência dos enxames. De entre os algoritmos evolutivos dá-se um destaque especial aos algoritmos genéticos. Assim, apresentam-se os aspectos principais de construção e implementação dos algoritmos genéticos, os problemas teóricos e práticos e algumas das vantagens destes algoritmos relativamente a outros algoritmos de optimização e pesquisa.

Este trabalho inclui uma aplicação dos algoritmos genéticos ao cálculo fraccionário, mais concretamente à optimização de fracções racionais que constituem uma aproximação de derivadas fraccionárias no cálculo em tempo discreto. Inicialmente, faz-se uma análise às técnicas usuais baseadas em expansões por séries de Taylor e fracções de Padé. Numa segunda fase, o problema é reavaliado numa perspectiva de optimização aproveitando a flexibilidade dos algoritmos genéticos.

Palavras-Chave

Algoritmos Evolutivos, Inteligência dos Enxames, Algoritmos Genéticos, Optimização, Cálculo Fraccionário, Derivadas Fraccionárias.

Abstract

This thesis supplies a global vision of the evolutionary computation, with emphasis in evolutionary algorithms and swarm intelligence. Among the evolutionary algorithms a particular attention is given to the genetic algorithms. In this line of thought the main aspects of construction and implementation of genetic algorithms are presented. Also theoretical and practical problems, as well as some of the advantages of these algorithms are compared with other algorithms of search and optimization.

This work includes an application of the genetic algorithms to the fractional calculus, namely to the optimization of rational fraction approximations for the discrete time calculation of fractional derivatives. Initially, it is addressed the analysis to the standard techniques based on Taylor series and Padé fraction expansions. In a second phase, the problem is reevaluated in an optimization perspective by taking advantage of the flexibility of the genetic algorithms.

Keywords

Evolutionary Algorithms, Swarm Intelligence, Genetic Algorithms, Optimization, Fractional Calculus, Fractional Derivatives.

Résumé

Cette dissertation présente une vision globale du calcul évolutif, notamment des algorithmes évolutifs et de l'intelligence en essaim. Parmi les algorithmes évolutifs, les algorithmes génétiques sont analysés plus profondément. Les principaux détails de la construction et du développement des algorithmes génétiques, les questions théoriques aussi bien que pratiques, et la comparaison des avantages de ces algorithmes vis à vis d'autres algorithmes d'optimisation et recherche y sont présentés.

Ce travail inclut une application des algorithmes génétiques au calcul fractionnaire, plus précisément à l'optimisation de fractions rationnelles qui aboutissent d'une approximation de dérivés fractionnaires dans le calcul en temps discret. D'abords, une analyse aux techniques usuelles est développée, prenant forme d'expansion en séries de Taylor et fractions de Padé. Par la suite, le problème est réévalué dans une perspective d'optimisation, en profitant de la flexibilité des algorithmes génétiques.

Mots-clés

Algorithmes Évolutifs, Intelligence en essaim, Algorithmes Génétiques, Optimisation, Calcul Fractionnaire, Dérivées Fractionnaires.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AE – Algoritmo Evolutivo

AG – Algoritmo Genético

AGS Algoritmo Genético Simples

CE – Computação Evolutiva

CF – Cálculo Fraccionário

DF – Derivada Fraccionária

DIF – Derivada e Integral Fraccionário

EE – Estratégia Evolutiva

f – Função de Aptidão

fav – Função de Avaliação

IA – Inteligência Artificial

IE – Inteligência dos Enxames

$o(H)$ – Ordem de um esquema

OCF – Optimização por Colónias de Formigas

OEP – Optimização por Enxames de Partículas

OFB – Optimização através de um Forragear Bacteriano

p_c – Probabilidade de cruzamento

PCV – Problema do Caixeiro Viajante

p_d – Probabilidade de destruição

PE – Programação Evolutiva

PG – Programação Genética

p_m – Probabilidade de mutação

p_r – Probabilidade de reprodução

p_s – Probabilidade de sobrevivência

TF – Transformada de Fourier

TL – Transformada de Laplace

$\delta(H)$ – Comprimento de um esquema

1. INTRODUÇÃO

Este capítulo tem como objectivo enquadrar o tema desta dissertação de modo a fornecer uma visão breve e geral da tese de Mestrado "Algoritmos Genéticos: Aplicação na Síntese de alguns Algoritmos de Controlo".

O capítulo está estruturado da seguinte forma: a secção 1.1 começa por fazer o enquadramento dos algoritmos genéticos e do cálculo fraccionário. A secção 1.2 indica os objectivos a atingir com a realização deste trabalho. Por último, a secção 1.3 apresenta a descrição da estrutura deste relatório.

1.1. ÂMBITO DO TRABALHO

1.1.1. ALGORITMOS GENÉTICOS

Nas últimas décadas têm sido propostos algoritmos inspirados em certos processos biológicos, dando origem à área da Computação Evolutiva (CE). A CE abrange vários tipos de algoritmos, sendo os Algoritmos Genéticos (AG) um dos seus principais ramos. Os AGs são, fundamentalmente, métodos de pesquisa e optimização inspirados na genética e evolução dos seres vivos. Um AG actua sobre um conjunto de potenciais soluções do problema que se pretende resolver. Inicialmente, essas potenciais soluções são escolhidas ao acaso e, como seria de esperar, na maior parte dos casos não resolvem o problema adequadamente. Entretanto, no decorrer da execução do AG, estas soluções são avaliadas,

alteradas e combinadas por meio de operações de selecção, cruzamento e mutação. A cada iteração do AG, estas operações produzem novas soluções, que tendem a ser melhores do que as suas antecessoras. Desta forma, são geradas soluções cada vez melhores até que se encontre uma ou mais soluções suficientemente adequadas para o problema.

Em muitos casos os AGs podem ser uma interessante alternativa aos métodos tradicionais de pesquisa e optimização. Em primeiro lugar, os conceitos, processos e estruturas computacionais associados aos AGs são mais simples, e mais fáceis de implementar, do que a maioria dos métodos tradicionais. Em segundo lugar, os AGs são mais flexíveis que os métodos tradicionais, podendo ser utilizados para resolver, virtualmente, qualquer problema de pesquisa e optimização. Um problema associado aos AGs é o seu custo computacional, que pode ser muito elevado, dependendo dos procedimentos que são necessários para fazer a avaliação das potenciais soluções.

Os AGs foram introduzidos na década de sessenta com os trabalhos de John Holland (Holland 1962) e posteriormente desenvolvidos por Goldberg (Goldberg 1989). Nos nossos dias existem publicações especializadas, muitas aplicações propostas e um elevado número de investigadores trabalhando nessa área.

1.1.2. CÁLCULO FRACCIONÁRIO

O conceito de Cálculo Fraccionário (CF) remonta ao início da teoria do cálculo integral e diferencial de ordem inteira e aborda a generalização para valores não inteiros (reais ou complexos) (Oldham e Spanier 1974), (Samko, Kilbas e Marichev 1993), (Miller e Ross 1993) e (Podlubny 1999). No entanto, a complexidade acrescida desta teoria, levou a que só nos últimos anos tenha existido um desenvolvimento acentuado em aplicações nas ciências de engenharia, nomeadamente na modelação e no controlo. De facto, assiste-se actualmente a um interesse crescente na sua aplicação e no desenvolvimento de um conjunto de estudos nas mais diversas áreas científicas, tais como a viscoelasticidade, biologia, electrónica, processamento de sinais, difusão e propagação de ondas, entre outros.

Em relação à área de controlo de sistemas, só nas últimas décadas é que a teoria do CF encontrou as suas primeiras aplicações. No entanto, a sua utilização nesta área tem revelado elevados desempenhos em relação às estratégias tradicionais, fazendo com que seja, actualmente, uma das áreas com mais aplicações da teoria do CF.

A aplicação dos conceitos dos AGs na área do controlo de sistemas fraccionários está numa fase embrionária, existindo um vasto campo de investigação e de desenvolvimento de novos algoritmos.

1.2. OBJECTIVOS

O objectivo deste trabalho é, numa primeira fase, fazer um estudo do funcionamento dos algoritmos genéticos. Numa segunda fase é proposta uma nova metodologia para a optimização de aproximações através de fracções racionais para o cálculo de derivadas fraccionárias, em tempo discreto, utilizando os AGs.

1.3. ORGANIZAÇÃO DO RELATÓRIO

Este trabalho está organizado em cinco capítulos e um apêndice.

No presente capítulo apresenta-se uma breve introdução ao trabalho e aos seus objectivos.

O capítulo 2 aborda a área da computação evolutiva, fazendo uma descrição sumária de alguns dos métodos computacionais existentes.

No capítulo 3 é realizada uma análise detalhada do funcionamento dos AGs, introduzindo os principais conceitos e descrevendo os detalhes da sua implementação. Neste capítulo apresenta-se ainda uma introdução da selecção natural, refere-se as analogias destes algoritmos com a biologia e indicam-se algumas das vantagens dos AGs em relação a outros algoritmos de optimização e pesquisa.

No capítulo 4 é introduzido o cálculo da derivada fraccionária, enunciado o problema de aproximação por fracções racionais através de algoritmos genéticos e são desenvolvidas várias experiências que demonstram a eficácia do método proposto.

O capítulo 5 apresenta as principais conclusões que decorrem do trabalho efectuado, e discute possíveis aspectos de investigação futura.

O apêndice A faz referência aos aspectos fundamentais do cálculo fraccionário.

2. COMPUTAÇÃO EVOLUTIVA

2.1. INTRODUÇÃO

A Computação Evolutiva (CE) é uma área da Inteligência Artificial (IA) que engloba um conjunto de métodos computacionais inspirados nos mecanismos biológicos encontrados na natureza, tais como a evolução, a mutação, a selecção natural e o cruzamento. A vantagem mais significativa da CE reside na possibilidade de resolver problemas através de uma descrição matemática simples da solução pretendida, não havendo necessidade de se indicar explicitamente os passos até ao resultado. Alguns dos casos mais importantes da CE são os Algoritmos Evolutivos (AEs) e a Inteligência dos Enxames (*Swarm Intelligence*).

Os primeiros trabalhos envolvendo AEs datam da década de 1930, quando os sistemas evolutivos naturais passaram a ser investigados como algoritmos de exploração de uma função objectivo. Porém, apenas com um acesso mais fácil a computadores é que se intensificaram os desenvolvimentos de AEs, a partir da década de 1960, com a realização de diversos estudos teóricos e práticos. Nesse contexto, foram desenvolvidas de forma independente três abordagens de AEs: a programação evolutiva (Fogel 1962), as estratégias evolutivas (Rechenberg 1965) e os algoritmos genéticos (Holland 1962). Mais tarde apareceram os sistemas classificadores (Booker, Goldberg e Holland 1989) e a programação genética (Koza 1992).

Relativamente ao grupo de Inteligência dos Enxames podemos destacar os algoritmos de optimização por colónias de formigas (*Ant Colony Optimization*), inicialmente propostos por Marco Dorigo em 1992, os algoritmos de optimização por enxame de partículas (*Particle Swarm Optimization*) descritos por Kennedy e Eberhart em 1995 e os algoritmos de optimização através de um forragear bacteriano (*Optimization Foraging Bacterial*) introduzidos por Passino em 2002.

Neste capítulo é feita uma abordagem aos principais AEs e a alguns algoritmos de Inteligência dos Enxames. Assim, na secção 2.2 iremos apresentar os AEs canónicos, nomeadamente a Programação Evolutiva e as Estratégias Evolutivas. A Inteligência dos Enxames encontra-se na secção 2.3, onde se faz uma introdução aos algoritmos: Optimização por Colónia de Formigas, Optimização por Enxame de Partículas e Optimização através de um Forragear Bacteriano.

2.2. ALGORITMOS EVOLUTIVOS

Como já foi referido, ramos distintos de pesquisa desenvolveram técnicas computacionais que deram origem aos chamados AEs. A programação evolutiva, as estratégias evolutivas e os algoritmos genéticos foram fundamentais para o desenvolvimento desta área de pesquisa e foram denominados AEs canónicos. As principais características dos AEs canónicos são importantes e necessárias para o entendimento de propostas de solução baseadas em conceitos evolutivos, bem como para o desenvolvimento de novos AEs. Nas subsecções seguintes descreve-se sucintamente os AEs canónicos, com excepção dos Algoritmos Genéticos que irão ser abordados separadamente no capítulo 3.

2.2.1. PROGRAMAÇÃO EVOLUTIVA

A Programação Evolutiva (PE) foi proposta por Fogel em 1962, com o objectivo de utilizar os conceitos de evolução no desenvolvimento da IA. Ainda que a proposta original tratasse de predição de comportamento de máquinas de estados finitos, a PE adapta-se facilmente a qualquer estrutura de problema. Na PE a reprodução é feita apenas por operadores de mutação, sendo que todos os indivíduos da população actual geram novos descendentes. Este processo caracteriza a chamada reprodução assexuada. Na selecção de indivíduos para a geração seguinte, os descendentes competem com os pais e somente os indivíduos com maior aptidão sobrevivem. Na figura 2.1 podemos ver um exemplo de um algoritmo de PE, onde P(t) é a população na geração t.

```
Início
t = 0
inicializar aleatoriamente P(t)
avaliação P(t)
repetir
  mutação P(t)
  avaliação P(t)
  seleccionar P(t + 1) a partir de P(t)
  t = t + 1
até condição de conclusão verificada
fim do algoritmo
```

Figura 2.1 Exemplo de algoritmo de Programação Evolutiva

2.2.2. ESTRATÉGIAS EVOLUTIVAS

As Estratégias Evolutivas (EEs) foram desenvolvidas com o objectivo de solucionar problemas de optimização de parâmetros. As EEs foram propostas originalmente por Rechenberg em 1965, que desenvolveu a estratégia com dois membros (1 + 1)-EE. Neste modelo a população é constituída apenas por um indivíduo e só é utilizado o operador de mutação, isto é, um pai gera um único descendente por geração através da mutação e ambos competem pela sobrevivência.

Nas EEs, cada gene existente no cromossoma descreve um parâmetro do problema, sendo que o alelo é representado em vírgula flutuante. Os genes são compostos por dois valores: um representa o valor do parâmetro e o outro indica o desvio padrão desse parâmetro. A geração de um novo indivíduo é feita por meio da aplicação de um operador de mutação, com distribuição de probabilidade Gaussiana, com média zero e com desvio padrão do gene correspondente no pai.

Mais tarde, esta teoria evoluiu para a estratégia multimembros, (μ, 1)-EE, na qual uma população de μ indivíduos se recombina de modo aleatório para formar um descendente, o qual, após sofrer mutação, substitui (se for o caso) o pior elemento da população. Ainda que esta estratégia nunca tenha sido largamente utilizada, ela permitiu a transição para as estratégias ($\mu + \lambda$)-EE e (μ, λ)-EE, já no final dos anos 70. Na estratégia ($\mu + \lambda$)-EE, a partir da população de μ indivíduos, utilizando operações de cruzamento e mutação, são gerados $\lambda \geq \mu$ descendentes. Sobrevivem apenas μ indivíduos seleccionados entre os μ indivíduos actuais e os λ novos indivíduos gerados. Na (μ, λ)-EE, também são gerados $\lambda \geq \mu$ descendentes a partir da população de μ indivíduos. Não existe competição entre os μ

indivíduos já existentes com os λ novos indivíduos, pois os μ pais são eliminados e, dos λ descendentes, apenas são escolhidos os μ mais aptos. Na figura 2.2 podemos ver um exemplo de um algoritmo de EE, onde P(t) é a população na geração t, μ é o número de indivíduos da população P e λ é o número de descendentes.

```
Início
t = 0
inicializar P(t)com μ indivíduos
avaliação P(t)
enquanto condição de conclusão não verificada
  repetir λ vezes,
     cruzamento P(t)
     mutação P(t)
     avaliação P(t)
  fim de repetir
  t =t + 1
  seleccionar os μ melhores indivíduos de λ ou μ+λ
fim do enquanto
fim do algoritmo
```

Figura 2.2 Exemplo de algoritmo de uma Estratégia Evolutiva

2.3. INTELIGÊNCIA DOS ENXAMES

A Inteligência dos Enxames (IE) é uma técnica de IA que foi desenvolvida a partir do estudo do comportamento colectivo auto-organizado de sistemas descentralizados, constituídos por um número elevado de agentes, com o objectivo de realizar uma determinada tarefa.

O termo "enxame" é utilizado de forma genérica para se referir a qualquer colecção estruturada de agentes capazes de interagir. O exemplo clássico de um enxame é um enxame de abelhas. Entretanto, a metáfora de um enxame pode ser estendida a outros sistemas com uma arquitectura similar. Uma colónia de formigas pode ser vista como um enxame, onde os agentes são formigas; um bando de pássaros é um enxame, onde os agentes são pássaros; um cardume de peixes é um enxame, onde os agentes são os peixes, um engarrafamento é um enxame, onde os agentes são carros; uma multidão é um enxame de pessoas, um sistema imunológico é um enxame de células e moléculas, e uma economia é um enxame de agentes económicos. Embora a noção de enxame sugira um aspecto de

movimento colectivo no espaço, como num 'enxame de pássaros', estamos interessados em todos os tipos de comportamentos colectivos e não apenas no movimento espacial.

Algumas propriedades da inteligência colectiva:

- Proximidade: os agentes devem ser capazes de interagir;
- Qualidade: os agentes devem ser capazes de avaliar seus comportamentos;
- Diversidade: permite ao sistema reagir a situações inesperadas;
- Estabilidade: nem todas as variações ambientais devem afectar o comportamento de um agente;
- Adaptabilidade: capacidade de se adequar a variações ambientais.

Sendo assim, um sistema de enxame é composto por um conjunto de agentes capazes de interagir entre si e com o meio ambiente

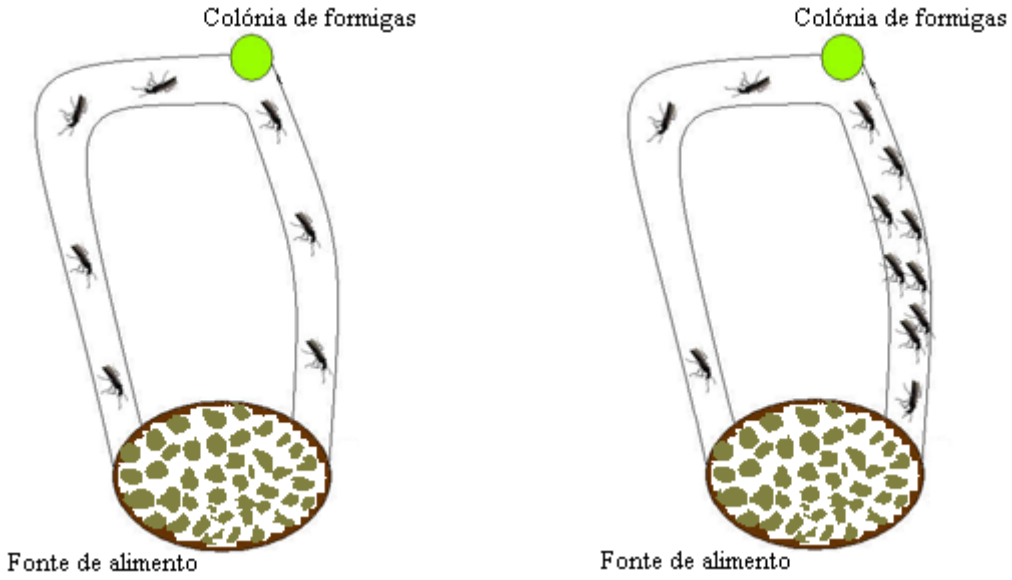
A IE já é utilizada por empresas para aperfeiçoar a eficiência e em aplicações militares, nomeadamente no desenvolvimento de robots terrestres e aéreos. No entanto, a IE poderá melhorar a gestão de outros casos, como sejam os transportes públicos, o movimento nos aeroportos e a orientação de aviões em situações de mau tempo.

2.3.1. OPTIMIZAÇÃO POR COLÓNIAS DE FORMIGAS

A Optimização por Colónias de Formigas (OCF), do inglês *Ant Colony Optimization*, é um algoritmo de optimização heurística baseado no comportamento de uma colónia de formigas. Proposto inicialmente por Marco Dorigo (Dorigo 1992), o primeiro algoritmo tinha como objectivo procurar o melhor caminho num grafo sendo inspirado no comportamento das formigas que procuram um caminho entre a sua colónia e uma fonte de alimento.

No mundo real as formigas andam aparentemente sem rumo até que quando encontram comida, elas retornam à colónia deixando um rasto de uma substância química denominada feromônio. Se outras formigas encontram um caminho previamente percorrido detectam o rasto de feromônio e então podem decidir, com uma alta probabilidade, segui-lo, reforçando assim o caminho com o seu próprio feromônio. Com o decorrer do tempo, os

rastos de feromônio começam a evaporar-se, reduzindo sua força atractiva. Quanto mais formigas passarem por um caminho predeterminado, mais tempo demorará o feromônio do caminho a evaporar. A evaporação do feromônio possui a vantagem de evitar a convergência para uma solução local óptima: se não houvesse a evaporação, os caminhos escolhidos pelas primeiras formigas tornar-se-iam excessivamente atractivos para as outras e, neste caso, a exploração do espaço da solução tornava-se limitado. Quando toda a comida é recolhida, o feromônio deixa de ser renovado, e ao fim de algum tempo desaparece. Deste modo, as formigas retomarão o seu rumo aleatório. Na figura 2.3 podemos ver o comportamento das formigas procurando o caminho mais curto entre o formigueiro e uma fonte de alimento.



(a) (b)

Figura 2.3 Comportamento das formigas na escolha do caminho mais curto entre o formigueiro e uma fonte de alimento: (a) inicialmente ambos os caminhos são igualmente prováveis; (b) no final as formigas escolhem o caminho mais curto

A ideia do OCF consiste em imitar este comportamento através de formigas artificiais que caminham por um grafo, que por sua vez, representa o problema a ser resolvido.

Semelhanças entre OCF e o sistema biológico:

- ambos os sistemas são constituídos por múltiplos agentes cooperando entre si;

- ambos os sistemas utilizam um factor de cooperação, através do qual ocorre a sinergia entre os agentes (feromônio). O feromônio representa a informação colectiva e é essencial no desenvolvimento de ambos os sistemas. Verifica-se também a evaporação do feromônio, o que permite às formigas explorarem novos caminhos;

- os agentes dividem a mesma função em ambos os sistemas: procurar o menor caminho entre uma origem (colónia) e um destino (alimento).

Diferenças entre o OCF e o sistema biológico:

- as formigas artificiais possuem movimentação discreta, sendo que os seus movimentos têm origens e destinos discretos;

- nas formigas artificiais existe um estado interno ou memória, para que não haja sobreposição de movimentos;

- no mundo artificial ocorre um depósito de feromônio com base na qualidade da solução encontrada;

- as formigas artificiais deixam o feromônio em cada ponto visitado após chegar ao destino. Na vida real as formigas deixam o feromônio durante o movimento e não após chegar ao seu destino.

2.3.2. OPTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

A Optimização por Enxame de Partículas (OEP), traduzido do inglês *Particle Swarm Optimization*, é um algoritmo de optimização estocástico proposto por Eberhart e Kennedy (Kennedy e Eberhart 1995). A OEP é uma forma de IE onde é simulado o comportamento de um sistema social biológico tal como um bando de pássaros (figura 2.4), ou um cardume de peixes.

Figura 2.4 Bando de pássaros

Quando um enxame procura o alimento, os seus membros espalham-se numa determinada região. Tendo em vista facilitar a exploração do espaço, tipicamente cada indivíduo deve ter um certo nível de aleatoriedade no seu movimento, a fim de que o movimento do enxame tenha uma certa capacidade exploratória. Por outras palavras, cada indivíduo deve ser influenciado pelos restantes elementos do enxame, mas, por outro lado, deve explorar independentemente até uma certa extensão. Assim, mais cedo ou mais tarde, um deles encontrará algum alimento e, sendo social, anuncia isso aos restantes membros do enxame. Estes elementos podem então, também aproximar-se da fonte do alimento.

A OEP é apropriada para lidar com problemas onde a melhor solução pode ser representada como um ponto, ou como uma superfície, num espaço multidimensional. Assim, este tipo de problemas é modelado por partículas (membros do enxame) que se encontram no espaço de soluções do problema, que têm uma posição e uma velocidade, e que são avaliadas a cada iteração do algoritmo de acordo com uma função de avaliação. Cada partícula é inicializada com posição e velocidade aleatórias, e guarda consigo a informação sobre a melhor posição por ela já visitada. Além disso, o algoritmo guarda a melhor posição já alcançada pelo enxame. Em cada iteração as partículas são avaliadas e as melhores posições de cada partícula e do enxame são actualizadas. A velocidade de cada partícula é ajustada levando em conta a influência da melhor posição já atingida pelo enxame (influência social) e a melhor posição já atingida pela própria partícula (decisão individual). A figura 2.5 apresenta um exemplo de pseudocódigo de um algoritmo OEP.

```
Início
t = 0
inicializar aleatoriamente P(t)
repetir
  avaliação P(t)
  modificar a velocidade das partículas com base
  na melhor posição da cada partícula e na melhor
  posição do enxame até ao momento
  t = t + 1
até condição de conclusão verificada
fim do algoritmo
```

Figura 2.5 Exemplo de pseudocódigo de um algoritmo OEP

Deste modo, os membros do enxame movem-se pelo espaço de busca procurando atingir a solução óptima com base na decisão individual e na influência social.

2.3.3. OPTIMIZAÇÃO ATRAVÉS DE UM FORRAGEAR BACTERIANO

As bactérias movem-se na busca de nutrientes e afastam-se de elementos nocivos, este método biológico é conhecido como forragear (*foraging*). O algoritmo de Optimização através de um Forragear Bacteriano (OFB), do inglês *Optimization Foraging Bacterial*, foi introduzido por Passino (Passino 2002) e modela o comportamento de forragear das bactérias *Escherichia Coli* (*E. Coli*) que normalmente estão presentes no intestino dos seres humanos. As bactérias *E. Coli* têm uma estratégia de forragear que consiste em quatro processos: quimiotaxia, enxameação, reprodução, e eliminação-dispersão, conforme descrito de seguida.

A bactéria *E. Coli* move-se numa alternância entre deslocamentos contínuos unidireccionais (*swim*), ver figura 2.6 (a), e mudanças aleatórias de direcção (*tumble*), ver figura 2.6 (b). Uma bactéria deste tipo desloca-se durante um certo período de tempo em linha recta, até que pára, muda de direcção de forma aleatória, e continua o seu movimento unidireccional. A *E. Coli* consegue, desta forma, mover-se e encontrar zonas de maior concentração de nutrientes.

(a) (b)

Figura 2.6 Quimiotaxia: (a) Deslocamento em linha recta (*swim*); (b) mudança aleatória de direcção (*tumble*)

A bactéria possui sensores que detectam a presença de nutrientes, alterando a velocidade e a duração dos movimentos contínuos bem como o número de mudanças de direcção num meio mais nutritivo. Através da comparação de valores de concentração de nutrientes medidos pelos sensores em instantes diferentes durante o movimento unidireccional, a bactéria consegue determinar se a concentração está a diminuir ou a aumentar:

- caso esteja a aumentar, a *E. Coli* aumenta o tempo em que se desloca em linha recta;

- caso a concentração esteja a diminuir, a bactéria diminui o tempo em que se desloca em linha recta. Assim, irá mudar de direcção mais cedo, na "esperança" de encontrar mais nutrientes noutra direcção.

Note-se que a mudança de direcção é sempre aleatória e que a bactéria não tem qualquer noção onde se encontram os nutrientes. A única acção que a bactéria executa consiste em influenciar a sua marcha aleatória, sendo isso suficiente para que a bactéria atinja regiões com concentrações elevadas de nutrientes. Este processo, conhecido como quimiotaxia (*chemotaxis)*, é realizado durante todo o tempo de vida das bactérias.

As bactérias são capazes de comunicar entre si. No caso das bactérias *E. Coli*, a comunicação é efectuada pela secreção de substâncias químicas, que alertam as outras bactérias, quer tenham encontrado elevada concentração de nutrientes, quer as condições encontradas não sejam apropriadas. Deste modo, se uma bactéria encontrou o melhor

trajecto para uma grande concentração de nutrientes, deve tentar atrair outras bactérias de modo a que estas alcancem o lugar desejado mais rapidamente. A enxameação faz com que as bactérias formem conjuntos e depois se movimentem em grupo de forma concêntrica e com densidade bacteriana elevada.

Na reprodução, as bactérias menos saudáveis morrem, enquanto que outras bactérias mais saudáveis se dividem, dando origem cada uma a duas novas bactérias, que substituem, na mesma posição, a bactéria que lhes deu origem. A saúde das bactérias é medida pela quantidade de nutrientes que conseguiu durante sua vida de forragear. Assim, quanto menor o número de nutrientes menos saudável é a bactéria e menor é a probabilidade que se reproduza.

É possível que, no ambiente local, a vida de uma população de bactérias sofra mudanças graduais devido ao consumo de nutrientes, ou mudanças súbitas devido a outras influências. As mudanças podem ocorrer de modo a que todas as bactérias numa dada região sejam mortas, ou que um grupo seja dispersado numa parte nova do ambiente. Por exemplo, um aumento local e significativo de temperatura pode matar uma população de bactérias que se encontra numa região com uma elevada concentração de nutrientes. Alternativamente, pode verificar-se que a acção do vento disperse uma população de bactérias para novas posições. Este processo é conhecido como eliminação-dispersão.

Antes de aplicar o algoritmo OFB a um determinado problema deve-se especificar qual o tempo de vida das bactérias, isto é, deve-se definir o número de etapas quimiotaxicas (N_q), o número de etapas de reprodução (N_r), o número de etapas de eliminação-dispersão (N_{ed}) e a probabilidade (p_{ed}), o comprimento máximo do movimento em linha recta (*swim*), e o número máximo de movimentos em linha recta permissíveis num determinado sentido (N_s). De modo a que o tamanho da população se mantenha constante, o número de bactérias que morrem deve ser igual ao número de bactérias que se reproduzem. Em relação ao processo de eliminação-dispersão, se uma bactéria for eliminada da população, então deve ser substituída por uma nova bactéria situada numa posição aleatória dentro do domínio da optimização. O processo de enxameação pode ser utilizado, ou não, no algoritmo OFB. A figura 2.7 representa um algoritmo de OFB, sem enxameação, onde a população é constituída por N bactérias.

```
Início
inicializar aleatoriamente população
calcular a saúde de cada Bacteria i
 ciclo para Eliminação-dispersão l
   ciclo para Reprodução k
      ciclo para Quimiotaxia j
         ciclo para Bactéria i
            mudança aleatória de direcção (tumble)
            movimento unidireccional (swim)
            repetir
               movimento unidireccional (swim)
            até atingir Ns
            i=i+1
        fim do ciclo Bactéria se i=N
        j=j+1
      fim do ciclo Quimiotaxia se j=Nq
      calcular a saúde de cada Bacteria i
      ordenar população por ordem descendente de saúde
      S/2 bactérias menos saudáveis morrem e as restantes S/2
      dividem-se em duas
      k=k+1
   fim do ciclo Reprodução se k=Nr
   eliminar e dispersar cada Bactéria com probabilidade ped
   l=l+1
 fim do ciclo Eliminação-dispersão se l=Ned
fim do algoritmo
```

Figura 2.7 Exemplo de um algoritmo de OFB

3. ALGORITMOS GENÉTICOS

3.1. INTRODUÇÃO

Os Algoritmos Genéticos (AG) são o ramo mais conhecido da CE que se encontra em crescimento, na área da IA. Os AGs são métodos adaptativos, que podem ser utilizados para resolver problemas de pesquisa e de optimização, baseados no princípio evolutivo da selecção natural.

Estes algoritmos foram introduzidos por John Holland no início dos anos sessenta, sendo, de seguida, desenvolvidos em conjunto com os seus alunos e colegas da Universidade de Michigan nos anos sessenta e setenta. A pesquisa inicial de Holland não tinha como objectivo principal o desenvolvimento de um algoritmo para resolução de problemas, mas, sobretudo, estudar os fenómenos relacionados com a adaptação das espécies e a selecção natural que ocorre na natureza, bem como desenvolver modelos onde os mecanismos de adaptação natural pudessem ser incorporados nos sistemas computacionais (Mitchell 1997). Deste estudo resultou a obra *Adaption in Natural and Artificial Systems* (Holland 1975), publicada em 1975. Nos anos 80, David E. Goldberg, um dos alunos de Holland, consegue resolver um problema complexo envolvendo o controlo de transmissão de gás em gasodutos através de AGs (Goldberg 1983). Em 1989, Goldberg edita *Genetic algorithms for search, optimization, and machine learning* (Goldberg 1989), que ainda hoje é considerado um dos livros mais importantes sobre Algoritmos Genéticos. Desde então, a área de AGs tem evoluído constantemente e, actualmente, utilizam-se AGs em

várias áreas de conhecimento como, por exemplo, medicina, economia, engenharia, biologia ou na música (Cagnoni, et al. 1999), (Vinterbo e Ohno-Machado 2000), (Smigrodzki, et al. 2005), (J. R. Oliver 1993), (Yuret e Maza 1994), (Bras 2002), (Davidor 1991), (McDonnell, et al. 1992), (Page, McDonnell e Andersen 1992), (Schaffer e Eshelman 1993), (Lohn, et al. 2000), (Handley 1993), (Punch, et al. 1993), (Jacob 1995), (Ganeshan e Pickard 2004) e (Romero e Machado 2007).

Neste capítulo analisaremos detalhadamente o funcionamento dos AGs. Nas secções 3.2 e 3.3 é feita uma introdução da selecção natural e é estabelecida a correspondência entre a terminologia utilizada em Biologia e nos AGs. De seguida, na secção 3.4 são apresentados os aspectos principais da construção e implementação dos AGs, incluindo os parâmetros, operadores genéticos e estratégias para melhoria do desempenho do algoritmo. Na secção 3.5, são abordados alguns problemas teóricos e práticos que os AGs apresentam. Por último, na secção 3.6 são referidas algumas das vantagens dos AGs relativamente a outros algoritmos de optimização e pesquisa.

3.2. TEORIA DA SELECÇÃO NATURAL

A ideia básica da selecção natural foi apresentada por Charles Darwin (1809-1882) e representa uma das maiores conquistas da ciência. Embora a teoria de Darwin tenha prevalecido e seja aceite como verdadeira até hoje, não foi ele o primeiro a defender uma teoria evolutiva. As origens do pensamento evolutivo, onde a ideia de evolução é apresentada pela primeira vez, em oposição ao criacionismo, são atribuídas a Jean-Baptiste de Lamarck (1744-1829), naturalista francês. A sua teoria foi apresentada em 1809, num livro intitulado *Philosophie Zoologic*. A teoria de Lamarck, conhecida até hoje como *Lamarckismo*, tinha como base duas leis fundamentais:

- Lei do uso e desuso: o uso de determinadas partes do corpo do organismo faz com que estas se desenvolvam, e o desuso faz com que se atrofiem.

- Lei da transmissão dos caracteres adquiridos: alterações provocadas em determinadas características do organismo, pelo uso e desuso, são transmitidas aos descendentes.

De acordo com Lamarck as novas espécies aparecem, por evolução, devido a aquisição ou perda de caracteres. A primeira suposição da Lamarck está correcta, o uso e o desuso de

determinadas partes do corpo provocam alteração nos organismos. Todos sabemos que os atletas desenvolvem os seus músculos através do uso, enquanto que a paralisação das pernas, por exemplo, dá origem a atrofia. A falha está na segunda hipótese, os caracteres adquiridos por uso e desuso nunca são transmitidos aos seus descendentes.

As ideias de Lamarck, embora estivessem incorrectas, deram início a um movimento de estudos e experiências, começando as ideias sobre a evolução a ser difundidas e discutidas.

No ano de 1859, o cientista inglês Charles Darwin publica o seu livro *On the Origin of Species by means of Natural Selection*. Nesta obra ele apresenta a sua teoria completa, na qual concluiu que todos os organismos que nascem nem sempre apresentam condições de sobrevivência. Apenas sobrevivem os que têm maior capacidade de se adaptarem às condições ambientais, e eles reproduzem-se deixando descendentes férteis. Nessas circunstâncias as variações favoráveis tenderiam a ser preservadas e as desfavoráveis a serem destruídas.

O mecanismo de evolução proposto por Darwin pode ser resumido em seis etapas:

- Os indivíduos de uma mesma espécie mostram muitas variações na forma e na fisiologia;
- Boa parte dessas variações são transmitidas aos descendentes;
- Se todos os indivíduos de uma espécie se reproduzissem, as populações cresceriam exponencialmente;
- Como os recursos naturais são limitados, os indivíduos de uma população lutam pela sua sobrevivência e pela sobrevivência da sua descendência;
- Em consequência, somente alguns, os mais aptos, sobrevivem e deixam descendentes. A sobrevivência e a reprodução dependem das características desses indivíduos que, por serem hereditárias, serão transmitidas aos seus filhos;
- Através dessa selecção natural, as espécies serão representadas por indivíduos cada vez mais adaptados.

O principal problema do *darwinismo* foi a falta de uma teoria satisfatória que explicasse a origem e a transmissão das variações.

Darwin não conseguiu responder adequadamente a estas críticas e, somente em 1865 esses problemas puderam ser resolvidos graças a Gregor Mendel, que apresentou o seu trabalho sobre as leis da hereditariedade, hoje designadas por Leis de Mendel e que regem a transmissão dos caracteres hereditários. Apesar disso, não há dúvida de que a teoria moderna da evolução deve mais a Darwin do que a qualquer outro cientista, e o seu conceito de selecção natural continua válido até hoje.

3.3. BIOLOGIA E TERMINOLOGIA

Assim como acontece no meio ambiente, num AG existe um grupo de indivíduos que competem entre si para garantirem a própria sobrevivência e para assegurar que as suas características sejam passadas para os seus descendentes. Nessa analogia, cada indivíduo do AG é, de facto, uma solução candidata para o problema em questão, que, por sua vez, faz o papel do próprio meio ambiente, já que estabelece os critérios que permitem avaliar se um indivíduo/solução é adaptado, ou não.

As melhores soluções contam com uma maior probabilidade de "sobreviver" e, através de operadores genéticos, permitem gerar outras soluções de boa qualidade. Da combinação de bons indivíduos podem surgir outros ainda melhores, que tenham herdado as boas características daqueles que lhes deram origem. Cada iteração do processo é denominada de geração. Após várias gerações, existe uma grande probabilidade de que a população tenha aumentado a sua qualidade média, isto é, que seja composta por indivíduos mais bem adaptados ao ambiente em questão, neste caso o problema em estudo.

Muitos dos termos utilizados nos AGs são originários da teoria da evolução natural e da genética. Para melhor compreensão destes termos, é feita a comparação entre a terminologia dos AGs e o processo biológico.

Na natureza, o genótipo de um indivíduo é definido por um conjunto de cromossomas. A espécie humana, por exemplo, apresenta 23 pares de cromossomas. Nos AGs, cada solução candidata é codificada num único cromossoma, geralmente consistindo num vector que contém elementos de um alfabeto pré-definido.

Os cromossomas são interpretados como uma sequência de genes. Um gene é a parte de um cromossoma que codifica uma característica específica do fenótipo. O fenótipo consiste no conjunto de características observáveis de um indivíduo como, por exemplo:

morfologia, desenvolvimento e comportamento. O fenótipo resulta da expressão dos genes do indivíduo, da influência de factores ambientais e da possível interacção entre os dois. Por exemplo, no caso da cor dos olhos, o fenótipo azul é condicionado pelo genótipo 'aa', e o fenótipo castanho pode ser condicionada pelo genótipo 'AA' ou pelo genótipo 'Aa'. Neste exemplo estamos representando por 'A' o gene que condiciona cor castanha (dominante), e por 'a', o gene que condiciona cor azul (recessivo). Tal como na natureza, nos AGs as operações não são realizadas directamente nas soluções candidatas, mas sobre uma codificação das mesmas.

Os valores que um gene pode assumir são denominados alelos. Assim, para a cor dos olhos os alelos são preto, castanho, azul e verde. O conjunto de genes que define a estrutura do cromossoma é chamado de genoma. O genoma pode ser visto como um "filtro" através do qual o cromossoma é interpretado. Dois cromossomas idênticos podem representar fenótipos completamente distintos, dependendo da definição do genoma. A posição de cada gene num genoma é designada de *locus* (lugar). Nos AGs, uma definição racional da posição dos genes no cromossoma pode melhorar consideravelmente o desempenho do processo.

A tabela 3.1 apresenta um resumo da correspondência entre os termos utilizados em Biologia e nos Algoritmos Genéticos.

Tabela 3.1 Terminologia utilizada em Biologia/Algoritmos Genéticos

Termo em Biologia	**Termo nos AGs**
Cromossoma, indivíduo	Vector
Gene	Parâmetro, característica
Alelo	Valor
Lugar (*locus*)	Posição no vector
Genótipo	Estrutura de codificação
Fenótipo	Estrutura de descodificação
Geração	Iteração

Convém referir que, embora muitas das características dos AGs derivem directamente da sua analogia com a natureza, existem, no entanto, algumas diferenças fundamentais entre os dois processos. Uma diferença crucial entre o processo natural e a sua simulação

computacional refere-se à estrutura genética. Muitas espécies evoluídas são diplóides, ou seja, apresentam pares de cromossomas em vez de apresentá-los isoladamente. Isso significa que cada característica é organizada em dois genes, cada um pertencente a um dos cromossomas homólogos. Em caso de conflito entre os valores, o gene dominante define o fenótipo, mas o outro gene, chamado recessivo, ainda será herdado pelos descendentes. No entanto, este aspecto não é normalmente implementado nos AGs, cujos indivíduos geralmente são haplóides.

Outra diferença interessante é a distinção de género, que não ocorre no modelo computacional. Nos AGs, o cruzamento de dois indivíduos quaisquer pode gerar descendentes, ao contrário do que acontece no processo natural, onde em geral apenas a relação macho-fêmea deixa novos descendentes.

Estas diferenças são interessantes, pois podem dar origem a novas abordagens a serem adoptadas nos AGs.

3.4. IMPLEMENTAÇÃO DE UM ALGORITMO GENÉTICO

Antes de implementar um AG é necessário proceder à escolha dos requisitos necessários para a sua implementação.

O primeiro passo consiste em definir o modo de representar as potenciais soluções, ou seja, em estabelecer um genótipo que descreva as características (*i.e.*, o fenótipo) do problema. Este processo é conhecido como codificação.

O segundo passo consiste em definir uma maneira de avaliar os indivíduos, isto é, analisar quantitativamente a utilidade de um vector. A medida numérica de qualidade de um indivíduo é denominada aptidão. A tarefa de avaliar o genótipo de um indivíduo e de lhe atribuir uma aptidão é realizada pela função de aptidão (*f*).

A codificação e a função de aptidão constituem as únicas etapas do processo de definição de um AG que são intrinsecamente dependentes do problema.

Depois de estabelecidos estes dois requisitos, deve-se determinar os parâmetros e as variáveis para controlar o algoritmo entrando em linha de conta com:

- o número de vectores da população;
- o número de gerações;
- a probabilidade de reprodução (p_r);
- a probabilidade de cruzamento (p_c);
- a probabilidade de mutação (p_m);
- a percentagem de população que vai ser substituída.

Por último deve-se estabelecer o critério para terminar o processo.

Dados estes passos, pode-se dar início à implementação do AG. A figura 3.1 representa a estrutura de um AG simples, onde P(t) é a população na geração t.

```
Início
t = 0
inicializar aleatoriamente P(t)
avaliação P(t)
repetir
   cruzamento P(t)
   mutação P(t)
   avaliação P(t)
   seleccionar P(t + 1) a partir de P(t)
   t = t + 1
até condição de conclusão verificada
fim do algoritmo
```

Figura 3.1 Algoritmo genético simples

A implementação de um algoritmo genético começa pela geração de uma população formada por um conjunto de indivíduos que são possíveis soluções do problema. Geralmente, essa inicialização é feita de modo aleatório. No entanto, podem ser adoptados outros métodos, como tentar gerar soluções iniciais distribuídas em todo o espaço de pesquisa possível, ou inserir soluções que tenham sido encontradas de um outro modo, para garantir que aquela região do espaço seja visitada.

A seguir, deve-se seleccionar os indivíduos (em geral dois) para serem recombinados. É importante observar que a selecção não é feita de modo determinístico. Qualquer indivíduo pode ser escolhido, contando cada um com uma probabilidade proporcional à sua aptidão.

Aplica-se então o operador de cruzamento, em geral chamado de *crossover*. O que o cruzamento faz, em termos simples, é combinar o material genético de dois ou mais indivíduos de forma a gerar descendentes.

Os indivíduos gerados pelo cruzamento devem então ser submetidos à mutação. A mutação, contrariamente ao *crossover*, é aplicada sobre um único indivíduo e provoca uma modificação, em geral pequena, no seu genótipo. Esse processo possibilita, em princípio, o surgimento de novas características, inexistentes até então na população.

As novas soluções geradas a partir do cruzamento e da mutação constituem a geração seguinte. Não é difícil perceber que os indivíduos deste novo conjunto tendem a ser melhores do que aqueles que lhe deram origem, uma vez que a selecção valoriza os indivíduos mais aptos.

Este processo deve ser repetido sucessivamente até que a condição de conclusão seja verificada.

3.4.1. CODIFICAÇÃO

A codificação dos cromossomas é das primeiras questões a tratar quando começamos a resolver um problema utilizando AGs. A codificação depende muito do problema e é essencial no desempenho do algoritmo.

A codificação binária é historicamente importante, uma vez que foi utilizada nos trabalhos pioneiros de Holland (Holland 1962). Além disso, é ainda a representação mais adoptada, por ser de fácil utilização e manipulação, assim como simples de analisar. Neste tipo de codificação os elementos do cromossoma pertencem ao conjunto {0, 1}. Na figura 3.2, podemos ver a representação de um cromossoma que utiliza uma codificação binária de 10 *bits*, onde cada um dos *bits* representa a presença, ou não, de uma determinada característica.

1	1	0	0	0	1	1	0	1	1

Figura 3.2 Codificação binária utilizando 10 *bits*

A codificação binária permite representar um número elevado de cromossomas, utilizando somente dois alelos (0 e 1). No caso da codificação representada na figura anterior podíamos representar até 1024 (2^{10}) cromossomas diferentes.

No entanto, existem casos onde o que se pretende codificar são valores de parâmetros. Nestas situações, os genes não podem ser representados por um único *bit* mas sim por um conjunto de *bits*, onde o número de *bits* a utilizar depende dos valores admissíveis para cada parâmetro. Por exemplo, se o objectivo do problema for o de maximizar uma função de três variáveis, $F(x, y, z)$ e a representação de cada uma das variáveis tiver que ser efectuada utilizando valores binários de 8 *bits*, o cromossoma terá assim três genes e um total de 24 *bits*. Este exemplo está ilustrado na figura 3.3.

1 1 0 1 0 0 0 1	0 0 1 1 1 0 1 1	1 0 0 0 1 1 0 1
Gene 1	**Gene 2**	**Gene 3**

Figura 3.3 Codificação binária de um cromossoma contendo três genes

Todavia, verifica-se uma crescente falta de unanimidade quanto a esta representação ser a mais adequada e natural. Um exemplo óbvio, quanto à sua falta de adequação, relaciona-se com a resolução de problemas que necessitam de valores com uma elevada precisão. Estes problemas requerem a utilização de um elevado número de *bits* o que requer um maior esforço computacional para explorar um vasto espaço de potenciais soluções. Uma representação mais natural para estes problemas passa pela utilização de números reais para representar as soluções. Este tipo de representações é utilizado em trabalhos apresentados por (Mühlenbein, Schomish e Born 1991) e (Michalewicz 1996). Na figura 3.4, podemos ver a representação de um cromossoma que utiliza codificação real.

1,232445	5,323456	0,455675
Gene 1	Gene 2	Gene 3

Figura 3.4 Codificação real de um cromossoma contendo três genes

Normalmente, quando se utiliza este tipo de codificação, os operadores de cruzamento e de mutação são diferentes dos adoptados para os AGs binários.

Alguns problemas de optimização combinatória apresentam também dificuldades ao nível da representação, quando se tenta utilizar os AGs para os solucionar. O Problema do Caixeiro Viajante (PCV) é um desses casos. As representações mais adequadas para estes problemas são as representações genéticas baseadas em permutações ou na ordem. Nestas representações, cada cromossoma é definido como uma permutação de todos os símbolos do alfabeto, ou seja, uma sequência ordenada de todos os símbolos, não sendo permitidas repetições dos símbolos. A figura 3.5 apresenta dois exemplos de codificação por permutação. No cromossoma A utiliza-se o alfabeto {1, 2, 3, 4, 5, 6, 7, 8, 9}, e no cromossoma B utiliza-se $\{a, b, c, d, e, f, g, h, i\}$.

Cromossoma A	**2**	**1**	**3**	**5**	**4**	**6**	**8**	**7**	**9**
Cromossoma B	***b***	***a***	***c***	***e***	***d***	***f***	***h***	***g***	***i***

Figura 3.5 Codificação por permutação

Na codificação por permutação, o número de símbolos do alfabeto utilizado determina qual o comprimento dos cromossomas, e o alfabeto define o conjunto de alelos de cada um dos genes.

Tal como na codificação real, os operadores de cruzamento e de mutação são distintos dos adoptados para os AGs binários, pois temos que preservar as permutações de modo a que o cromossoma gerado seja válido.

Um outro tipo de codificação possível consiste na representação em árvore. Este tipo de codificação é utilizado principalmente em programação genética (PG). Na codificação em

árvore cada cromossoma representa uma árvore composta por funções (nós da árvore) e terminais (folhas das árvores). As funções podem ser operadores aritméticos, operações de programação e funções matemáticas convencionais e/ou funções lógicas e funções específicas do domínio. Podemos ver um exemplo de codificação em árvore na figura 3.6.

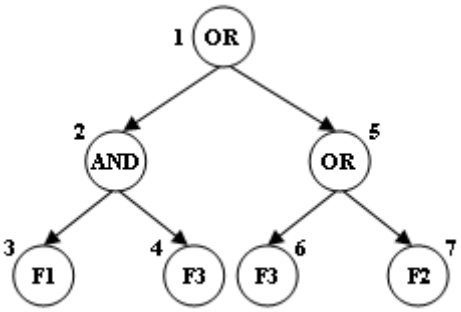


Figura 3.6 Codificação em árvore

3.4.2. FUNÇÃO DE APTIDÃO

A função de aptidão atribui a cada cromossoma um valor que mede a qualidade do indivíduo representado por esse cromossoma. Normalmente esta função pode ser determinada sem grande dificuldade, já que a própria definição do problema contém implicitamente os critérios que permitem separar as boas soluções das más. No entanto, deve-se ter algum cuidado ao definir a função de aptidão, uma vez que ela é a base do processo de selecção, fundamental para o funcionamento dos AGs.

Quando a questão se trata de um problema de optimização numérica convencional, o conceito de função de avaliação (*fav*) confunde-se com a ideia de função de aptidão. No entanto, devem-se distinguir as noções de função de avaliação e de função de aptidão usadas nos AGs. A primeira fornece uma medida do desempenho em relação a um conjunto particular de parâmetros e é restrita a um indivíduo. A segunda transforma essa medida de desempenho numa grandeza que representa oportunidade de reprodução. Além disso, a aptidão é sempre definida em relação aos outros indivíduos da população.

Algumas vezes o valor da aptidão pode ser fornecido pela própria função de avaliação. No entanto, isso nem sempre é possível. Se os valores fornecidos por essa função forem muito próximos uns dos outros o processo evolutivo corre o risco de tornar-se aleatório, dependendo do método de selecção escolhido. Um outro problema que pode ocorrer é a

aptidão de um indivíduo ser muito maior do que as demais e o "super indivíduo" dominar a população no momento da selecção. Esse indivíduo pode não representar uma solução óptima, o que levaria a uma convergência prematura.

Existem vários métodos para transformar a função de avaliação na aptidão do indivíduo. Um deles é conhecido por escalonamento linear (*linear scaling*), em que o valor de aptidão (f_i) do indivíduo *i* tem a seguinte relação linear com o valor da função de avaliação:

$$f_i = a\, fav_i + b \tag{3.1}$$

Se os parâmetros *a* e *b* forem constantes ao longo das gerações este escalonamento é independente do problema.

Pode-se também utilizar um escalonamento através de uma função potência, onde o valor de aptidão é calculado a partir da potência de ordem *k* do valor da avaliação:

$$f_i = fav_i^k \tag{3.2}$$

O valor de *k* varia de acordo com o problema e normalmente é um valor próximo da unidade.

Uma outra forma de transformar a função de avaliação na aptidão do indivíduo é utilizar o método da posição (*ranking method*). Deste modo, a população fica ordenada de acordo com a avaliação dos cromossomas e a aptidão passa a ser definida pela posição relativa que um indivíduo assume. Como a selecção é feita tendo como base a posição do indivíduo na população ordenada, a diferença de oportunidades entre os indivíduos que ocupam as posições *i* e *j* é constante e é independente do valor da sua avaliação.

Muitos problemas contêm restrições, isto é, contêm soluções não admissíveis. Uma das dificuldades no processo de definição da função de aptidão refere-se à escolha da estratégia a ser utilizada para lidar com restrições. Numa situação ideal, a representação deveria ser tal que apenas soluções admissíveis pudessem ser geradas. Porém, na prática, nem sempre isso é possível. Convém referir que excluir as soluções candidatas inválidas da população inicial não basta, uma vez que elas podem reaparecer durante o processo de evolução. Para contornar este problema, podem ser adoptadas várias estratégias que serão abordadas com mais detalhe na secção 3.4.10.

3.4.3. DEFINIÇÃO DA POPULAÇÃO INICIAL

Idealmente, a população inicial deveria ser tal que todos os possíveis alelos de cada gene estivessem representados, conduzindo assim a uma diversidade apreciável. A diversidade garante que todo o espaço de pesquisa esteja representado na população. Não é difícil perceber que, quanto maior o tamanho da população, mais fácil se torna essa tarefa. No entanto, o aumento do tamanho da população tem um custo computacional associado, tanto em memória como em desempenho.

O método mais utilizado na criação da população é a inicialização aleatória dos indivíduos.

Existe também um outro método conhecido como *seeding*. Este consiste em colocar na população inicial algumas soluções encontradas previamente por outros métodos de optimização ou que estamos certos de que apresentem grande potencial. Isso garante que aquela região do espaço de pesquisa seja visitada pelo AG.

Já em problemas com restrição, deve-se ter atenção para não gerar indivíduos inválidos na etapa de inicialização.

3.4.4. SELECÇÃO

Numa população onde a cada indivíduo foi atribuído um valor de aptidão, o processo de selecção permite seleccionar os progenitores que vão efectuar a reprodução e escolher os indivíduos que devem sobreviver para a geração seguinte. Quanto maior for a aptidão de um indivíduo, maior será a probabilidade de ele ser escolhido.

Um conceito importante a ser introduzido, quando se fala em selecção, é o de pressão de selecção. A pressão de selecção é a razão entre a aptidão máxima e a aptidão média da população. Quando os melhores indivíduos da população têm um valor de aptidão muito elevado em relação aos demais no momento da selecção, diz-se que a pressão de selecção é alta. Uma pressão alta diminui o tempo de convergência, mas pode ocasionar a convergência prematura. Uma pressão de selecção demasiado baixa, por outro lado, pode tornar o processo quase aleatório, devido à probabilidade de selecção ser praticamente igual para todos os indivíduos.

A maioria dos métodos de selecção é projectada para escolher, preferencialmente, indivíduos com maiores valores de aptidão, embora não exclusivamente, a fim de manter a diversidade da população

Existem vários métodos para seleccionar os indivíduos sobre os quais serão aplicados os operadores genéticos. De seguida serão descritos alguns métodos de selecção.

3.4.4.1. MÉTODO DA SELECÇÃO POR ROLETA

No método de selecção por roleta, que é um dos mais utilizados, indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta. Neste tipo de selecção, a probabilidade de um indivíduo da população ser seleccionado é proporcional à sua aptidão relativa. Considerando uma população de N elementos, a probabilidade de selecção (proporcional) do indivíduo I_i é dada por:

$$p(I_i) = \frac{f(I_i)}{\sum_{j=1}^{N} f(I_j)} \tag{3.3}$$

Desta forma, para indivíduos com alta aptidão, é dada uma fracção maior da roleta, enquanto que aos indivíduos de aptidão mais baixa é dada uma fracção proporcionalmente menor.

Para exemplificar este método vamos considerar uma população de 6 indivíduos cuja aptidão é dada por uma função qualquer (neste caso é simplesmente a conversão de binário para decimal) conforme é representado na tabela 3.2.

Tabela 3.2 Valores de exemplo para ilustrar o método selecção por roleta

Indivíduo I_i	**Vector**	**Aptidão $f(I_i)$**	**Aptidão relativa**	**Percentagem**
I_1	0100100	36	0.09	9%
I_2	0011011	27	0.07	7%
I_3	0110011	51	0.12	12%
I_4	1110010	114	0.28	28%
I_5	0111011	59	0.14	14%
I_6	1111011	123	0.30	30%
	$\sum$	410	1.00	100%

Com os valores da quarta coluna da tabela 3.2, elaborámos a roleta representada na figura 3.7. Esta roleta irá ser rodada 6 vezes para efectuar a selecção dos indivíduos que participarão da próxima geração. Consequentemente, leva-se em conta que os indivíduos com maior área na roleta têm maior probabilidade de serem seleccionados que os indivíduos menos aptos.

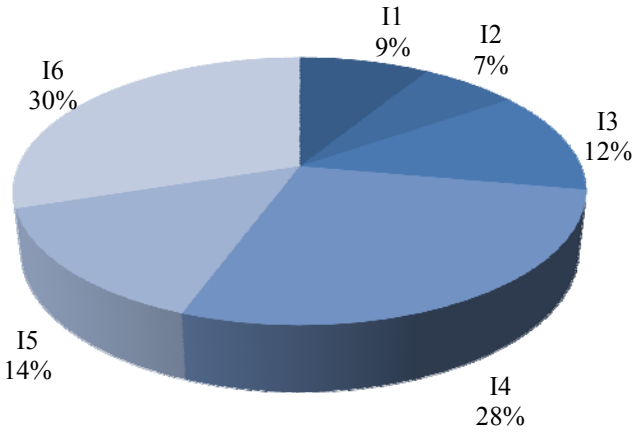


Figura 3.7 Representação gráfica da roleta

Não é difícil perceber que este tipo de selecção depende de aptidões numéricas. Este tipo de selecção não é adequado quando se utiliza o método da posição (*ranking*) para transformar a função de avaliação na aptidão do indivíduo. Além disso, exige valores não negativos, uma vez que a aptidão representa a área da fracção da roleta (onde não faz sentido grandezas negativas). Uma terceira observação a respeito deste tipo de selecção refere-se aos já citados super indivíduos. Por exemplo, na figura 3.7, os indivíduos I_4 e I_6 contariam, juntos, com uma probabilidade de serem escolhidos superior à soma de todos os outros.

3.4.4.2. MÉTODO DE SELECÇÃO POR POSIÇÃO (RANKING)

A selecção por posição (*ranking*) pode ser dividida em duas fases. Na primeira fase, os indivíduos são ordenados de acordo com seus valores da função de avaliação, por ordem crescente, se o objectivo for maximizar a função de avaliação, ou por ordem decrescente, caso se pretenda minimizá-la.

Considerando que pretendemos maximizar uma função de avaliação, teremos uma lista ordenada por ordem crescente, onde o melhor indivíduo se encontra na posição N,

enquanto que o pior indivíduo está na primeira posição (o pior terá valor 1, o segundo pior terá valor 2 e assim sucessivamente).

Tendo a lista ordenada, é atribuído a cada indivíduo um valor de aptidão equivalente à sua posição na lista. Este valor pode ser calculado de diversas formas dependendo se pretendemos uma projecção linear, ou não linear, e se existe, ou não, pressão de selecção.

No caso de não existir pressão de selecção, todos os indivíduos têm a mesma probabilidade de serem seleccionados, $1/N$, visto que a probabilidade de selecção é independente do valor da função de avaliação. Caso contrário, quanto melhor a posição do indivíduo no *ranking*, maior será a probabilidade ser seleccionado.

Numa segunda fase, é aplicado um procedimento similar à selecção por roleta.

3.4.4.3. MÉTODO DA SELECÇÃO POR TORNEIO

Um outro método frequentemente adoptado é a selecção por torneio, no qual um número $k \geq 2$ de indivíduos da população é escolhido aleatoriamente para formar uma subpopulação temporária. O indivíduo com maior aptidão é o seleccionado para a população intermediária sendo o processo repetido até que a população intermediária seja preenchida.

Este método é o mais utilizado, pois oferece a vantagem de não exigir que a comparação seja feita entre todos os indivíduos da população. A tabela 3.3 mostra um exemplo de utilização do método de selecção por torneio.

Tabela 3.3 Valores de exemplo para ilustrar o método selecção por torneio

Indivíduo I_i	**Aptidão** $f(I_i)$
I_1	234
I_2	435
I_3	56
I_4	162

Baseado na tabela 3.3, vamos supor que são sorteados os indivíduos I_1 e I_2. Seria o indivíduo 2 quem venceria o torneio, pois o seu valor de aptidão é maior. Devido a esse facto, I_2 seria o indivíduo seleccionado para reprodução.

Este método possui a grande vantagem de não gerar super indivíduos, pois a probabilidade do indivíduo com valor de aptidão mais elevado ser seleccionado permanece constante, independentemente do seu valor de aptidão ser alto. No exemplo ilustrado na tabela 3.3 a probabilidade do indivíduo I_2 ser seleccionado é 1/4, pois, se ele for sorteado, independentemente de quem seja seu rival, ele vencerá sempre o torneio. Se o valor de aptidão de I_2 fosse $f(I_2) = 1000$ em vez de $f(I_2) = 435$, a probabilidade de selecção continuaria a ser mesma, para este método. Por outro lado, no método da roleta, o intervalo de selecção iria aumentar muito, e por isso, a probabilidade do indivíduo ser seleccionado também iria ser superior.

Este método tem bastante popularidade devido à sua fácil implementação, do ponto de vista de eficiência computacional, e por permitir apurar o peso da selecção através do aumento ou diminuição do tamanho do torneio (k).

3.4.4.4. MÉTODO DA AMOSTRAGEM ESTOCÁSTICA UNIVERSAL

Este método pode ser considerado como uma variação do método da roleta, na qual, ao contrário de uma única agulha, são colocadas N agulhas igualmente espaçadas, sendo N o número de indivíduos a serem seleccionados para a próxima geração. Desta forma, a roleta é rodada uma única vez (ao contrário de N vezes como ocorre no método da roleta) seleccionando assim os N indivíduos de uma só vez.

Evidentemente, os indivíduos cujas regiões possuem uma maior área terão maior probabilidade de serem seleccionados várias vezes. Consequentemente, a selecção de indivíduos pode conter várias cópias de um mesmo indivíduo, enquanto outros podem desaparecer.

3.4.5. ELITISMO

O elitismo foi introduzido em 1975 por Kenneth De Jong (De Jong 1975) e normalmente apresenta-se associado a outros métodos de selecção, na tentativa de aumentar a velocidade de convergência do algoritmo. A estratégia de elitismo força os AGs a reterem um

determinado número dos melhores indivíduos em cada geração. Tais indivíduos podem ser perdidos se eles não forem seleccionados para reprodução ou se eles forem destruídos pelos operadores genéticos. Muitos pesquisadores têm encontrado no elitismo vantagens significativas para o desempenho dos AGs (Mitchell 1997).

Esta estratégia consiste basicamente em realizar o processo de selecção em duas etapas:

- Selecciona-se uma elite de *E* indivíduos entre os melhores da população actual, os quais serão incorporados directamente na geração seguinte sem nenhuma alteração;

- Os outros (*N*-*E*) indivíduos da população são gerados normalmente, através do método de selecção e posterior aplicação dos operadores genéticos.

Assim, as melhores soluções não são apenas passadas de uma geração para outra, mas também participam da criação dos novos membros da nova geração.

Geralmente, a elite tem um tamanho reduzido pois um elitismo elevado pode levar a convergência prematura do AG.

3.4.6. PARÂMETROS GENÉTICOS

A escolha dos operadores genéticos, juntamente com a determinação da função aptidão e da representação apropriada dos cromossomas é determinante para o bom funcionamento de um AG. Eles são utilizados para criar novas soluções baseadas nas soluções existentes na população. Antes de efectuar esta escolha, deve-se analisar de que modo alguns parâmetros genéticos influenciam no comportamento dos Algoritmos Genéticos, para que se possa estabelecê-los de acordo com as necessidades do problema e dos recursos disponíveis.

3.4.6.1. TAMANHO DA POPULAÇÃO

O tamanho da população indica o número de indivíduos em cada geração, normalmente constante durante a evolução, afectando o desempenho global e a eficiência dos AGs. Uma população pequena oferece uma pequena cobertura do espaço de busca, causando um baixo desempenho. Uma grande população fornece uma melhor cobertura do domínio do problema e previne a convergência prematura para soluções locais. No entanto, com uma

grande população tornam-se necessários recursos computacionais significativos e um tempo de processamento do algoritmo mais elevado.

3.4.6.2. PROBABILIDADE DE CRUZAMENTO

Quanto maior for a probabilidade de cruzamento, p_c, mais rapidamente novas soluções serão introduzidas na população. Por outro lado, esta estratégia pode gerar um efeito indesejado, pois a maior parte da população será substituída podendo ocorrer perda de indivíduos com aptidões elevadas. Para valores baixos desta probabilidade, serão gerados menos indivíduos por geração, o que pode originar um aumento do número de gerações necessárias para obter os mesmos resultados e, consequentemente, o algoritmo pode tornar-se muito lento. Na prática, esta probabilidade, varia entre 60% e 90%.

3.4.6.3. PROBABILIDADE DE MUTAÇÃO

A probabilidade de mutação, p_m, indica a frequência com que ocorrerá uma mutação de cromossomas nas populações ao longo da evolução.

A mutação é utilizada para fornecer novas informações dentro das populações, prevenindo que as mesmas se tornem saturadas com cromossomas semelhantes (convergência prematura). Com uma probabilidade muito alta a pesquisa torna-se essencialmente aleatória além de potenciar que a possibilidade de que uma boa solução seja destruída. A probabilidade de mutação ideal dependerá da aplicação a ser resolvida; todavia, a maioria dos valores utilizados varia entre 0,1% e 5%.

3.4.6.4. TAXA DE SUBSTITUIÇÃO

A taxa de substituição é um parâmetro que define qual a proporção de indivíduos da população substituída em cada geração. Se a percentagem de indivíduos a substituir for de 100% todos os indivíduos da população actual são substituídos pelos novos indivíduos resultantes da reprodução. Quanto menor for esta percentagem, mais lenta se torna a convergência do algoritmo.

3.4.7. OPERADORES GENÉTICOS

O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações, estendendo a pesquisa até chegar a um resultado satisfatório. Os

operadores genéticos são necessários para que a população se diversifique e mantenha as características de adaptação adquiridas pelas gerações anteriores.

Durante a fase de reprodução de um AG, seleccionam-se dois indivíduos da população que serão recombinados para formar descendentes, que, por sua vez, constituirão a geração seguinte. Os pares são seleccionados aleatoriamente, usando-se um método que favoreça os indivíduos melhor adaptados. Logo que forem escolhidos os pares, os seus códigos genéticos misturam-se e combinam-se, usando o operador de cruzamento (*crossover*).

O processo de cruzamento constitui um dos passos (se não o passo) mais importantes no mecanismo evolutivo dos AGs. Existem diversos operadores de cruzamento. Embora a maioria deles possa ser utilizada com um número maior de indivíduos, será descrito aqui o caso em que eles são aplicados a um par de cromossomas, visto que constitui a situação mais comum.

Durante o processo de evolução que ocorre nos AGs, pode acontecer que alguns elementos do alfabeto adoptado desapareçam definitivamente da população. Para evitar que isso ocorra e que uma porção do espaço de busca seja abandonada, utiliza-se o operador genético conhecido como mutação. A mutação é um processo simples, geralmente encarado como um operador secundário, que ajuda a manter a diversidade na população. Ao contrário do cruzamento, a mutação opera sobre um único indivíduo, provocando pequenas perturbações no seu genótipo que se irão reflectir de alguma forma no fenótipo. Este operador é aplicado a todos os descendentes após o cruzamento. Cada gene pode ser modificado com uma probabilidade p_m.

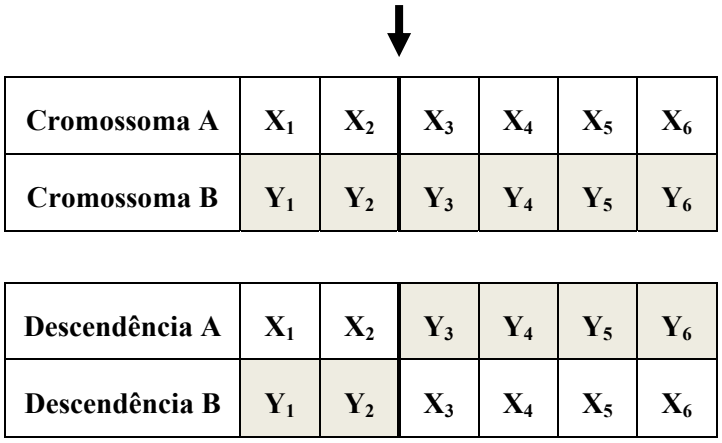
3.4.7.1. CRUZAMENTO PARA CODIFICAÇÃO BINÁRIA

O cruzamento consiste em dividir aleatoriamente os cromossomas, produzindo segmentos anteriores e posteriores que realizam um intercâmbio para obter novos cromossomas (descendentes).

As três formas mais comuns de reprodução em AGs são o cruzamento de um ponto, o cruzamento de dois pontos e o cruzamento uniforme, que serão detalhados a seguir.

Cruzamento de um ponto

Um ponto de cruzamento, chamado ponto de corte ou ponto de *crossover*, é sorteado aleatoriamente de forma a que as informações genéticas dos pais serão trocadas a partir desse ponto. As informações anteriores a esse ponto num dos pais são ligadas às informações posteriores a este ponto no outro progenitor. Na figura 3.8 é ilustrado o cruzamento de um ponto utilizando cromossomas de comprimento 6. O ponto de cruzamento foi aleatoriamente seleccionado para a posição 2 onde se dá o corte para troca de material genético.



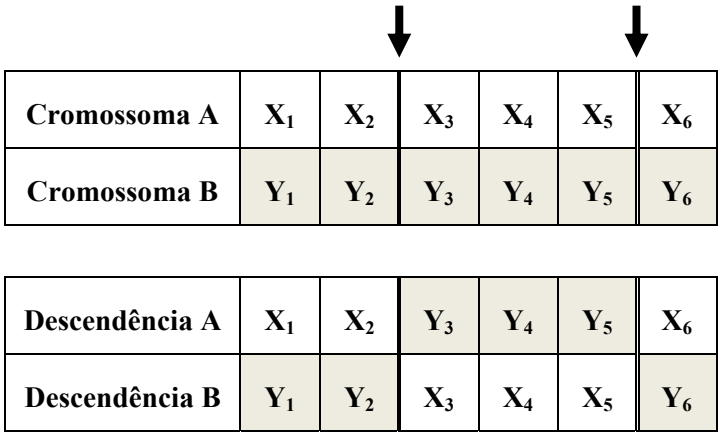
Cromossoma A	X_1	X_2	X_3	X_4	X_5	X_6
Cromossoma B	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6

Descendência A	X_1	X_2	Y_3	Y_4	Y_5	Y_6
Descendência B	Y_1	Y_2	X_3	X_4	X_5	X_6

Figura 3.8 Cruzamento de um ponto na posição 2

Cruzamento de dois pontos

Os dois pontos de corte são escolhidos aleatoriamente, e as secções entre os dois pontos são trocadas entre os pais. Um exemplo de cruzamento de dois pontos pode ser visto na figura 3.9, sendo, por hipótese, o primeiro ponto de cruzamento na posição 2 e o segundo na posição 5. Este tipo de cruzamento pode ser extensível a um número *n* de pontos.



Cromossoma A	X_1	X_2	X_3	X_4	X_5	X_6
Cromossoma B	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6

Descendência A	X_1	X_2	Y_3	Y_4	Y_5	X_6
Descendência B	Y_1	Y_2	X_3	X_4	X_5	Y_6

Figura 3.9 Cruzamento de dois pontos na posição 2 e 6

Cruzamento uniforme

No cruzamento uniforme é utilizada uma máscara, gerada aleatoriamente, que tem uma estrutura análoga à do cromossoma e que guarda em cada posição informação se o material genético deve ou não ser trocado. Nas posições onde houver o *bit* 1 ocorre a troca de valores entre os pais e nas posições onde estiver o *bit* 0 são mantidos os valores dos pais. Um exemplo de um cruzamento uniforme é ilustrado na figura 3.10.

Cromossoma A	X_1	X_2	X_3	X_4	X_5	X_6
Cromossoma B	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6

Máscara	**0**	**1**	**0**	**0**	**1**	**1**

Descendência A	X_1	Y_2	X_3	X_4	Y_5	Y_6
Descendência B	Y_1	X_2	Y_3	Y_4	X_5	X_6

Figura 3.10 Exemplo do cruzamento uniforme

3.4.7.2. CRUZAMENTO PARA CODIFICAÇÃO REAL

Os operadores convencionais (cruzamento de *n* pontos e cruzamento uniforme) são adequados à representação binária, mas podem também ser aplicados em algoritmos que utilizam a codificação real. No entanto, com excepção de alguma aplicação muito específica, esses operadores não parecem ser a melhor opção. Esta observação é válida porque, durante o processo de recombinação, os cruzamentos convencionais apenas trocam informações entre os pais, e não há a geração de novos alelos. De modo a ultrapassar esta deficiência, foram propostos vários operadores que parecem ser mais adequados à codificação real, sendo conhecidos como operadores aritméticos.

Cruzamento aritmético

Segundo o cruzamento aritmético (Michalewicz 1996), existindo dois cromossomas X_1 e Y_1 são produzidos dois novos cromossomas D_1 e D_2 da seguinte forma:

$$D_1 = a\,X_1 + (1 - a)\,Y_1 \quad \textbf{(3.4a)}$$

$$D_2 = a\,Y_1 + (1 - a)\,X_1 \quad \textbf{(3.4b)}$$

onde a é um número no intervalo [0, 1], que pode ser definido de várias maneiras, até mesmo aleatoriamente.

Cruzamento média

O cruzamento média (Davis 1991) dá origem a um único descendente, que é criado a partir da média aritmética dos pais:

$$D = \frac{X_1 + Y_1}{2} \quad \textbf{(3.5)}$$

Cruzamento geométrico

O cruzamento geométrico é uma variação do cruzamento média. O descendente é criado a partir da média geométrica (*i.e.*, da raiz quadrada do produto) dos pais:

$$D = \sqrt{X_1\,Y_1} \quad \textbf{(3.6)}$$

Cruzamento linear

No cruzamento linear (Wright 1991) numa primeira fase são gerados três descendentes da seguinte forma:

$$D_1 = 0{,}5X_1 + 0{,}5Y_1 \quad \textbf{(3.7a)}$$

$$D_2 = 1{,}5X_1 - 0{,}5Y_1 \quad \textbf{(3.7b)}$$

$$D_3 = -0{,}5X_1 + 1{,}5Y_1 \quad \textbf{(3.7c)}$$

Uma vez gerados os três descendentes, eles são avaliados e os dois melhores são considerados. Nota-se que, neste caso, diferentemente do que acontecia nos métodos anteriores, a aptidão influencia o processo de recombinação.

3.4.7.3. CRUZAMENTO PARA CODIFICAÇÃO POR PERMUTAÇÃO

Na codificação por permutação não é possível utilizar os cruzamentos referidos anteriormente, pois o cruzamento entre dois indivíduos deve dar origem a descendentes válidos. Por exemplo, num problema PCV onde cada cidade {1, 2, ..., 6} é visitada por ordem crescente tem-se a representação da figura 3.11:

Cromossoma	**1**	**2**	**3**	**4**	**5**	**6**

Figura 3.11 Exemplo de um cromossoma de um problema PCV

Com esta representação a função de aptidão do problema PCV só depende da posição dos genes. Assim, é necessário um operador análogo ao operador de cruzamento, que permita a troca da ordem do percurso de acordo com as características dos pais, mantendo o percurso válido. Os operadores de cruzamento devem ser operadores genéticos capazes de combinar a construção ordenada de blocos a partir dos pais. Com esse objectivo, foram desenvolvidos os operadores apresentados de seguida:

Cruzamento ordenado (*Order Crossover* - OX)

O cruzamento ordenado (Syswerda 1991) gera filhos a partir da escolha de uma sequência parcial dos genes de um dos pais, preservando a ordem relativa dos genes do outro pai.

Os filhos (ver figura 3.12), descendente A e descendente B, herdam as sequências entre os pontos de corte de seus respectivos pais, cromossoma A e cromossoma B. A seguir, partindo do segundo corte de um pai (por exemplo B), copia-se os genes na mesma ordem em que aparecem, removendo aqueles contidos entre os dois cortes do outro pai (no caso A).

Assim, a partir da sequência 5, 3, 2, 1, 4 e 6, obtêm-se a sequência 1, 4 e 6 (removendo 3, 2 e 5 já presentes) a ser inserida no descendente A a partir do seu segundo ponto de corte. Do mesmo modo, a partir da sequência 4, 1, 6, 3, 2 e 5 do cromossoma A, têm-se a sequência 3, 2 e 5 (removendo 1, 4 e 6 do cromossoma B) a ser inserida no descendente B a partir de seu segundo corte. Desta forma, obtêm-se os filhos ilustrados na figura 3.12.

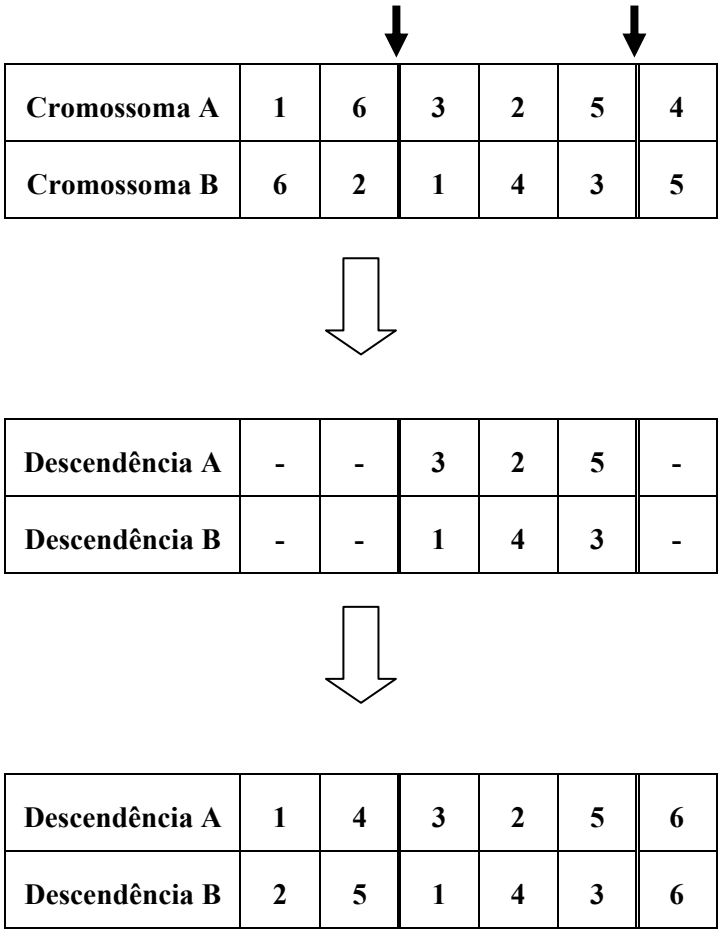


Figura 3.12 Exemplo do cruzamento aplicando operador OX

Cruzamento parcialmente semelhante (Partially matched crossover - PMX)

O procedimento para efectuar o cruzamento parcialmente semelhante (Goldberg e Lingle 1985) é o seguinte (ver figura 3.13): tendo dois cromossomas A e B, efectuam-se dois pontos de corte aleatórios. Os filhos, descendente A e descendente B, herdam integralmente as sequências parciais entre os dois pontos de corte, respectivamente do cromossoma A e do cromossoma B, ou seja, há uma troca de subsequências. Para isso, deve-se preservar a ordem e a posição de cada gene nas subsequências.

Esta troca define também as seguintes projecções: 1↔3, 4↔2, 6↔5, para reparar rotas inviáveis. Cada gene de descendente A, ainda não conhecido após a colocação das sequências parciais, é preenchido a partir de seu pai A, e cada gene de descendente B não conhecido é preenchido a partir de seu pai B, desde que não forme uma rota inviável. A figura 3.13 descreve o procedimento do cruzamento PMX.

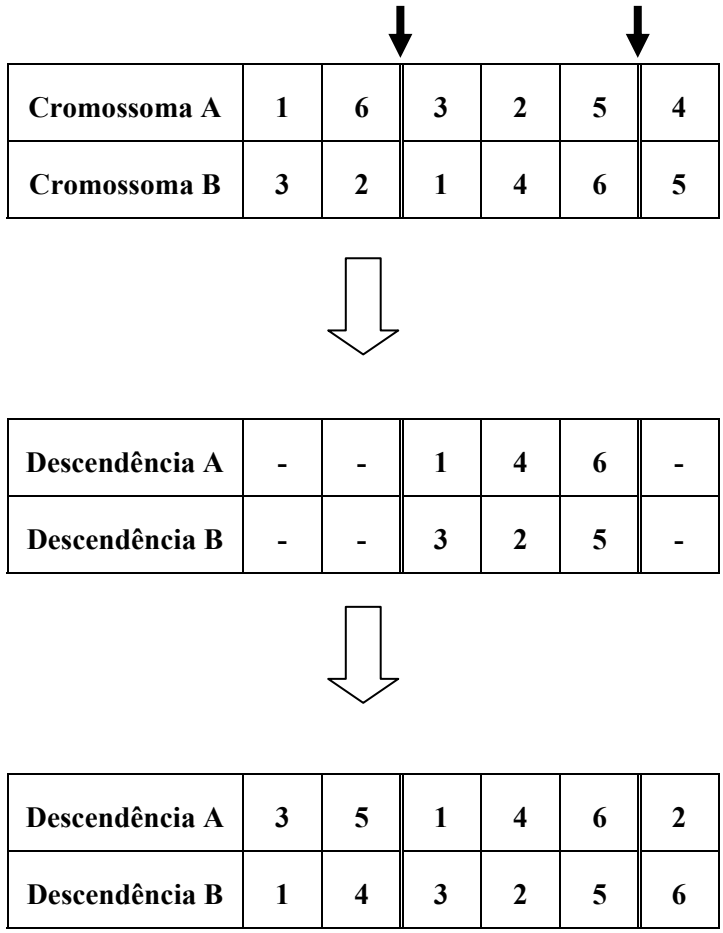


Figura 3.13 Exemplo do cruzamento aplicando operador PMX

Cruzamento cíclico (C*ycle crossover* – CX)

O operador de cruzamento cíclico (Oliver, Smith e Holland 1987) gera filhos que preservam a posição absoluta das cidades provenientes dos cromossomas pais. Dados dois pais A e B (ver figura 3.14), a primeira posição no descendente A recebe o primeiro gene do cromossoma A. O próximo gene do filho A é obtido do cromossoma B. Este gene é copiado para o filho na posição em que este mesmo gene aparece no cromossoma A. O algoritmo segue copiando genes do cromossoma B na posição correspondente no cromossoma A até que se forme um ciclo, ou seja, até que na posição correspondente no

cromossoma B encontra-se um gene já inserido no filho. Neste caso, se restarem posições não preenchidas no descendente A, essas são preenchidas da esquerda para a direita no vector do filho com os genes do cromossoma B na ordem relativa que aparecem neste pai. Trocando-se o cromossoma A pelo B é possível obter, de modo semelhante o outro descendente. A figura 3.14 apresenta um exemplo do cruzamento aplicando operador CX.

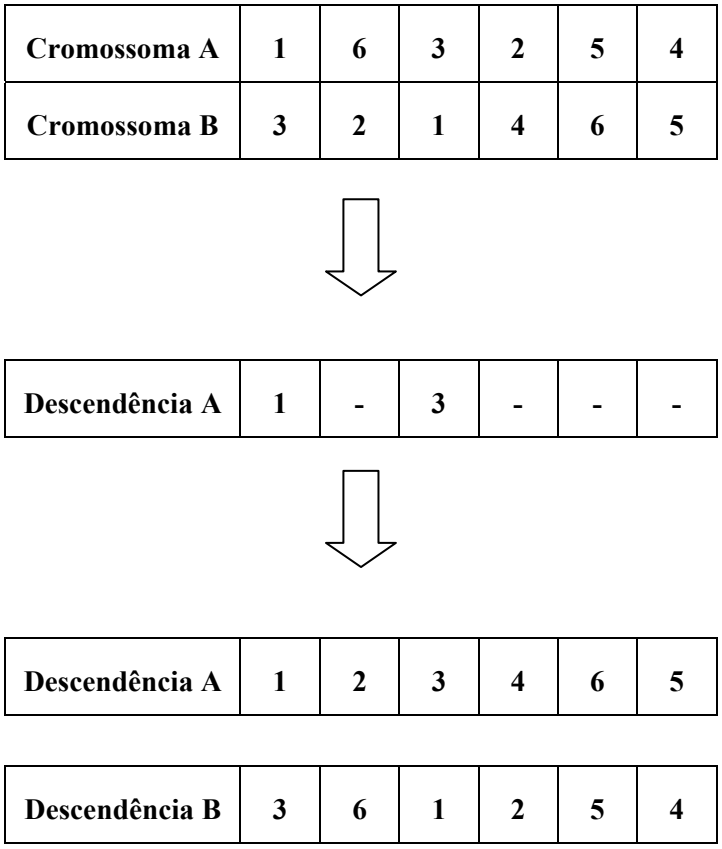


Figura 3.14 Exemplo do cruzamento aplicando operador CX

3.4.7.4. CRUZAMENTO PARA CODIFICAÇÃO EM ÁRVORE

O cruzamento em árvore é efectuado tendo em conta duas funções escolhidas na fase de selecção. Depois de escolhidas as funções, a sua informação genética é partilhada, dando origem a duas novas funções diferentes das anteriores. Para tal, é efectuado um corte em cada uma das árvores que representam uma função, trocando depois os ramos cortados de cada uma delas e formando duas novas funções.

Considere-se dois pais da figura 3.15. Se forem escolhidos, de forma aleatória, o nó 5 da árvore da esquerda e o nó 3 da árvore da direita então os descendentes são obtidos trocando essas subárvores.

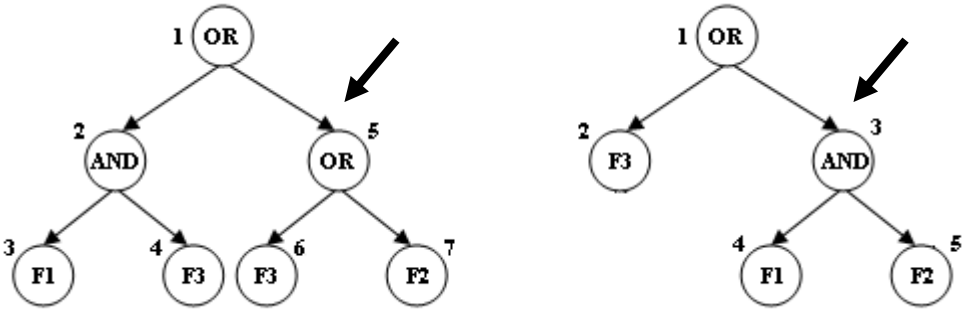


Figura 3.15 Estruturas de dois pais seleccionados para participar na operação de cruzamento

Após terem sido efectuados os cortes em cada uma das funções, a informação genética de ambas é partilhada dando origem a duas novas funções com características genéticas diferentes das primeiras, tal como indica a figura 3.16.

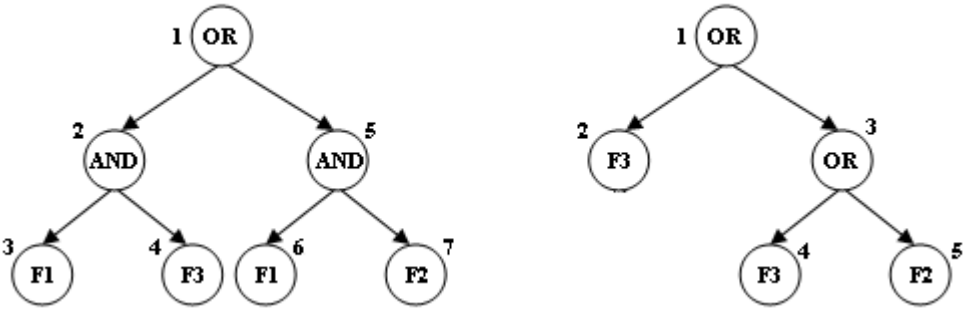


Figura 3.16 Estruturas descendentes da operação de cruzamento

Em termos de expressões gramaticais, os dois pais representam uma expressão equivalente:

$(F_1\ AND\ F_3)\ OR\ (F_3 OR\ F_2)$ $F_3\ OR\ (F_1\ AND\ F_2)$.

onde AND e OR representam as operações booleanas *e* e *ou* inclusivo.

No final da operação de cruzamento as novas estruturas a inserir na nova população correspondem às seguintes expressões:

$(F_1\ AND\ F_3)\ OR\ (F_1\ AND\ F_2)$ $F_3\ OR (F_3 OR\ F_2)$.

Convém referir que quando os dois pontos sujeitos a cruzamento são dois terminais é verificada apenas uma troca de terminais entre as árvores. Por outro lado, se os dois pontos escolhidos para cruzamento são as raízes, os descendentes são iguais aos pais.

3.4.7.5. MUTAÇÃO PARA CODIFICAÇÃO BINÁRIA

Na representação binária podemos realizar a mutação simplesmente escolhendo de forma aleatória um *bit* para sofrer mutação, ou seja, para ser mudado. A mudança de um *bit* no indivíduo pode fazê-lo representar outro ponto completamente diferente do espaço. Na figura 3.17 está ilustrado a alteração do *bit* 4 de um cromossoma quando este sofre uma mutação nesse local. Neste caso o *bit* que tem o valor '0', após a mutação passa a ter o valor '1'.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Cromossoma	1	0	0	**0**	1	1	0	1
Cromossoma após mutação	1	0	0	**1**	1	1	0	1

Figura 3.17 Mutação do bit número 4 de um cromossoma

3.4.7.6. MUTAÇÃO PARA CODIFICAÇÃO REAL

A principal diferença na implementação do operador de mutação, no caso de se adoptar uma codificação real em vez da codificação binária, é que na primeira os genes consistem nas próprias variáveis do problema e na segunda consistem em *bits*. Além disso, a codificação real apresenta maior flexibilidade na operação dos genes, não se limitando a simples cópia ou troca de *bits*.

Neste tipo de codificação, os operadores de mutação mais populares são a mutação uniforme e a mutação gaussiana; todavia, existem outros operadores como podemos ver a seguir.

Mutação uniforme

O operador de mutação uniforme é semelhante ao utilizado na codificação binária. Considere-se um cromossoma A (ver figura 3.18), onde cada parâmetro tem a mesma

probabilidade de ser escolhido para sofrer a mutação. Se o gene X_k é seleccionado para a mutação então é substituído por outro gene gerado aleatoriamente, segundo uma distribuição uniforme, entre os limites mínimo e máximo permitidos para esse gene. Nesta distribuição uniforme todos os números possuem a mesma probabilidade de ocorrer. Na figura 3.18, podemos ver um exemplo desta operação de mutação.

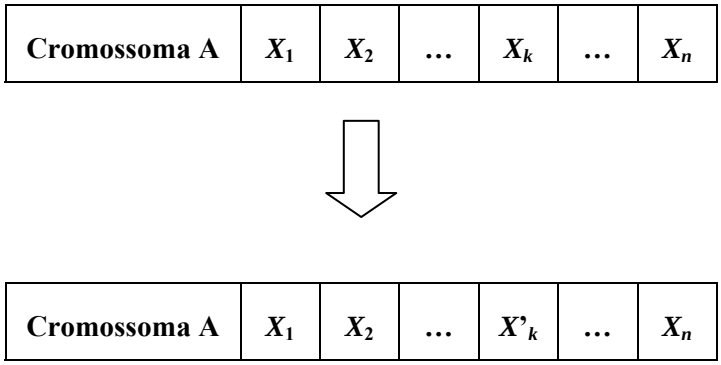


Figura 3.18 Mutação uniforme no elemento X_k

onde X'_k tem um valor aleatório do domínio compreendido entre X_{kmin} e X_{kmax}.

Mutação gaussiana

A mutação gaussiana consiste em substituir o gene seleccionado por outro gerado a partir de uma distribuição normal $N(pi, \sigma)$, com média *pi* igual ao valor do gene a ser substituído e desvio padrão σ definido pelo utilizador.

Mutação não uniforme

A mutação não uniforme (Michalewicz 1996) consiste na simples substituição de um gene por um número extraído de uma distribuição de probabilidade não uniforme, ou seja, onde os valores possuem probabilidades distintas de ocorrerem.

A mutação não uniforme múltipla consiste em aplicar a mutação não uniforme em todos os genes do cromossoma seleccionado.

Mutação c*reep*

Na mutação *creep*, acrescenta-se, ou subtrai-se, um pequeno número aleatório obtido de uma distribuição normal $N(0, \sigma)$, onde o desvio padrão σ assume um valor pequeno, ou de uma distribuição uniforme. Alternativamente, a mutação *creep* pode ser realizada, multiplicando o gene por um número aleatório próximo da unidade.

3.4.7.7. MUTAÇÃO PARA CODIFICAÇÃO POR PERMUTAÇÃO

Na codificação por permutação deve-se ter em atenção que qualquer mutação a ser aplicada deve gerar uma estrutura de dados que também seja uma permutação. Por outras palavras, a estrutura deve ser uma sequência ordenada de todos os símbolos, não sendo permitidas repetições dos símbolos. Por exemplo, se for utilizada uma mutação de reinício aleatório nos genes, provavelmente o cromossoma gerado não será uma permutação. Os operadores descritos a seguir preservam a estrutura de dados.

Mutação baseada na posição (*Position-Based Mutation*)

Na mutação baseada na posição (Syswerda 1991) são escolhidos aleatoriamente a posição de um gene e um ponto do cromossoma. De seguida o gene escolhido é inserido no ponto seleccionado, causando uma perturbação na sequência dos elementos. A figura 3.19 pretende descrever o procedimento da mutação baseada na posição.

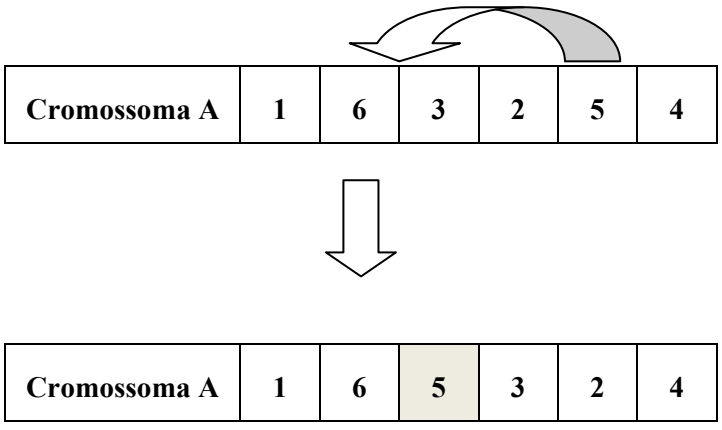


Figura 3.19 Exemplo da mutação baseada na posição

Mutação de mistura (*Scramble Mutation*)

A mutação de mistura (Syswerda 1991) escolhe aleatoriamente dois pontos do cromossoma de modo a formar um subconjunto de genes. Em seguida, misturam-se aleatoriamente os genes do subconjunto seleccionado. A figura 3.20 ilustra um exemplo da mutação de mistura.

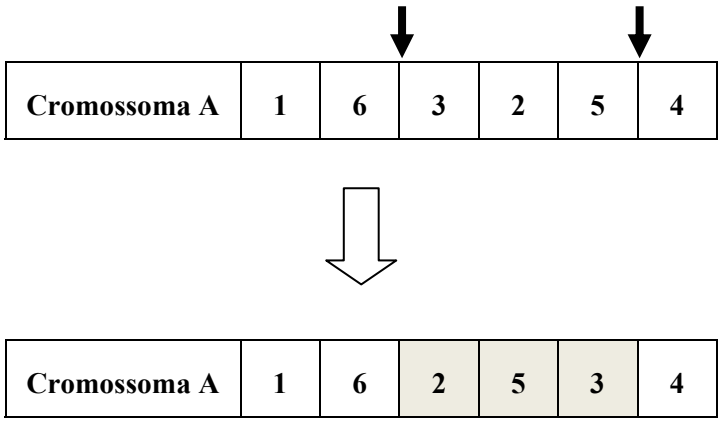


Figura 3.20 Exemplo da mutação de mistura

Mutação baseada em ordem (*Order-Based Mutation*)

Na mutação baseada em ordem (Syswerda 1991) são seleccionadas duas posições aleatoriamente e os seus genes são trocados. Este operador preserva a maior parte da adjacência da informação. Um exemplo da mutação baseada em ordem está representado na figura 3.21.

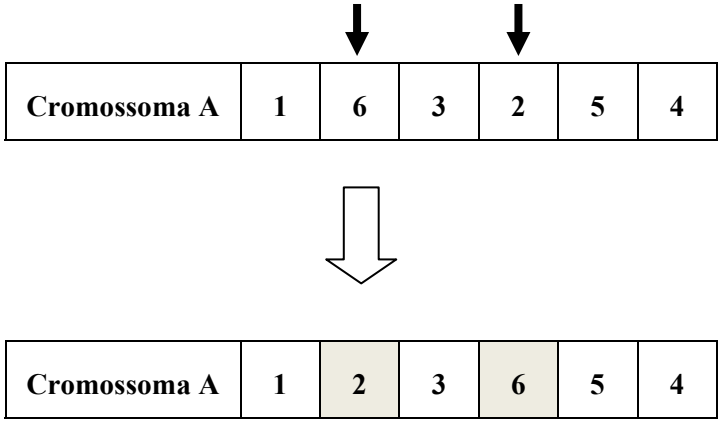


Figura 3.21 Exemplo da mutação baseada em ordem

Mutação de inversão

O operador de mutação de inversão (Lin e Kernighan 1973) escolhe aleatoriamente dois pontos do cromossoma e inverte a ordem dos genes entre os pontos seleccionados. Na figura 3.22 apresenta-se um exemplo da mutação de inversão.

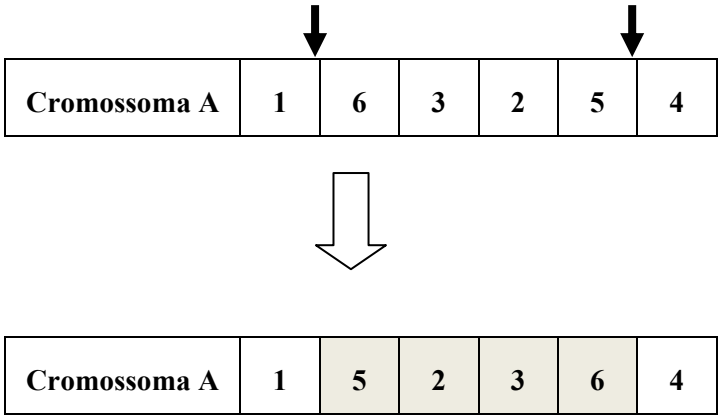


Figura 3.22 Exemplo da mutação de inversão

3.4.7.8. MUTAÇÃO PARA CODIFICAÇÃO EM ÁRVORE

Quando é efectuada a mutação para codificação em árvore, é escolhida aleatoriamente uma função que sofrerá a mutação. Nessa altura, essa função sofrerá uma mutação que poderá ser a nível local de um nó da árvore, ou modificando um ramo da mesma, escolhido de forma aleatória.

A figura 3.23, exemplifica uma mutação efectuada apenas a nível de um nó da árvore (nó 3). Neste caso a operação *OR* é substituída pela operação *XOR*, mantendo-se igual a restante estrutura da árvore. *XOR* representa a operação booleana *ou* exclusivo.

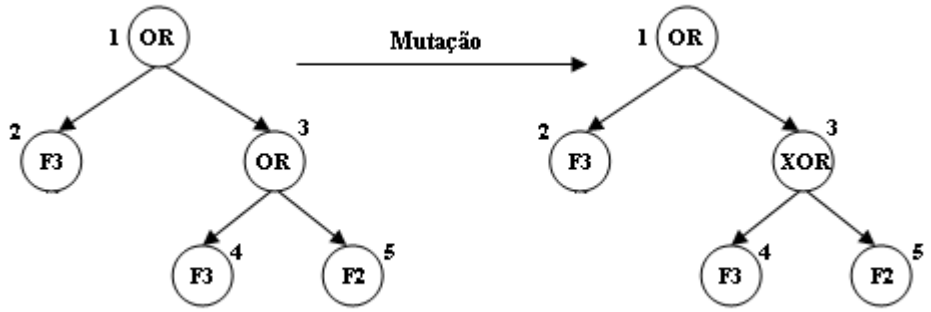


Figura 3.23 Exemplo da operação mutação no nó 3 de uma árvore

Um outro tipo de mutação consiste em escolher aleatoriamente um ramo da árvore, retirá-lo da mesma e substituí-lo por um outro ramo gerado pelo algoritmo para esse efeito. Este tipo de mutação é capaz de modificar a estrutura da árvore, dado que não obriga a que o novo nó tenha o mesmo tamanho daquele que foi retirado. A figura 3.24 pretende ilustrar o processo que foi descrito anteriormente.

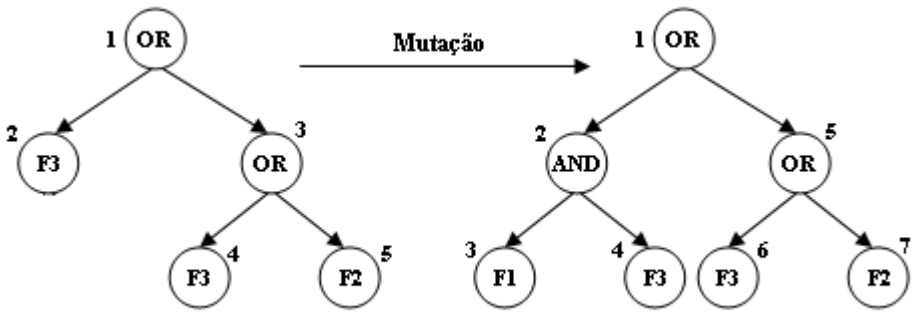


Figura 3.24 Exemplo da operação mutação no ramo do nó 2 de uma árvore

3.4.8. SUBSTITUIÇÃO DA POPULAÇÃO

Este processo controla a percentagem da população actual que será substituída pela nova população. Como a população é finita, deve-se adoptar um critério de substituição. Existem dois tipos básicos de substituição de cromossomas: substituição geracional e substituição em estado estacionário (*steady-state*).

Substituição geracional

Denomina-se substituição geracional ao processo de substituição de todos os cromossomas de uma população pelos seus descendentes a fim de compor a nova população. Nesta situação o número de descendentes gerado é igual ao número de indivíduos existentes na população anterior. No AG geracional, indivíduos de gerações diferentes não têm qualquer convivência (nunca ocorre o "incesto"). O problema deste tipo de substituição é que se corre o risco de perder material genético de boa qualidade. Assim, pode-se utilizar uma substituição geracional com elitismo, de modo a que seja introduzido o melhor indivíduo, ou um conjunto dos melhores indivíduos, da geração actual na geração seguinte.

Substituição em estado estacionário (*steady-state)*

Na substituição em estado estacionário (Davis 1991), são criados apenas dois (ou um) filhos em cada geração, que substituem os dois piores cromossomas da população. Alternativamente, os dois filhos podem substituir os pais ou os dois cromossomas mais velhos da população, com base na suposição de que um cromossoma existente na população há muitas gerações, já transmitiu os seus genes à população. Um dos problemas deste procedimento é o facto de que, em alguns casos, os indivíduos gerados são exactamente iguais a outros já existentes. No cruzamento entre um pai e um filho, por exemplo, existe uma elevada probabilidade que ocorra este tipo de problema. Para evitar este problema, o modelo *steady-state* sem duplicação, segue as regras da substituição *steady-state* simples, mas só aceita a substituição se o candidato descendente for diferente dos cromossomas existentes na população actual.

3.4.9. CONDIÇÃO DE CONCLUSÃO DO ALGORITMO

A condição de conclusão do AG depende do problema em causa e do esforço computacional que é exigido. Em face do tempo requerido e dos recursos disponíveis, é necessário definir qual a qualidade da solução que se pretende. Uma condição utilizada com frequência consiste em definir *a priori* o número máximo de gerações em que a evolução deve ocorrer. Uma segunda condição possível passa pela definição de um valor mínimo para o desvio padrão do valor de aptidão dos indivíduos na população. Uma vez atingido esse valor mínimo o algoritmo pára. Ainda uma outra condição bastante comum

de paragem consiste em fazer evoluir o algoritmo até se verificar que não se registam melhorias significativas das soluções ao longo de um dado número de gerações. No entanto, se for possível avaliar a qualidade das soluções encontradas, a condição de conclusão pode ser o de encontrar uma "boa" solução.

3.4.10. TRATAMENTO DE RESTRIÇÕES

Como já foi referido anteriormente, na prática, muitos problemas possuem restrições, isto é, existem valores que determinadas variáveis não podem assumir porque violam alguma característica importante do problema. Por isso, os AGs, devem tentar encontrar a solução com o melhor valor para a função de avaliação, do conjunto de soluções que satisfazem às restrições do problema. Quando uma restrição envolve somente o valor de uma das variáveis do problema, ela pode ser facilmente incorporada num AG. Entretanto, se uma restrição precisar ser expressa como uma função de duas ou mais variáveis, sua incorporação no AG pode ser bastante difícil.

Existem diversas estratégias para se lidar com as restrições que envolvem funções de mais de uma variável como, por exemplo a Rejeição, o Reparo, a Penalização e os Operadores genéticos especiais, que vamos descrever a seguir.

Rejeição

Na estratégia de rejeição, os indivíduos que não são considerados válidos segundo alguma restrição são simplesmente substituídos por outros indivíduos, gerados aleatoriamente ou por reprodução. Geralmente, este método é de fácil implementação, mas possui alguns problemas. Em primeiro lugar, verificar a validade de cada indivíduo pode ter um custo computacional muito alto, dependendo do problema. Em segundo lugar, pode acontecer que a população fique, em algum momento, quase toda preenchida por indivíduos não válidos, o que pode distorcer a evolução do AG, dando origem a problemas de convergência do algoritmo. Por último, esta estratégia pode desperdiçar material genético potencialmente útil, uma vez que indivíduos não admissíveis podem estar próximos da solução óptima.

Reparo

A segunda abordagem possível é o reparo. Nesta estratégia, os descendentes gerados são analisados, e aqueles que forem considerados inválidos são modificados de modo a ficarem de acordo com as restrições. Este método só serve para alguns problemas de optimização combinatória, sendo a sua implementação muito difícil, ou mesmo impossível em outros tipos de aplicações. Quando pode ser utilizada, a implementação desta técnica é muito específica para cada problema, pois depende completamente daquela que é considerada uma solução válida no contexto do problema. Este método também tende a aumentar o custo computacional do AG, em função do esforço adicional para a análise das soluções.

Penalização

A penalização consiste em diminuir artificialmente a aptidão das soluções consideradas inválidas, adicionando a estas uma penalização. Desta forma, mantém-se baixa a probabilidade de encontrar soluções inválidas na população, mas permite-se que estas sejam eventualmente geradas, possibilitando ao AG caminhar por regiões proibidas, o que pode ser vantajoso no caso de espaços de pesquisa não convexos. Esta estratégia é a mais utilizada na prática, dado que é fácil de implementar e pode ser adoptada em praticamente qualquer aplicação de AGs. O principal problema desta estratégia é que tem que se definir as penalizações que serão aplicadas aos indivíduos inválidos. A desvantagem deste método surge quando o problema tem muitas restrições pois, nesse caso, encontrar uma solução admissível torna-se tão difícil como encontrar a melhor solução.

Operadores genéticos especiais

Uma técnica frequentemente utilizada é o desenvolvimento de operadores genéticos especiais. Estes operadores efectuam o cruzamento e a mutação de forma a produzir sempre indivíduos válidos, dispensando a necessidade de qualquer tipo de tratamento para as restrições. Quando pode ser executada, esta é considerada a melhor abordagem para o tratamento de restrições, pois evita todos os problemas que as outras abordagens possuem. No entanto, a sua implementação pode ser extremamente complicada e, frequentemente, impossível.

3.5. PROBLEMAS TEÓRICOS E PRÁTICOS COM UM AG

3.5.1. TEOREMA DOS ESQUEMAS

A teoria dos esquemas, desenvolvida principalmente por Holland (Holland 1975), tenta descrever e justificar o funcionamento dos AGs, baseando-se nas características do Algoritmo Genético Simples (AGS). A motivação inicial de Holland ao propor esta teoria foi a de que problemas complexos poderiam ser resolvidos mais facilmente se fossem divididos em subproblemas mais simples.

O teorema dos esquemas parte do pressuposto que o AG utiliza uma população de cromossomas binários, selecção proporcional à aptidão, cruzamento com um único ponto de corte e mutação por mudança aleatória dos alelos.

Jonh Holland constatou que os AGs manipulam determinados segmentos da cadeia de bits. Tais segmentos foram por ele denominados de esquemas. Um esquema é formado pelos símbolos 0, 1 e *. A ocorrência do símbolo * numa posição no esquema significa que esta posição pode ser ocupada pelo símbolo 1 ou 0. Assim, se forem consideradas sequências de 5 caracteres numa codificação binária, pode-se definir, por exemplo, o esquema {000**}, que inclui as sequências {00000, 00001, 00010, 00011}, que podem ser vistas como instâncias do esquema {000**}. Esta definição de esquemas inclui também o esquema {*****}, que engloba todas as sequências de 5 caracteres possíveis, e esquemas como {10011}, {11110}, etc. que possuem uma única instância. Também é importante notar que um determinado vector pertence a vários esquemas ao mesmo tempo. Por exemplo, o vector {00011} pertence ao esquema {000**}, ao esquema {***11}, ao esquema {0***1} e a muitos outros. Um esquema representa 2^k sequências, onde k é o número de símbolos *. Por outro lado, um vector com comprimento l pode ser representado por 2^l esquemas.

Um esquema possui três atributos: ordem, comprimento e aptidão. A ordem, $o(H)$, é o número de posições fixas, que define o esquema, e o comprimento, $\delta(H)$, é a distância entre a primeira e a última posição fixa. Assim 010** e 1**10 são ambos esquemas de ordem 3, mas enquanto 010** tem comprimento 2, o esquema 1**10 tem $\delta(H) = 4$.

Análise do processo de selecção na teoria dos esquemas

O processo de selecção escolhe um indivíduo de acordo com a sua aptidão. Considerando uma população com N cromossomas, um indivíduo X_i com aptidão $f(X_i)$ será escolhido com uma probabilidade:

$$p(X_i) = \frac{f(X_i)}{\sum_{j=1}^{N} f(X_j)} \tag{3.8}$$

Pensando agora em termos de esquemas, se $m(H,t)$ designar o número de instâncias do esquema H numa população de N indivíduos, no tempo t, então é possível calcular o número provável de representantes de H na geração seguinte como:

$$m(H, t+1) = N \frac{\sum_{X \in H} f(X)}{\sum_{j=1}^{N} f_j} \tag{3.9}$$

Definindo $f(H)$ como a aptidão média do esquema H:

$$f(H) = \frac{\sum_{X \in H} f(X)}{m(H,t)} \tag{3.10}$$

Assim, pode-se reescrever $m(H, t+1)$ através da equação seguinte:

$$m(H, t+1) = m(H,t)\, N \frac{f(H)}{\sum_{j=1}^{N} f_j} \tag{3.11}$$

Sabendo que a aptidão média da população é dada por:

$$\bar{f} = \frac{\sum_{j=1}^{N} f_j}{N} \tag{3.12}$$

Pode-se fazer uma última alteração na fórmula de $m(H, t+1)$:

$$m(H, t+1) = m(H,t) \frac{f(H)}{\bar{f}} \tag{3.13}$$

Desta equação decorre um princípio bastante simples: esquemas com aptidão acima da média tendem a aumentar o número de instâncias ao longo do tempo. Esta variação ainda pode ser tornada mais precisa. Admita-se que a aptidão média do esquema varia de uma constante C relativamente à qualidade média da população. Então:

$$m(H,t+1) = m(H,t)\,\frac{\bar{f} + C\bar{f}}{\bar{f}} = \; m(H,t)\,(1+C) \tag{3.14}$$

Esta recorrência pode resolver-se por iteração vindo:

$$m(H,t) = m(H,0)\,(1+C)^t \tag{3.15}$$

O que significa que a selecção atribui um número exponencialmente crescente (ou decrescente) de instâncias de esquemas acima (abaixo) da média.

Análise do operador de cruzamento na teoria dos esquemas

De seguida é ilustrado o efeito que o cruzamento introduz no número de esquemas esperados na geração seguinte.

Para entender como um esquema é afectado pelo cruzamento, suponha-se que se tem um cromossoma A, com $l = 7$, e dois esquemas H_1 e H_2, ambos com ordem igual a 2, mas com comprimentos $\delta(H_1) = 5$ e $\delta(H_2) = 1$, como mostrado a seguir:

A = 011|1000

H_1=*1*|***0

H_2=***|10**

O ponto de corte no cruzamento é simplesmente uma escolha aleatória entre 1 e $l-1$. É claro que, se o cromossoma A for escolhido para o cruzamento, o esquema H_2 tem maior probabilidade de sobreviver ao cruzamento, pois este esquema só será desfeito se o corte acontecer entre as posições 4 e 5. Por outro lado, H_1 pode ser destruído se o corte ocorrer entre as posições 2 e 7. Logo, a probabilidade de destruição p_d de H_2 é 1/6, enquanto que para H_1 se tem $p_d = 5/6$. Generalizando,

$$p_d(H) = \frac{\delta(H)}{l-1} \tag{3.16}$$

Tem-se então, que a probabilidade de sobrevivência, p_s, do esquema H é dada por:

$$p_s(H) = 1 - \frac{\delta(H)}{l-1} \tag{3.17}$$

Sendo p_c a probabilidade de cruzamento, então vem:

$$p_s(H) = 1 - p_c \frac{\delta(H)}{l-1} \tag{3.18}$$

Se o cruzamento é feito entre dois cromossomas representantes do mesmo esquema *H*, esse esquema nunca será destruído, mesmo que o ponto de cruzamento seja seleccionado entre posições fixas do esquema. Deste modo,

$$p_s(H) \geq 1 - p_c \frac{\delta(H)}{l-1} \tag{3.19}$$

De onde se conclui que os esquemas com comprimento baixo têm maior probabilidade de sobrevivência após o cruzamento.

Análise do operador de mutação na teoria dos esquemas

Considerando a probabilidade de ocorrer uma mutação p_m, a probabilidade de alterar um *bit* é p_m e a probabilidade do bit sobreviver é p_s=1−p_m. Assim, como cada mutação é estatisticamente independente, a probabilidade de sobrevivência de um esquema, será o produto das probabilidades para cada posição fixa, ou seja a probabilidade de sobrevivência de um esquema com ordem $o(H)$ é:

$$p_s(H) = (1 - p_m)^{o(H)} \tag{3.20}$$

Como $p_m << 1$ a equação anterior pode ser aproximada por

$$p_s(H) \approx 1 - p_m \, o(H) \tag{3.21}$$

Desta expressão podemos concluir que os esquemas de baixa ordem têm maior probabilidade de não serem destruídos pelo operador mutação.

Combinando a selecção, o cruzamento e a mutação, obtém-se a expressão:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1}\right] [1 - p_m \times o(H)] \tag{3.22}$$

Que traduz o **Teorema dos Esquemas**, também chamado por Holland de **Teorema Fundamental dos Algoritmos Genéticos,** que de uma forma simples se pode resumir da seguinte forma: o número de instâncias de um determinado esquema cresce exponencialmente ao longo do tempo se estes possuírem três características – aptidão acima da média, comprimento pequeno e ordem baixa.

Os esquemas de ordem baixa, comprimento pequeno e aptidão média alta, são favorecidos na execução de um AGS, e são denominados blocos construtores. Baseia-se nestes blocos a **Hipótese dos Blocos Construtores**, segundo a qual a operação básica do AGS é descobrir e juntar blocos construtores, de modo a obter sequências que pertençam simultaneamente a muitos esquemas de alta aptidão. Assume-se então que indivíduos formados por vários blocos construtores têm necessariamente uma elevada aptidão e, portanto, resolvem bem o problema.

Outro aspecto importante é conhecido por **Paralelismo Implícito**. Admita-se que se trabalha com uma população de N indivíduos, cada um deles de comprimento l. Cada indivíduo da população é instância de 2^l esquemas. Deste modo, em toda a população haverá no máximo $N \times 2^l$ esquemas. De acordo com John Holland, o número de esquemas processados num AG com uma população de N indivíduos é proporcional a N^3.

3.5.2. PROBLEMAS ENGANADORES

Existem problemas onde esquemas de alta aptidão média, quando juntos numa mesma instância, produzem uma solução muito má. Estes problemas são chamados de problemas enganadores (*deceptive*), e têm sido muito pesquisados, pois podem ajudar a entender melhor o funcionamento dos AGs. O chamado Problema Enganador Mínimo pode ser usado para exemplificar o fenómeno. Nele, tem-se que o cromossoma 11 é a solução óptima, mas os esquemas *0 e 0* tem maior aptidão média que respectivamente os esquemas *1 e 1*. Como, segundo a teoria dos blocos construtores, o AGS tentaria sempre juntar os esquemas complementares de maior aptidão, 0* e *0, a tendência seria sempre formar o vector 00 como resposta, e não o vector 11. É claro que, no caso específico do Problema Enganador Mínimo, é muito fácil para o AGS descobrir a resposta, pois é grande a probabilidade de que o cromossoma 11, que tem poucas posições, seja formado por acaso. Porém, podem ser formulados problemas enganadores maiores, e estes apresentam grande dificuldade para serem solucionados por AGs.

Os problemas enganadores podem ser contornados utilizando os chamados AGs desordenados (AGs *messy*) (Goldberg, Deb e Korb 1991), ou aplicando-se transformações no espaço de pesquisa, que desfaçam as características de um problema enganador.

3.5.3. BALANÇO *EXPLORATION-EXPLOITATION*

Um algoritmo de optimização eficiente deve usar estas duas técnicas para encontrar o valor óptimo global da função de avaliação (Beasley, Bull e Martin 1993). Podemos traduzir a palavra inglesa *exploration* como exploração, no sentido de procurar algo novo. A palavra *exploitation* também seria traduzida como exploração, só que no sentido de utilizar algum recurso disponível. Como as duas palavras seriam traduzidas da mesma forma a expressão perderia o sentido. Esta expressão não é exclusiva ou originária dos trabalhos com AGs, mas neste contexto ela refere-se ao dilema de como os AGs devem dividir seus "esforços" entre investigar novas áreas do espaço de busca (*exploration*) e intensificar a busca em regiões que se mostraram promissoras (*exploitation*). Se não fizermos suficiente *exploitation*, a pesquisa será muito desordenada aproximando-se de um processo de pesquisa aleatória. Se, por outro lado, negligenciarmos a necessidade de fazer *exploration*, o AG poderá convergir rapidamente para um máximo, mas perderá capacidade de pesquisa global. O equilíbrio entre estas duas actividades é regulado pelas operações de selecção, cruzamento e mutação, e é fundamental um balanço adequado entre elas para que o AG tenha sucesso. O cruzamento e a mutação são dois mecanismos de pesquisa dos AGs que levam à exploração de pontos inteiramente novos do espaço de pesquisa (*exploration*). Por sua vez, a selecção dirige a pesquisa em direcção aos melhores pontos do espaço de pesquisa (*exploitation*).

3.5.4. CONVERGÊNCIA PREMATURA

Quando, num número relativamente pequeno de gerações, a aptidão média da população não se altera significativamente e, além disso as soluções obtidas são de baixa qualidade, pode ter ocorrido uma convergência prematura. Este problema está associado à perda de diversidade da população. Por outras palavras, os indivíduos da população são tão parecidos entre si que fica difícil obter soluções diferentes. É comum que ocorra este fenómeno quando se usa o AG para optimizar funções multimodais. De facto, pode acontecer que, na fase inicial do AG, alguns indivíduos da população inicial estejam muito próximo de máximos locais, tendo aptidões muito altas em relação ao resto da população.

Tais indivíduos podem obter muitas cópias no processo de selecção e dominar a população rapidamente, acabando com a sua diversidade e impedindo que o AG prossiga a busca por uma solução de melhor qualidade. Outra causa da convergência prematura deve-se ao desvio genético (*genetic drift*). Este fenómeno consiste no desaparecimento de determinados genes da população no decorrer das gerações. Deste modo, o AG fica impossibilitado de explorar completamente o espaço de busca, acabando por convergir para um mínimo ou máximo local. O desvio genético pode ser minimizado utilizando taxas adequadas de mutação, de modo a conduzir a uma boa diversidade de genes da população a fim de contrabalançar as perdas de genes.

3.5.5. CONVERGÊNCIA LENTA

A convergência lenta representa o problema oposto à convergência prematura. Depois de muitas gerações, a população quase convergiu, porém, o máximo global não foi encontrado. A aptidão média é elevada, e a diferença entre o melhor e o pior indivíduo é pequena. Contudo, não há diversidade suficiente nos valores de aptidão para localizar um máximo global.

3.5.6. TEMPO DE PROCESSAMENTO ELEVADO

Grande parte das aplicações de AGs requer um elevado tempo de processamento. Muitas vezes, o problema que se deseja optimizar é complexo, tornando demorada a avaliação de todos os indivíduos da população (uma avaliação para cada indivíduo da população, a cada geração). Outro factor que influencia o tempo de processamento é o espaço de pesquisa. Se o espaço de pesquisa for muito grande, então requer grandes populações e, consequentemente, o AG coloca um peso computacional mais elevado.

Para este tipo de problema não existe solução, a não ser a utilização de processamento paralelo na execução dos AGs. De facto, a estrutura dos AGs torna muito vantajoso o uso de máquinas paralelas, pois eles podem ser facilmente adaptados tal que o cálculo das aptidões, que é a parte mais pesada, seja dividido entre vários processadores.

As operações de cruzamento e de mutação são também independentes e podem ser feitas em paralelo. Com o intuito de explorar estas propriedades, vários modelos para implementação de AGs paralelos têm sido propostos na literatura (Cantú-Paz 2001).

3.6. VANTAGENS DOS ALGORITMOS GENÉTICOS

Os AGs diferem dos outros algoritmos de optimização e pesquisa nos seguintes pontos:

- Os AGs trabalham com os parâmetros do problema codificados em vez do uso directo dos parâmetros;
- Os AGs pesquisam um conjunto de pontos, e não apenas um ponto o que permite realizar buscas simultâneas em vários espaços de pesquisa;
- Os AGs utilizam informação de troca (funções de avaliação), ao contrário do uso de derivadas e outros conhecimentos auxiliares;
- Os AGs usam regras de transição estocásticas em vez de regras determinísticas.

Algumas vantagens dos AGs são:

- Funcionarem tanto com parâmetros contínuos como com parâmetros discretos ou, então, com uma combinação dos dois tipos;
- Não ser necessário conhecimento matemático aprofundado do problema considerado;
- Optimizarem um número elevado de variáveis;
- Serem fáceis de implementar em computador;
- Serem modulares e portáteis, no sentido que o mecanismo de evolução é independente da representação particular do problema considerado. Assim, eles podem ser transferidos de um problema para outro;
- Serem flexíveis para trabalhar com restrições arbitrárias e optimizar múltiplas funções com objectivos conflituosos;
- Serem também facilmente combinados com outras técnicas e heurísticas.

Apesar destas vantagens, os AGs são bastante lentos: normalmente estão ainda a avaliar a população inicial quando métodos clássicos já encontraram a solução. Porém, desde que se possa efectuar o cálculo sem restrições severas de tempo, o tempo mais longo é aceitável. O principal campo de aplicação dos AGs encontra-se em problemas complexos, com múltiplos mínimos/máximos, para os quais não existe um algoritmo de optimização específico.

4. CÁLCULO DE DERIVADAS FRACCIONÁRIAS ATRAVÉS DE ALGORITMOS GENÉTICOS

4.1. INTRODUÇÃO

O Cálculo Fraccionário (CF) representa a generalização do cálculo integral e diferencial para valores não inteiros e, nas últimas décadas, a sua aplicação tem verificado um grande desenvolvimento nas áreas da física e da engenharia.

Os aspectos fundamentais da teoria do CF (para os aspectos fundamentais ver Apêndice 1) e o estudo das suas propriedades podem ser consultados nas referências (Oldham e Spanier 1974), (Samko, Kilbas e Marichev 1993), (Miller e Ross 1993) e (Podlubny 1999). Apesar do CF remontar a Leibniz, só recentemente se verificou um progresso significativo na sua aplicação. Assim, pode-se mencionar um elevado número de publicações sobre viscoelasticidade, biologia, electrónica, processamento de sinais, difusão e propagação de ondas, modelação e controlo (Bagley e Torvik 1983), (Oustaloup 1991), (Anastasio 1994), (Mainardi 1996), (Machado 1997), (Nigmatullin 2006), (Tarasov e Zaslavsky 2006), (Sabatier, Agrawal e Tenreiro Machado 2007), (Hedrih 2008) e (Baleanu 2009). No

entanto, o CF é ainda considerado uma ferramenta matemática "exótica" e sua adopção requer alguns esforços no sentido do desenvolvimento de algoritmos claros.

Uma das razões para esta situação deve-se à complexidade dos algoritmos envolvidos no cálculo da Derivada Fraccionária (DF). A generalização do operador integrodiferencial exige a adopção de aproximações de funções irracionais através de séries ou expansões de fracções racionais (Podlubny 1999), (Machado 2001), (Chen e Moore 2002), (Tseng 2001), (Vinagre, Chen e Petras 2003), (Chen e Vinagre 2003), (Barbosa, Machado e Silva 2006) e (Al-Alaoui 1993). Embora o principal número de contribuições se tenha centrado na obtenção de esquemas de expansão, ainda não se obteve um procedimento de optimização sistemático.

Nesta linha de pensamento, o presente trabalho aborda o cálculo de expressões de ordem fraccionária e está organizado da seguinte forma. Na secção 4.2 é introduzido o cálculo da DF e é optimizada, através de AGs, a aproximação de expressões analíticas irracionais com base em fracções racionais. Na secção 4.3 apresenta-se um conjunto de experiências que demonstram a eficácia do método de optimização proposto.

4.2. FORMULAÇÃO DO PROBLEMA E FERRAMENTAS ADOPTADAS

Esta secção apresenta os principais conceitos matemáticos e algoritmos utilizados neste trabalho. Inicialmente é formulada a definição da DF, adoptada em conjunto com as regras de discretização para cálculo em tempo real. Em seguida, embora já tenham sido abordados no capítulo 3, são descritos os aspectos fundamentais subjacentes ao esquema de optimização do AG.

4.2.1. EXPRESSÕES DE ORDEM FRACCIONÁRIA

Desde o início do desenvolvimento do cálculo diferencial a generalização do conceito de derivada e integral de valores não inteiros α tem sido objecto de várias abordagens, como as definições de Riemann-Liouville, de Grünwald-Letnikov, de Caputo e de Fourier/Laplace.

A definição de Grünwald-Letnikov de uma DF de ordem α do sinal $x(t)$, $D^{\alpha}x(t)$, é dada por:

$$D^{\alpha}x(t) = \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{k=0}^{\infty} \frac{(-1)^k \, \Gamma\,(\alpha+1) x(t-kh)}{\Gamma(k+1)\,\Gamma(\alpha-k+1)} \tag{4.1}$$

onde Γ representa a função Gama e *h* é o incremento no tempo. Esta formulação inspira um algoritmo de cálculo discreto no tempo, baseado na aproximação do incremento no tempo *h* através do período de amostragem *T*, resultando a equação no domínio *z*:

$$Z\{D^{\alpha}x(t)\} \approx \left[\frac{1}{T^{\alpha}} \sum_{k=0}^{\infty} \frac{(-1)^k \, \Gamma\,(\alpha+1)}{k!\;\Gamma(\alpha-k+1)}\; z^{-k}\right] X(z) \tag{4.2}$$

onde $X(z) = Z\{x(t)\}$.

A implementação da expressão (4.2) corresponde a uma série truncada de *r* termos dada por:

$$Z\{D^{\alpha}x(t)\} \approx \left[\frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^k \, \Gamma\,(\alpha+1)}{k!\;\Gamma(\alpha-k+1)}\; z^{-k}\right] X(z) \tag{4.3}$$

A expressão (4.2) representa a aproximação de Euler (ou equação às diferenças em atraso de primeira ordem) no esquema de conversão denominado $s \rightarrow z$. Outra possibilidade, frequentemente adoptada no projecto de sistemas de controlo, consiste na regra de Tustin (ou bilinear). As expressões racionais de Euler e Tustin, $H_0(z^{-1}) = \frac{1}{T}\,(1 - z^{-1})$ e $H_1(z^{-1}) = \frac{2}{T}\,\frac{1-z^{-1}}{1+z^{-1}}$, são muitas vezes chamadas funções geradoras, respectivamente de ordem zero e primeira ordem. Por consequência, a generalização destes métodos de conversão conduzem a resultados de ordem não inteira α:

$$s^{\alpha} \approx \left[\frac{1}{T}\,(1 - z^{-1})\right]^{\alpha} = {H_0}^{\alpha}(z^{-1}) \tag{4.4a}$$

$$s^{\alpha} \approx \left[\frac{2}{T}\;\frac{1 - z^{-1}}{1 + z^{-1}}\right]^{\alpha} = {H_1}^{\alpha}(z^{-1}) \tag{4.4b}$$

Pode-se obter uma família de diferenciadores fraccionários gerada por ${H_0}^{\alpha}(z^{-1})$ e ${H_1}^{\alpha}(z^{-1})$ ponderada pelos factores *p* e 1−*p*, resultando:

$$H_{av}[z^{-1};(p,\alpha)] = pH_0{}^{\alpha}(z^{-1}) + (1-p)H_1{}^{\alpha}(z^{-1}) \quad \textbf{(4.5)}$$

Para obter uma expressão racional, a aproximação final corresponde a uma série de Taylor truncada ou a uma expansão de fracção racional. Devido ao seu elevado desempenho muitas vezes é usada uma fracção:

$$H_k(z^{-1}) = \frac{\sum_{i=0}^{k} a_i z^{-i}}{\sum_{i=0}^{k} b_i z^{-i}}\ , a_i, b_i\ \in\ \Re \quad \textbf{(4.6)}$$

onde $k \in \aleph$ indica a ordem da aproximação. Além disso, normalmente é adoptada uma expansão de Padé na vizinhança de $z^{-1} = 0$ e, desde que um parâmetro seja linearmente dependente, é estabelecido $b_0 = 1$.

4.2.2. OPTIMIZAÇÃO ATRAVÉS DE ALGORITMOS GENÉTICOS

Como já foi referido no capítulo 3, um AG é uma técnica computacional para determinar soluções exactas ou aproximadas para problemas de optimização e de pesquisa. Os AGs são simulados num sistema de computação e consistem numa população de representações de soluções, de um problema de optimização que evolui para soluções melhores. Quando a representação genética e a função de aptidão estão definidas, o AG inicializa uma população de soluções aleatoriamente, e de seguida, aplica repetitivamente os operadores de selecção, cruzamento e de mutação, para encontrar a melhor solução.

Geralmente, a evolução começa a partir de uma população de indivíduos gerados aleatoriamente. Em cada geração, não só a aptidão de cada indivíduo na população é avaliada, mas também diversos indivíduos são estocasticamente seleccionados a partir da população actual e modificados para formar uma nova população. A nova população é então utilizada na próxima iteração do algoritmo. O AG termina quando o número máximo de gerações *N* é produzido, ou um nível satisfatório de aptidão é atingido.

Durante as sucessivas gerações, uma parte ou a totalidade da população é seleccionada para criar uma nova geração. As soluções individuais são seleccionadas através de um processo baseado na aptidão, onde as melhores soluções (medidas por uma função de avaliação) são normalmente mais susceptíveis de serem seleccionadas. O pseudocódigo de um AG é:

1. Escolher a população inicial
2. Avaliar a aptidão de cada indivíduo na população
3. Repetir
    3.1. Seleccionar os melhores indivíduos para reproduzir
    3.2. Produzir nova geração através do cruzamento e da mutação e gerar descendentes
    3.3. Avaliar a aptidão dos indivíduos da descendência
    3.4. Substituir parte da população, com aptidão mais baixa, pela descendência
4. Até condição de conclusão encontrada

4.3. DERIVADA DE ORDEM FRACCIONÁRIA

Nesta secção, vamos apresentar a aproximação por fracção para o cálculo das expressões fraccionárias. Na subsecção 4.3.1 analisamos as propriedades das fracções de Padé e, na subsecção 4.3.2, é formulado um novo método de optimização baseado num AG.

4.3.1. EXPANSÃO POR FRACÇÃO DE PADÉ

Tendo em mente as expressões (4.3) a (4.6), examina-se a aproximação de D^{α}, $\alpha = 1/2$, quando $p = 3/4$ para expansão por fracção de Padé de ordem $k = \{1, 2, 3, 4\}$. A comparação pode ser estabelecida quer no domínio do tempo, ou no domínio da frequência. Embora conceptualmente equivalente, de modo a permitir a definição de um critério simples de optimização para implementar a função de avaliação no AG, foi adoptada a resposta em frequência. Por isso, para as expressões básicas em frequência é considerada a transformação $z^{-1} = e^{-j\Omega}$, $\Omega = \omega T$, $j = \sqrt{-1}$.

A figura 4.1 representa o diagrama polar e diagramas de Bode, de amplitude e de fase, *versus* o traçado ideal para $T = 1$. Pode-se observar que os gráficos de H_k, $k = \{1, 2, 3, 4\}$, são do mesmo tipo, mas variam com a ordem *k* de aproximação. A resposta em frequência não é o objectivo da optimização visto que o critério para a expansão por fracção é simplesmente a aproximação na vizinhança de $z^{-1} = 0$. Portanto, conclui-se que o problema está mal colocado e que deve ser claramente formulado como um processo de optimização.

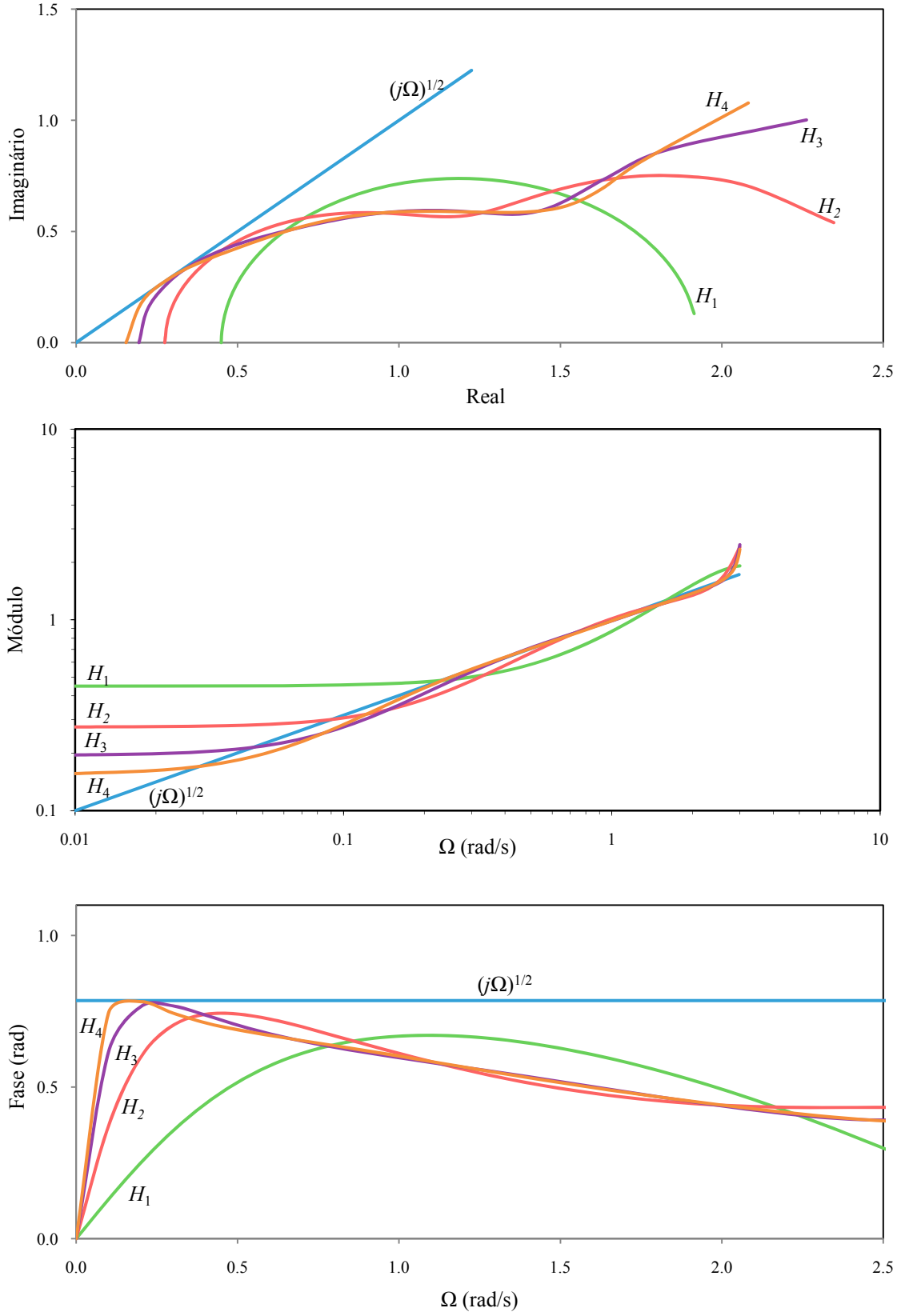


Figura 4.1 **Diagrama polar e diagramas de Bode, de amplitude e de fase, da aproximação de $D^{1/2}$ para expansão por fracção de Padé $H_k(z^{-1})$, k = {1, 2, 3, 4}, baseado nas expressões (4.5) e (4.6), com $p = 3/4$, *versus* o caso ideal $H_d(j\Omega) = (j\Omega)^{1/2}$, $0 \leq \Omega \leq 3$ rad s^{-1}, $T = 1$**

4.3.2. OPTIMIZAÇÃO DE FRACÇÕES COM ALGORITMOS GENÉTICOS

Nesta secção desenvolve-se o AG de optimização de aproximação a fracção racional para expressões de ordem fraccionária. A população do AG, com *N* indivíduos, é constituída por uma série de candidatos coeficientes da fracção $[a_0 \cdots a_k \mid b_0 \cdots b_k]$, com $b_0 = 1.0$. Na optimização, são estudados dois critérios possíveis, conduzindo a diferentes funções de avaliação:

$$J_1 = \begin{cases} \sum_{i=1}^{n} \langle \{\mathrm{Re}[H_d(\Omega_i)] - \mathrm{Re}[H_k(\Omega_i)]\}^2 + \{\mathrm{Im}[H_d(\Omega_i)] - \mathrm{Im}[H_k(\Omega_i)]\}^2 \rangle & \text{, se } H \text{ estável} \\ J_{10} & \text{, se } H \text{ instável} \end{cases} \quad \textbf{(4.7)}$$

$$J_2 = \begin{cases} \sum_{i=1}^{n} \dfrac{\{\mathrm{Re}[H_d(\Omega_i)] - \mathrm{Re}[H_k(\Omega_i)]\}^2 + \{\mathrm{Im}[H_d(\Omega_i)] - \mathrm{Im}[H_k(\Omega_i)]\}^2}{\{\mathrm{Re}[H_d(\Omega_i)] + \mathrm{Re}[H_k(\Omega_i)]\}^2 + \{\mathrm{Im}[H_d(\Omega_i)] + \mathrm{Im}[H_k(\Omega_i)]\}^2} & \text{, se } H \text{ estável} \\ J_{20} & \text{, se } H \text{ instável} \end{cases} \quad \textbf{(4.8)}$$

onde $H_d(\Omega_i)$ e $H_k(\Omega_i)$ indicam a resposta em frequência, desejada e de ordem *k*, respectivamente, da fracção à frequência Ω_i e Re[ ] e Im[ ] representam a parte real e imaginária. Para a avaliação da aptidão foram adoptados dois conjuntos alternativos de frequência de amostragem, $S_i(\Omega) = \Omega_{\min} \leq \Omega \leq \Omega_{\max}$, $i = 1, 2$, nomeadamente com variação linear (S_1) e variação logarítmica (S_2). Além disso, foi necessário aplicar neste AG restrições de modo a garantir que a função de transferência H_k era estável, pelo que a restrição utilizada foi a de uma penalização. Isto é, a avaliação verifica se a função de transferência H_k é estável, atribuindo-lhe um valor elevado J_{10} ou J_{20} para caso de instabilidade, no caso de H_k ser estável é calculado o desvio entre $H_d(\Omega_i)$ e $H_k(\Omega_i)$.

No que diz respeito às funções de avaliação, a expressão J_1 representa os pontos para a minimização do erro absoluto, ao passo que a expressão J_2 é inspirada na minimização do erro relativo. No que diz respeito aos conjuntos de frequência de amostragem, S_1 dirige-se a representações com escalas lineares, enquanto que S_2 é sugerida para representações com escalas logarítmicas.

O AG foi implementado em linguagem *Delphi*, foi utilizada a codificação real para os indivíduos da população. A tabela 4.1 indica os parâmetros genéticos utilizados na implementação do AG.

Tabela 4.1 Parâmetros genéticos utilizados no AG implementado

Parâmetros Genéticos	
Tamanho da População	1000
Número Máximo de Gerações	100000
Taxa de Cruzamento	50%
Tipo de Cruzamento	Cruzamento de um ponto
Tipo de Mutação	Mutação Uniforme
Taxa de Mutação	10%
Taxa de Substituição	Adaptativa

Na implementação do AG foi considerada a estratégia de elitismo. Verificou-se que o AG convergia mais facilmente quando a população inicial era composta por funções de transferência estáveis. Por isso, os coeficientes das fracções foram gerados por meio de uma função de distribuição de probabilidade uniforme, mas as soluções que originavam fracções instáveis eram eliminadas, sendo substituídas por novos elementos, antes da avaliação inicial do AG.

Por outro lado, adoptou-se uma taxa de substituição adaptativa, podendo variar entre 0% e 100%. Assim, após gerar os descendentes, procede-se à ordenação do conjunto {população (incluindo os pais)} + {descendentes}, passando para a geração seguinte os indivíduos com maior aptidão, sendo consequentemente, eliminados os menos aptos.

Foi verificada a avaliação da derivada de ordem $\alpha = 1/2$ através da aproximação de fracções $H_k, k = \{1, 2, 3\}$, com $T = 1$. Além disso, foram adoptados 30 pontos de amostragem (n = 30) nos intervalos $S_1(\Omega) \equiv \{\Omega_{\min}, \Omega_{\max}\} \equiv \{0, 3\}$ [rad/s] ou $S_2(\Omega) \equiv \{\Omega_{\min}, \Omega_{\max}\} \equiv \{0{,}001, 1\}$ [rad/s] e considerou-se $J_{10} = J_{20} = 9000$.

As figuras 4.2 a 4.7 representam a resposta em frequência, nomeadamente os diagramas polares e diagramas de Bode, de amplitude e de fase, para todas as combinações da função de avaliação e conjuntos de frequência, *versus* a resposta em frequência ideal.

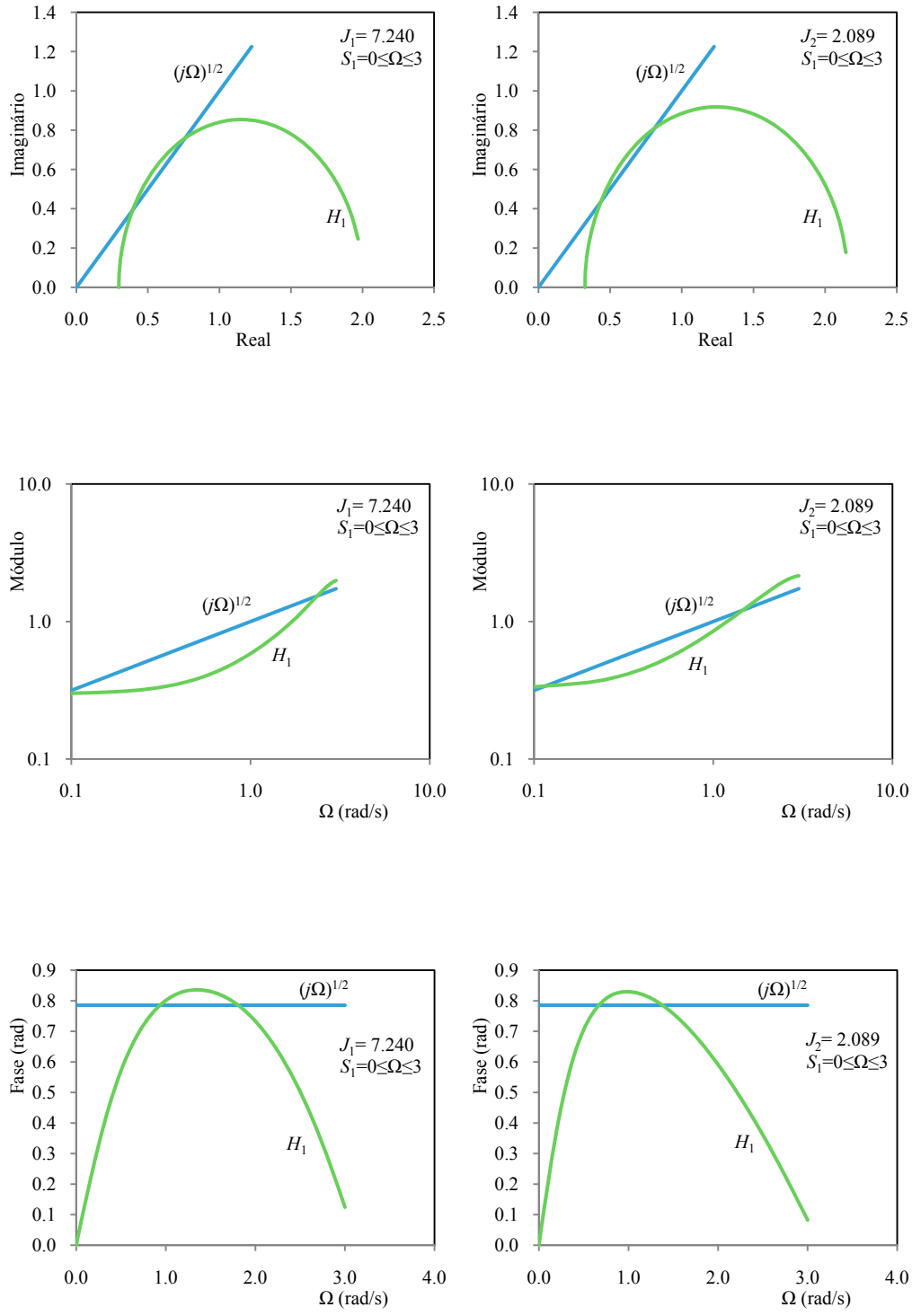


Figura 4.2 **Diagrama polar e diagramas de Bode, de amplitude e de fase, para aproximação por fracções de ordem $k = 1$, $\{J_1, J_2\} \times \{S_1\}$**

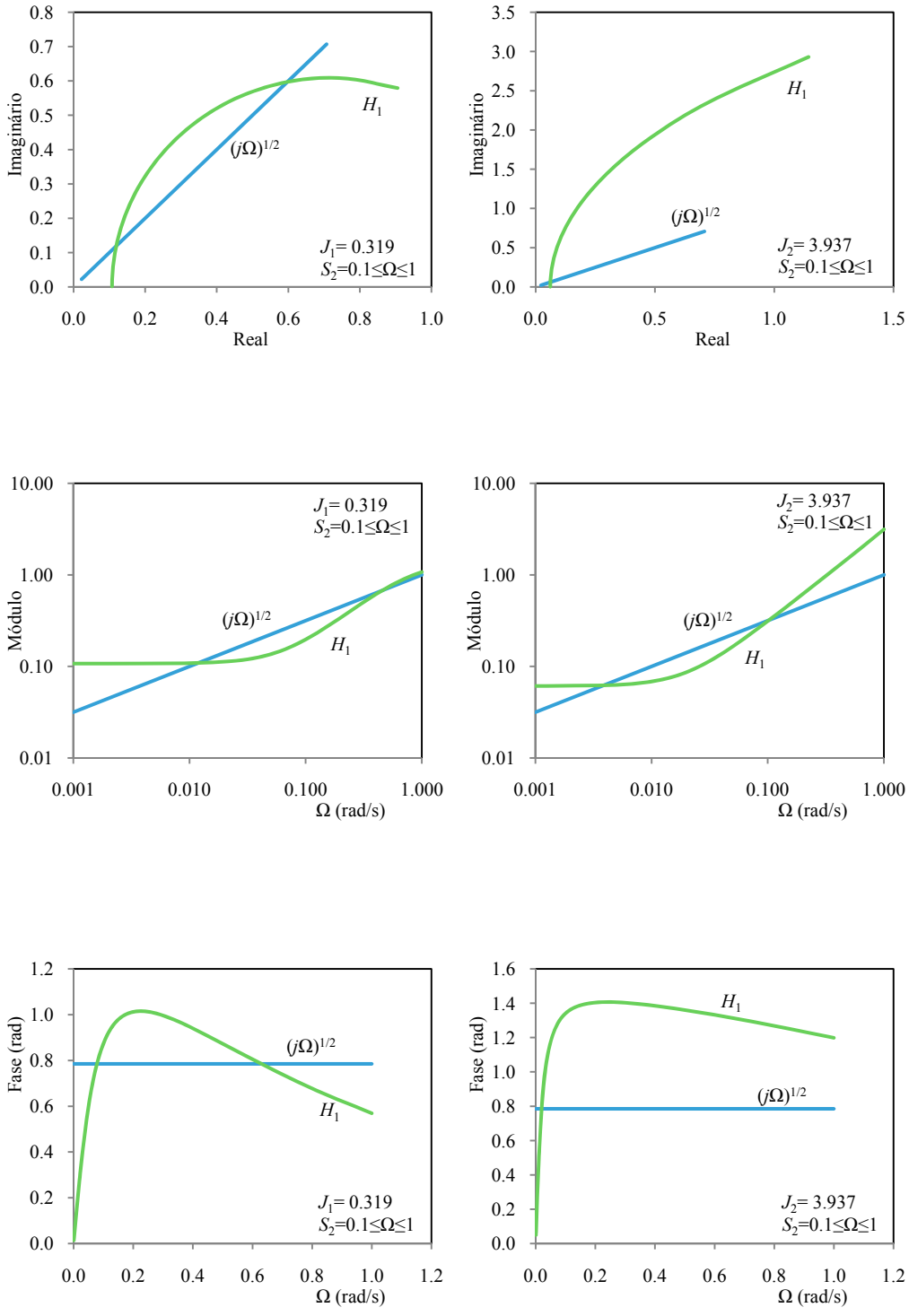


Figura 4.3 **Diagrama polar e diagramas de Bode, de amplitude e de fase, para aproximação por fracções de ordem $k = 1$, $\{J_1, J_2\} \times \{S_2\}$**

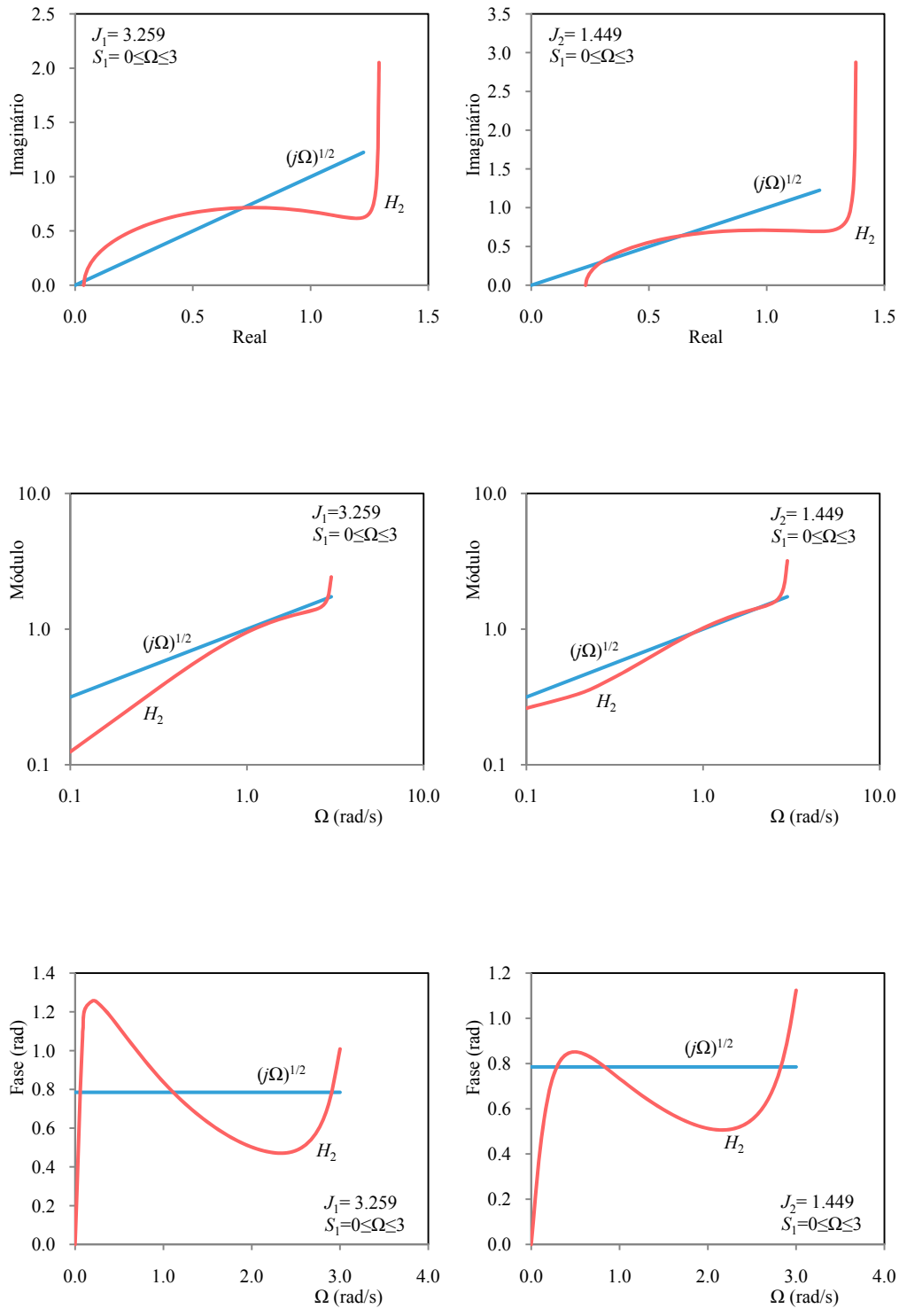


Figura 4.4 **Diagrama polar e diagramas de Bode, de amplitude e de fase, para aproximação por fracções de ordem $k = 2$, $\{J_1, J_2\} \times \{S_1\}$**

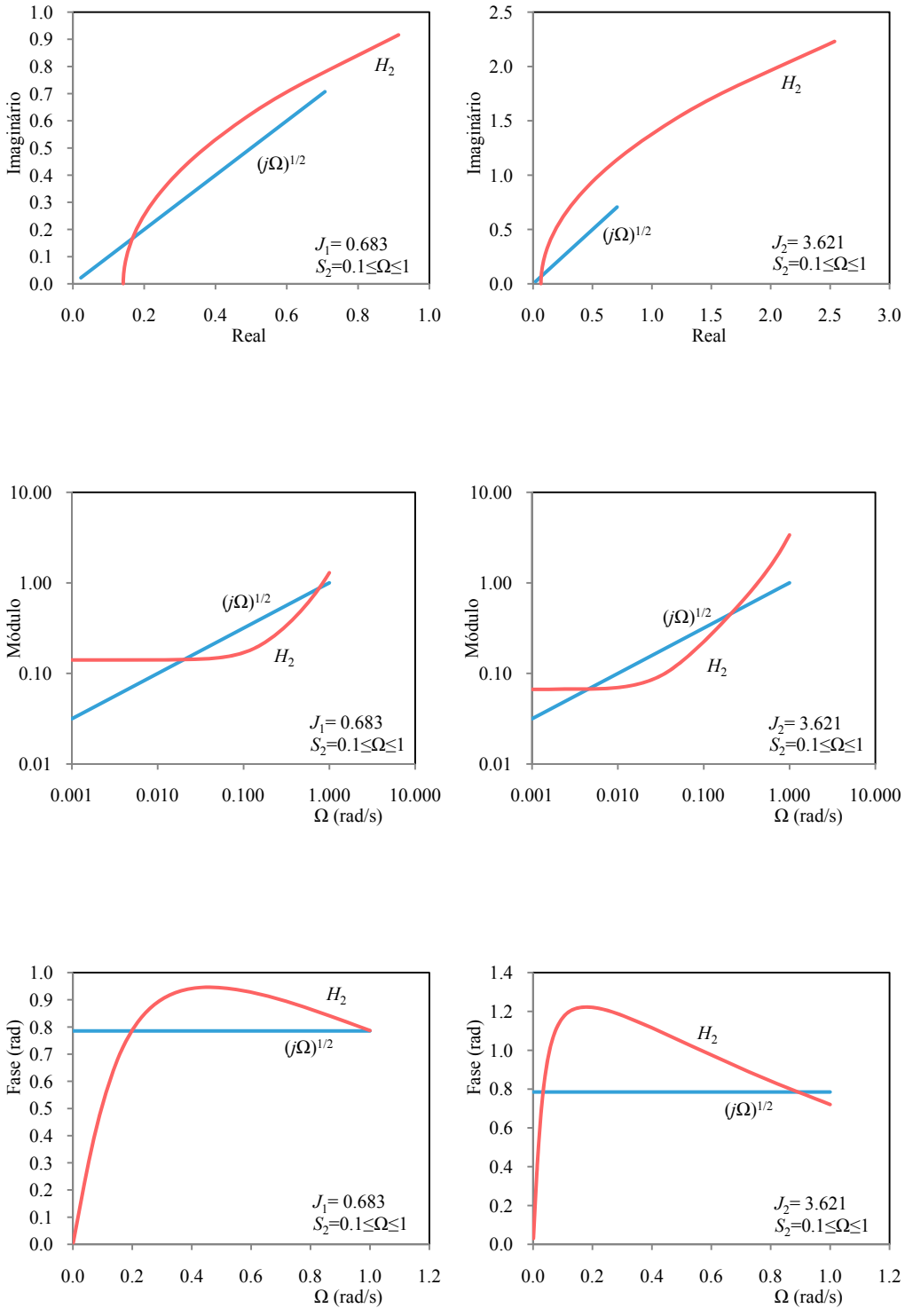


Figura 4.5 **Diagrama polar e diagramas de Bode, de amplitude e de fase, para aproximação por fracções de ordem $k = 2$, $\{J_1, J_2\} \times \{S_2\}$**

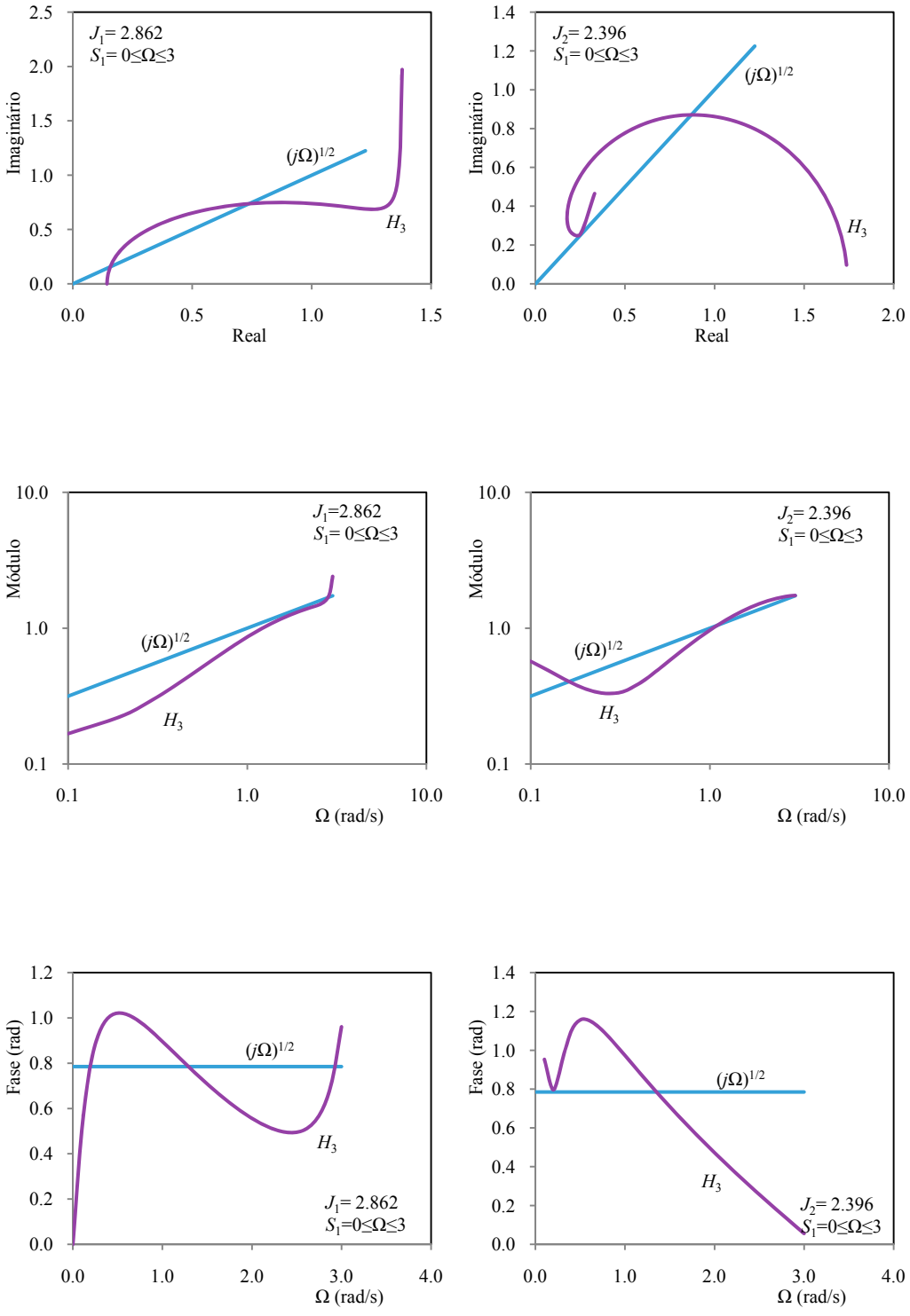


Figura 4.6 **Diagrama polar e diagramas de Bode, de amplitude e de fase, para aproximação por fracções de ordem $k = 3$, $\{J_1, J_2\} \times \{S_1\}$**

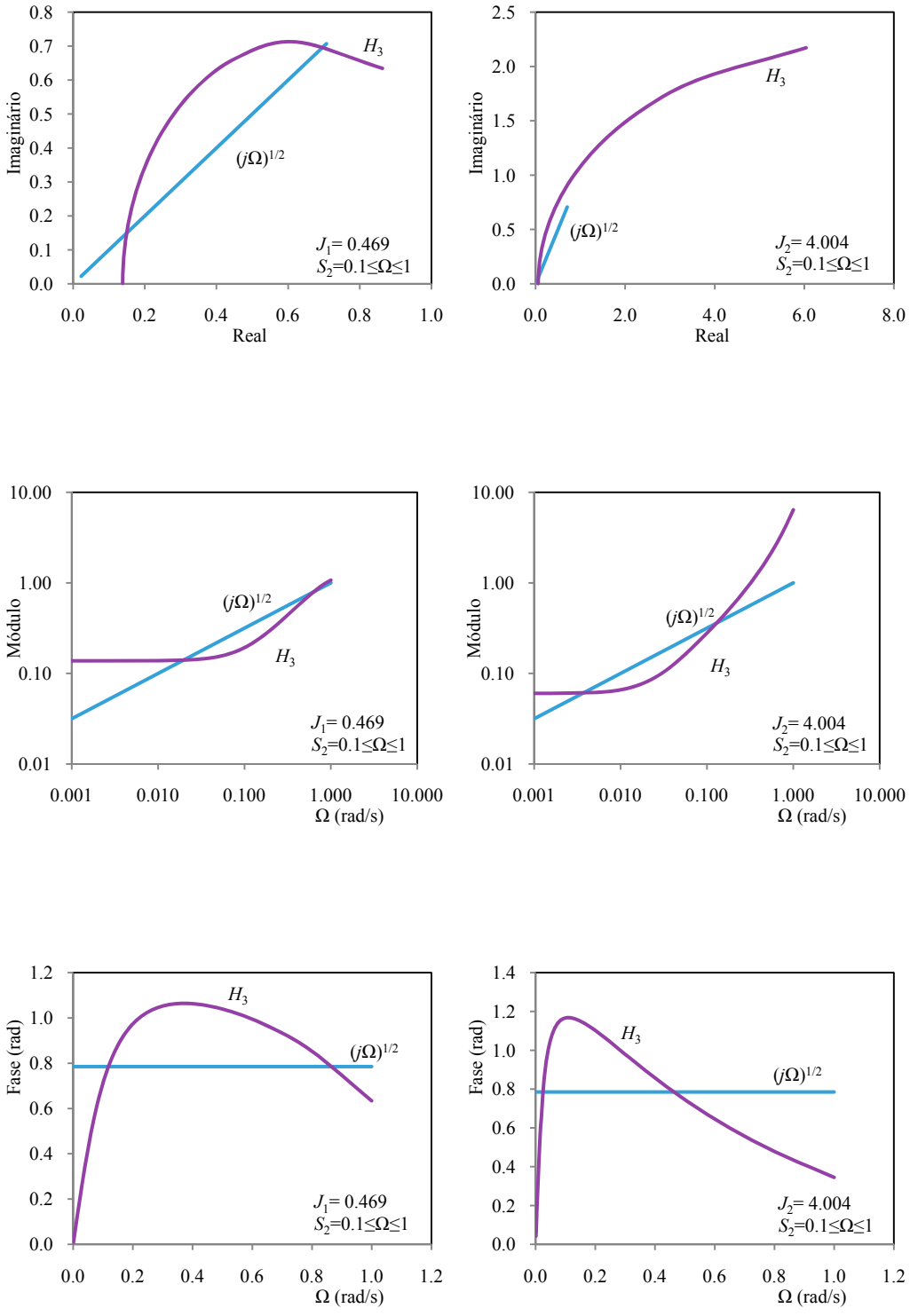


Figura 4.7 **Diagrama polar e diagramas de Bode, de amplitude e de fase, para aproximação por fracções de ordem $k = 3$, $\{J_1, J_2\} \times \{S_2\}$**

As tabelas 4.2 a 4.4 mostram os coeficientes, truncados a cinco casas decimais, correspondentes à aproximação por fracções de H_k, $k = \{1, 2, 3\}$.

Tabela 4.2 Coeficientes de $D^{1/2}$, para aproximação por fracções de primeira ordem $\{J_1, J_2\} \times \{S_1, S_2\}$

Função de avaliação (J)	Amostragem (S)	a_0	a_1	b_0	b_1
J_1	S_1	0.85139	-0.45058	1.0	0.35107
J_2	S_1	1.09736	-0.72000	1.0	0.15966
J_1	S_2	0.98007	-0.91888	1.0	-0.43176
J_2	S_2	3.69753	-3.62458	1.0	0.19300

Tabela 4.3 Coeficientes de $D^{1/2}$, para aproximação por fracções de segunda ordem $\{J_1, J_2\} \times \{S_1, S_2\}$

Função de avaliação (J)	Amostragem (S)	a_0	a_1	a_2	b_0	b_1	b_2
J_1	S_1	0.97630	-0.15382	-0.76947	1.0	0.72884	-0.27108
J_2	S_1	1.15184	-0.07881	-0.72812	1.0	0.74670	-0.25329
J_1	S_2	-0.76760	5.45407	-4.12185	1.0	1.99933	0.99935
J_2	S_2	-4.00099	16.39559	-12.13058	1.0	1.97611	0.97615

Tabela 4.4 Coeficientes de $D^{1/2}$, para aproximação por fracções de terceira ordem $\{J_1, J_2\} \times \{S_1, S_2\}$

Função de avaliação (J)	Amostragem (S)	a_0	a_1	a_2	a_3	b_0	b_1	b_2	b_3
J_1	S_1	0.97065	-0.53467	-0.67628	0.36639	1.0	0.36485	-0.55871	0.07615
J_2	S_1	0.97405	-2.67585	2.45254	-0.76178	1.0	-1.87254	0.96984	-0.09720
J_1	S_2	2.78499	-3.35851	6.27490	-4.97890	1.0	2.30723	1.61623	0.30832
J_2	S_2	-9.85890	12.31676	24.82486	-26.83178	1.0	2.76120	2.74724	0.98548

Note-se que novas funções de avaliação, diferentes conjuntos de pontos de amostragem, e outros valores para os limites do intervalo de aproximação, podem conduzir a coeficientes

de fracção diferentes. Por outras palavras, o AG produz uma solução específica para cada formulação de optimização e para cada conjunto de restrições.

O AG tem uma natureza estocástica e, como de costume, quando se aplicam algoritmos evolutivos os resultados de uma dada execução do AG podem corresponder a um mínimo local, em vez de um mínimo global. Portanto, é necessária uma verificação cuidadosa da última fracção antes de aceitar os resultados. No entanto, vários tipos de experiências demonstraram que, com o actual conjunto de parâmetros numéricos do AG, as soluções geradas têm uma pequena variação. De facto, o AG foi executado várias vezes e verificou-se que as soluções obtidas diferiam muito pouco entre si. Além disso, como é frequente na CE, experimentaram-se várias combinações dos valores numéricos dos parâmetros do AG e constatou-se que o conjunto adoptado era aquele que conduzia a melhores resultados.

Tal como se previa, pois foi imposta uma restrição de estabilidade, as fracções correspondentes aos coeficientes gerados pelo AG são estáveis. As figuras 4.8 a 4.10 mostram a localização dos pólos das funções H_k, $k = \{1, 2, 3\}$ no plano z em relação ao círculo unitário.

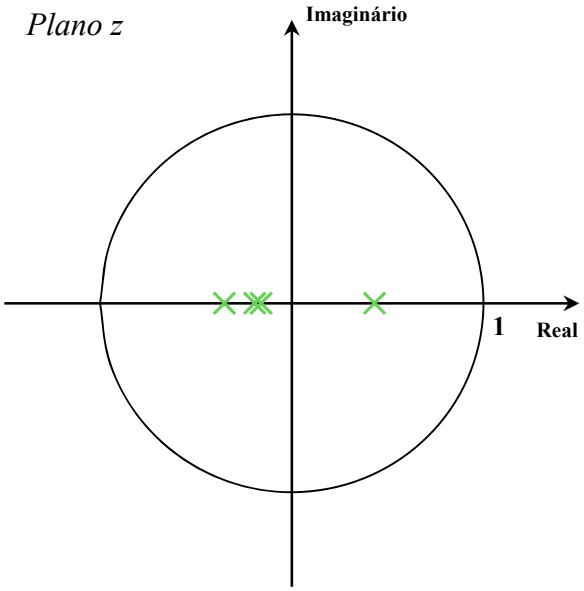


Figura 4.8 Localização dos pólos da função de ordem $k = 1$ no plano z

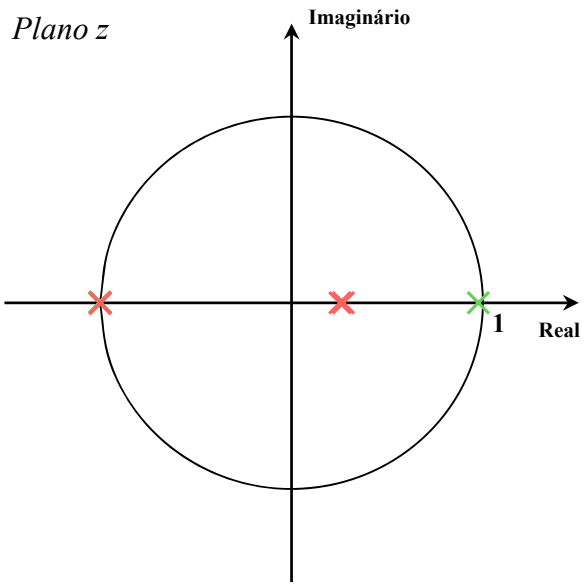


Figura 4.9 Localização dos pólos da função de ordem *k* = 2 no plano *z*

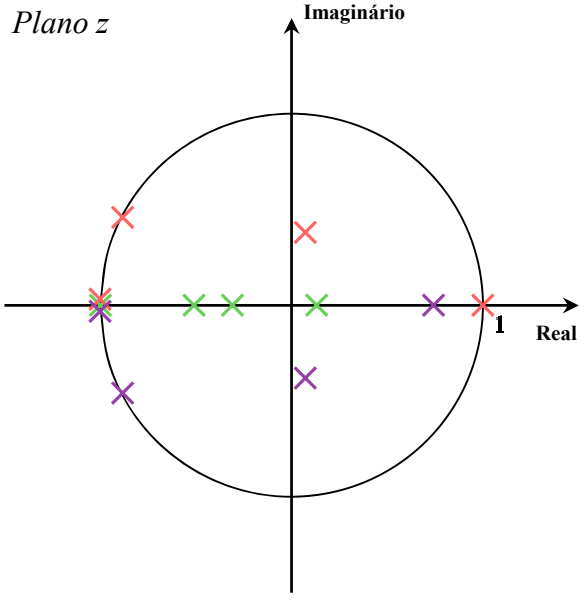


Figura 4.10 Localização dos pólos da função de ordem *k* = 3 no plano *z*

Como se pode confirmar pelas figuras 4.8, 4.9 e 4.10, todos os pólos das funções encontram-se dentro do círculo unitário, porém, em vários casos, verificou-se que as raízes

do denominador estavam próximas da condição limite $|z| = 1$. No entanto, se forem necessárias condições mais exigentes de estabilidade é fácil adaptar as condições (4.7) e (4.8) de modo a obter uma condição do tipo $|z| = c$, com $0 \leq c < 1$.

Analisando os resultados, concluímos que:

- Em geral, quanto maior a ordem da fracção, melhor será a aproximação.
- A função de avaliação J_1 produz resultados superiores para o diagrama polar, enquanto J_2 produz melhores resultados para o diagrama de Bode.
- O conjunto de pontos de amostragem S_1 parece estar bem adaptado ao diagrama polar, enquanto que S_2 produz melhores resultados para o diagrama de Bode.

Obviamente, a optimização pelo AG requer um maior tempo computacional, aumentando o número de pontos de amostragem, aumentando o número de elementos da população e, se o número de coeficientes for superior, isto é, aumentando o número de ordem da aproximação por fracção. No entanto, uma vez que o cálculo da fracção é realizado *off-line*, não há qualquer problema quanto à carga computacional e testes numéricos demonstraram que o algoritmo requer recursos computacionais aceitáveis para um computador comum.

Comparando as expansões de Padé e as fracções geradas por AG, constatamos claramente que a resposta em frequência é superior quando a optimização é efectuada por um programa evolutivo.

Deve notar-se que a abordagem de optimização evolutiva não está restrita ao cálculo das derivadas e integrais de ordem fraccionária. Na verdade, o sistema pode ser facilmente utilizado na aproximação por fracções de qualquer expressão fraccionária, sendo um caso típico o controlador fraccionário $PI^{\lambda}D^{\mu}$ com função de transferência $G(s) = K_p + K_i s^{-\lambda} + K_d s^{\mu}, \lambda > 0, \mu \leq 1$, onde s representa a variável de Laplace. Assim, um campo de generalização imediata, que desde logo se apresenta, consiste na aplicação a outras expressões de ordem fraccionária. No entanto, tal estudo não se encontra no âmbito do presente trabalho constituindo uma meta para investigação futura.

5. CONCLUSÕES

5.1. INTRODUÇÃO

Neste capítulo são destacados os principais assuntos abordados neste trabalho e são realçados alguns aspectos que poderão merecer uma investigação mais profunda. Nesta ordem de ideias, na secção 5.2 são apresentadas as conclusões principais e na secção 5.3 são referidas algumas perspectivas para desenvolvimento futuro.

5.2. PRINCIPAIS CONCLUSÕES DECORRENTES DO TRABALHO DESENVOLVIDO

Nesta dissertação foram apresentados os aspectos principais sobre os algoritmos evolutivos e a inteligência dos enxames.

Foi realizada uma análise detalhada do funcionamento dos AGs. Foram apresentados os seus aspectos principais de concepção e implementação, incluindo os parâmetros, operadores genéticos e estratégias para melhoria do desempenho destes algoritmos.

Após ter estabelecido as metodologias fundamentais, de seguida foi proposto um novo método, baseado em conceitos evolucionários, para o cálculo de expressões fraccionárias. Nesta linha de pensamento, foram introduzidas duas funções de avaliação e dois conjuntos alternativos de amostragem da frequência para a optimização através de AGs.

Os resultados obtidos demonstram que o cálculo de DIFs através de AGs é fácil de implementar, conduz a desempenhos excelentes e revela uma adaptabilidade a diferentes tipos de expressões. Nesta ordem de ideias conclui-se que os AGs têm uma capacidade enorme em aplicações deste tipo pelo que o seu potencial merece uma investigação mais alargada.

5.3. PERSPECTIVAS PARA DESENVOLVIMENTOS FUTUROS

Com base no trabalho desenvolvido surgiram os seguintes aspectos que podem ser alvo de uma possível investigação no futuro, nomeadamente:

- Estudo de novas funções de avaliação.
- Aplicação a outras expressões de ordem fraccionária.
- Utilização no CF de outros algoritmos propostos no âmbito da computação evolutiva.

Referências Documentais

Al-Alaoui, M.A. "Novel digital integrator and differentiator." *Electronics Letters* 29(4) (1993): 376-378.

Anastasio, T. J. "The Fractional-Order Dynamics of Brainstem Vestibulo-oculomotor Neurons." *Biological Cybernetics* 72(1) (1994): 69-79.

Bagley, R. L., e P. J. Torvik. "Fractional calculus - a different approach to the analysis of viscoelastically damped structures." *AIAA Journal* 21 (1983): 741-748.

Baleanu, D. "About fractional quantization and fractional variational principles." *Communications in Nonlinear Science and Numerical Simulation* 14(6) (2009): 2520-2523.

Barbosa, R., J. Machado, e M. Silva. "Time Domain Design of FractionalDifferintegrators Using Least Squares Approximations." *Signal Processing* (Elsevier Science) 86(10) (2006): 2567-2581.

Beasley, D., D. R. Bull, e R. R. Martin. "An overview of genetic algorithms: Part 1, Fundamentals." *University Computing* 15 (2) (1993): 58-69.

Booker, L.B., D.E. Goldberg, e J.H. Holland. "Classifier Systems and Genetic Algorithms." *Artificial Intelligence* 40 (1989): 235-282.

Bras, A. *Algoritmos Geneticos, Aplicacoes a Economia [Genetic Algorithms: Applications to the Portuguese stock market], M.Sc. dissertation.* Lisboa: ISEG/Technical University of Lisbon, 2002.

Cagnoni, S., A. B. Dobrzeniecki, R. Poli, e J. C. Yanch. "Genetic algorithm-based interactive segmentation of 3D medical images." *Image and Vision Computing* 17(12) (October 1999): 881–895.

Cantú-Paz, E. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, 2001.

Chen, Y.Q., e B.M. Vinagre. "A new IIR-type digital fractional order differentiator." *Signal Processing* 83(11) (2003): 2359–2365.

Chen, Y.Q., e K.L. Moore. "Discretization schemes for fractional-order differentiators and integrators." *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications* 49(3) (2002): 363-367.

Davidor, Y. "A Genetic Algorithm Applied to Robot Trajectory Generation." In *Handbook of Genetic Algorithms*, de L. Davis, 124-143. New York: Van Nostrand Reinhold, 1991.

Davis, L. *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.

De Jong, K. A. *An analysis of the behavior of a class of genetic adaptive systems. PhD thesis.* Ann Arbor, MI: University of Michigan, 1975.

Dorigo, M. *Optimization, Learning and Natural Algorithms, PhD thesis.* Italy: Politecnico di Milano, 1992.

Fogel, L. J. "Autonomous Automata." *Industrial Research* 4 (1962): 14-19.

Ganeshan, K., e J. Pickard. "Paper Composing Music Using Genetic Algorithms." In *Proceedings of the 17th NACCQ*, de S. Mann e T. Clear. 2004.

Goldberg, D. E. *Computer-aided gas pipeline operation using genetic algorithms and rule learning, PhD thesis.* Ann Arbor, MI: University of Michigan, 1983.

Goldberg, D. E., e R. Lingle. "Alleles, loci, and the traveling salesman problem." In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, montagem por Lawrence Erlbaum Associates, 154-159. 1985.

Goldberg, D. E., K. Deb, e B. Korb. "Don't Worry, Be Messy." In *Proceedings of the 4th International Conference on Genetic Algorithms*, de Richard K. Belew and Lashon B. Booker, 24-30. San Diego, CA: Morgan Kaufmann, 1991.

Goldberg, David E. *Genetic algorithms for search, optimization, and machine learning.* Addison-Wesley, 1989.

Handley, S. "Automated Learning of a Detector for α-helices in Protein Sequences via Genetic Programming." In *Proceedings of the Fifth International Conference on Genetic Algorithms*, de S. Forrest, 271-278. Urbana, Champaigne: Morgan Kaufmann Publishers, 1993.

Hedrih, K. "Vibration Modes of an Axially Moving Double Belt System with Creep Layer." *Journal of Vibration and Control* 14(9/10) (2008): 1333-1347.

Holland, J. H. *Adaption in Natural and Artificial Systems.* Ann Arbor, MI: The University of Michigan Press, 1975.

Holland, John H. "Outline for a logical theory of adaptive systems." *J. Assoc. Comput. Mach.* 3 (1962): 297-314.

Jacob, B. L. "Composing With Genetic Algorithms." In *Proceedings of the International Computer Music Conference*. Banff Alberta, 1995.

Kennedy, J., e R. Eberhart. "Particle swarm optimization." In *Proc. of the IEEE Int. Conf. on Neural Networks*, 1942-1948. Piscataway, NJ, 1995.

Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, 1992.

Lin, S., e B. Kernighan. "An effective heuristic algorithm for the traveling salesman problem." *Operations Research* 21 (1973): 498-516.

Lohn, J. D., S. P. Colombano, G. L. Haith, e D. Stassinopoulos. "A parallel genetic algorithm for automated electronic circuit design." In *Proceedings of the Computational Aerosciences Workshop*. NASA Ames Research Center, 2000.

Machado, J. T. "Analysis and design of fractional-order digital control systems." *Journal Systems Analysis, Modelling, Simulation* 27 (1997): 107-122.

Machado, J. T. "Discrete-time fractional-order controllers." *Journal of Fractional Calculus & Applied Analysis* 4 (2001): 47-66.

Mainardi, F. "Fractional relaxation-oscillation and fractional diffusion-wave phenomena." *Chaos, Solitons & Fractals* 7 (1996): 1461-1477.

McDonnell, J. R., B. L. Andersen, W. C. Page, e F. G. Pin. "Mobile Manipulator Configuration Optimization Using Evolutionary Programming." In *Proceedings of the First Annual Conference on Evolutionary Programming*, de D. B. Fogel e W. Atmar, 52-62. 1992.

Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd edition. Berlin, Germany: Springer-Verlag, 1996.

Miller, K. S., e B. Ross. *An introduction to the fractional calculus and fractional differential equations.* New York: John Wiley and Sons, 1993.

Mitchell, M. *An introduction to genetic algorithms.* Cambridge: MIT Press, 1997.

Mühlenbein, H., M. Schomish, e J. Born. "The Parallel Genetic Algorithm as Function Optimizer." *Parallel Computing* 17 (1991): 619-632.

Nigmatullin, R. "The statistics of the fractional moments: Is there any chance to "read quantitatively" any randomness?" *Signal Processing* 86(10) (2006): 2529-2547.

Oldham, K. B., e J. Spanier. *The fractional calculus: theory and application of differentiation and integration to arbitrary order.* New York: Academic Press, 1974.

Oliver, I. M., D. J. Smith, e J. R. C. Holland. "A study of permutation crossover operators on the traveling salesman problem." In *Genetic Algorithms and Their Applications: Proceedings of the Second Internacional Conference*, de John J. Grefenstette, 224-230. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1987.

Oliver, J. R. "Discovering Individual Decision Rules: an Application of Genetic Algorithms." In *Proceedings of the Fifth International Conference on Genetic Algorithms*, de Stephanie Forrest, 216-222. Urbana-Champaign: Morgan Kaufmann Inc, 1993.

Oustaloup, A. *La commande CRONE: commande robuste d'ordre non entier.* Hermes, 1991.

Page, W. C., J. R. McDonnell, e B. L. Andersen. "An Evolutionary Programming Approach to Multi-dimensional Path Planning." In *Proceedings of the First Annual Conference on Evolutionary Programming*, de D. B. Fogel e W. Atmar, 63-70. 1992.

Passino, K. M. "Biomimicry of bacterial foraging for distributed optimization and control." *IEEE Control Systems Magazine* 22 (Jun 2002): 52-67.

Podlubny, I. *Fractional differential equations.* San Diego: Academic Press, 1999.

Podlubny, I. "Fractional-order systems and PIλDμ-controllers." *IEEE Transactions on Automatic Control* 44(1) (1999): 208-213.

Punch, W. F., E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, e R. Enbody. "Further Research on Feature Selection and Classification using Genetic Algorithms." In *Proceedings of the Fifth International Conference on Genetic Algorithms*, de S. Forrest, 557-564. Urbana, Champaigne: Morgan Kaufmann Publishers, 1993.

Rechenberg, I. "Cybernetic Solution Path of an Experimental Problem." Royal Aircraft Establishment, UK, 1965.

Romero, J., e P. Machado. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music.* Springer Berlin Heidelberg, 2007.

Sabatier, J., O. P. Agrawal, e J. A. Tenreiro Machado. *Advances in Fractional Calculus. Theoretical Developments and Applications in Physics and Engineering.* Springer, 2007.

Samko, S. G., A. A. Kilbas, e O. I Marichev. *Fractional integrals and derivatives: theory and applications.* Gordon and Breach Science Publishers, 1993.

Schaffer, J. D., e L. J. Eshelman. "Designing Multiplierless Digital Filters Using Genetic Algorithms." In *Proceedings of the Fifth International Conference on Genetic Algorithms*, de S. Forrest, 439-444. Urbana, Champaigne: Morgan Kaufmann Publishers, 1993.

Smigrodzki, R., B. Goertzel, C. Pennachin, L. Coelho, F. Prosdocimi, e W. D. Parker Jr. “Genetic algorithm for analysis of mutations in parkinson’s disease.” *Artificial Intelligence in Medicine* 35(3) (November 2005): 227–241.

Syswerda, G. “Schedule Optimization using Genetic Algorithms.” In *Handbook of Genetic Algorithms*, de L. Davis, 332-349. New York: Van Nostrand Reinhold, 1991.

Tarasov, V. E., e G. M. Zaslavsky. “Fractional dynamics of systems with long-range interaction.” *Communications in Nonlinear Science and Numerical Simulation* 11(8) (2006): 885-898.

Tseng, C.C. “Design of fractional order digital fir differentiators.” *IEEE Signal Processing Letters* 8(3) (2001): 77–79.

Vinagre, B.M., Y.Q. Chen, e I. Petras. “Two direct Tustin discretization methods for fractional-order differentiator/integrator.” *Journal of the Franklin Institute* 340(5) (2003): 349–362.

Vinterbo, S. A., e L. Ohno-Machado. “A genetic algorithm approach to multi-disorder diagnosis.” *Artificial Intelligence in Medicine* 18(2) (February 2000): 117–132.

Wright, A. H. “Genetic algorithms for real parameter optimization.” In *Foundations of genetic algorithms*, de G. J. Rawlins, 205-218. San Mateo, CA: Morgan Kaufmann, 1991.

Yuret, D., e M. Maza. “A genetic algorithm system for predicting the OEX.” *Analysis of Stocks & Commodities*, June de 1994: 255–259.

Apêndice A. Cálculo Fraccionário

A.1 INTRODUÇÃO

O Cálculo Fraccionário (CF) é a área da matemática que estuda os operadores de integração e diferenciação de ordem não inteira. O interesse nesta área surgiu quase ao mesmo tempo que as ideias sobre o cálculo integral e diferencial clássicas. O primeiro documento sobre CF data de 1695 quando L'Hôpital escreveu uma carta a Leibniz onde o questiona sobre o significado de $D^n y$ quando $n = 1/2$. No entanto, deveu-se a Euler (1738) o primeiro passo, quando este analisou o cálculo de Derivadas Fraccionárias (DFs) para a função potência. No seguimento deste resultado Laplace (1812), Lacroix (1820), Fourier (1822), Abel (1823), Liouville (1832), Riemann (1847) e Letnikov (1868) sugeriram, também, algumas ideias relativas ao cálculo de DFs.

Devem-se destacar os trabalhos de Abel e de Liouville, visto terem sido estes a dar o verdadeiro início da teoria relativa ao cálculo de derivadas e integrais fraccionários (DIFs). Abel investigou certas expressões fora do contexto do cálculo de DIFs, mas os resultados foram de importância considerável para o desenvolvimento da teoria. Por seu lado, Liouville estudou, explicitamente, várias questões nomeadamente a definição e o cálculo de DFs para valores complexos e a sua aplicação a certos tipos de equações diferenciais lineares ordinárias, o efeito de uma mudança de variável no cálculo de DIFs e a definição de uma DF como o limite do quociente ${D^\alpha}_h f/h^\alpha$, onde ${D^\alpha}_h f$ é uma diferença de ordem fraccionária. Posteriormente, Riemann, Holmgren (1865-1867) e Letnikov tiveram, também, papéis de relevo no prosseguimento da teoria. Entre outros resultados, Holmgren considerou, pela primeira vez, a derivação e a integração fraccionárias como operações inversas e generalizou a expressão de $d^\alpha\ (u\ v)/dx^\alpha$. Por sua vez, Letnikov desenvolveu a DF como limite da expressão $\lim_{h\to 0} {D^\alpha}_h f/h^\alpha$, demonstrou que as expressões propostas por Liouville e Riemann estavam de acordo com esta definição e generalizou a teoria dos DIFs para valores complexos. Mais próximo dos nossos dias, são de referir numerosas contribuições tais como as de Hadamard (1892), Weyl (1917) e Marchaud (1927), que tem vindo a ampliar o âmbito desta teoria.

Na secção A.2 é feita uma introdução à função Gama de Euler. De seguida, as secções A.3 e A.4 apresentam algumas das definições e propriedades do CF. Por fim, as secções A.5 e A.6 abordam a utilização das transformadas de Laplace e Fourier no CF.

A.2 FUNÇÃO GAMA DE EULER

Uma das funções básicas do CF é a função Gama de Euler, $\Gamma(z)$, que generaliza o conceito do factorial de $n!$ para valores reais e complexos de n. Como se pode observar pela figura A.1 trata-se de uma função analítica em todos os pontos excepto para $z = 0, -1, -2, \ldots$.

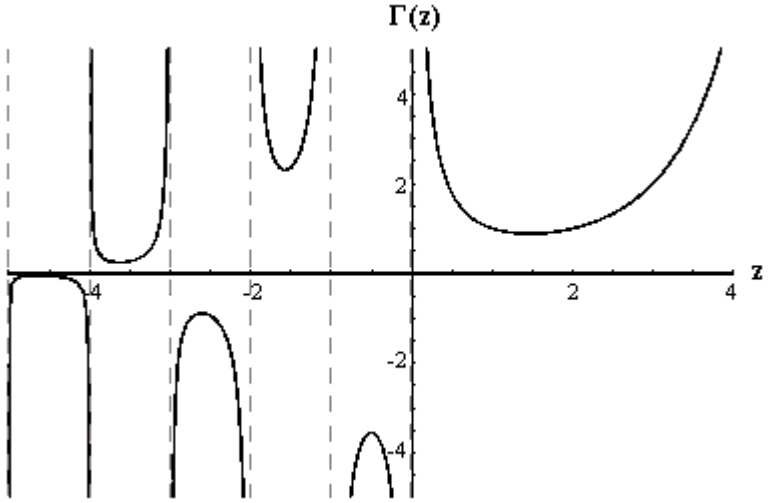


Figura A.1 Função Gama de Euler Γ(z)

A função Gama de Euler é definida através do integral definido:

$$\Gamma(z) = \int_0^\infty e^{-t}\, t^{z-1}\, dt, \;\; Re(z) > 0 \qquad \textbf{(A.1)}$$

Para valores reais do argumento a função Gama é dada por:

$$\Gamma(x) = (x-1)\,\Gamma(x-1) \qquad \textbf{(A.2a)}$$

ou

$$\Gamma(x+1) = x\,\Gamma(x) \qquad \textbf{(A.2b)}$$

Dado que Γ(1) = 1, usando a definição (A.2b) para valores inteiros de *x*, obtém-se:

$\Gamma(2) = 1 \cdot \Gamma(1) = 1 = 1!$
$\Gamma(3) = 2 \cdot \Gamma(2) = 2 \cdot 1! = 2!$
$\Gamma(4) = 3 \cdot \Gamma(3) = 3 \cdot 2! = 3!$
… … … …

Generalizando,

$$\Gamma(x) = (x-1)\Gamma(x-1) = (x-1)(x-2)! = (x-1)! \qquad \textbf{(A.3a)}$$

ou

$$\Gamma(x+1) = x\,\Gamma(x) = x\,(x-1)! = x! \qquad \textbf{(A.3b)}$$

para valores de *x* = 0, 1, 2, ….Podendo-se concluir que a função Gama para valores inteiros e positivos de *x* fica reduzida ao cálculo factorial.

A.3 DERIVADAS E INTEGRAIS FRACCIONÁRIOS

Muitas das expressões relativas às DIFs resultam do cálculo clássico das derivadas e integrais de ordem inteira. Assim, sempre que possível será apresentado em primeiro lugar o resultado de derivadas e integrais múltiplos ordinários e, de seguida, a correspondente definição e resultado para as derivadas e os integrais de ordem não inteira.

A.3.1 DEFINIÇÃO DE RIEMANN-LIOUVILLE

A definição mais frequente para a integração a uma ordem fraccionária é a extensão da conhecida fórmula de Cauchy para o *n*-ésimo integral:

$$\int_0^t dt_1 \int_0^{t_1} dt_2 \cdots \int_0^{t_{n-1}} f(t_n)dt_n = \frac{1}{(n-1)!}\int_0^t (t-\tau)^{n-1} f(\tau)d\tau, \;\; n \in \aleph \qquad \textbf{(A.4)}$$

substituindo o factorial $(n-1)!$ pela função Gama $\Gamma(n)$, obtém-se o chamado integral fraccionário de Riemann-Liouville de ordem $\alpha \geq 0$:

$$ {}_aD_t^{-\alpha} f(t) = \frac{1}{\Gamma(\alpha)}\int_a^t (t-\tau)^{\alpha-1} f(\tau)d\tau\,, t > a \qquad \textbf{(A.5)}$$

com ${}_aD_t^{-0} f(t) = f(t)$.

A escolha do limite inferior com $a = -\infty$ é devida originalmente a Liouville, enquanto que com $a = 0$ é devida a Riemann.

Para introduzir a noção de derivada fraccionária de ordem α, pode-se considerar a possibilidade de substituição de α por (– α) na equação (A.5). No entanto, deve-se ter algum cuidado, de forma a garantir a convergência do integral e a preservar as propriedades das derivadas de ordem inteira.

Considerando D^n, com $n \in \aleph$, o operador da derivada de ordem n, e I o operador identidade, sabe-se que:

$$D^n\, D^{-n} = I, D^{-n} D^n \neq I, n \in \aleph \tag{A.6}$$

Isto é, D^n é inversa à esquerda do correspondente operador integral D^{-n}. Deste modo, D^α deve ser definida como inversa à esquerda de $D^{-\alpha}$. Assim, introduzindo n, inteiro e positivo, tal que $n - 1 < \alpha < n$, obtém-se a definição Riemann-Liouville para a derivada fraccionária da função $f(t)$ de ordem $\alpha > 0$:

$$ {}_aD_t^\alpha f(t) = \frac{d^n}{dt^n}\Big({}_aD_t^{-(n-\alpha)} f(t)\Big)$$

$$= \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha+1-n}}\, d\tau, \qquad n-1 < \alpha < n \tag{A.7}$$

A definição de Riemann-Liouville (A.7) desempenha um papel importante no desenvolvimento da teoria das DIFs e na sua aplicação na matemática pura, nomeadamente na solução de equações diferenciais de ordem inteira, definições de novas classes de funções e somatório de séries.

No entanto, quando se trata da sua aplicação a problemas reais a sua utilização é muito restrita, devido ao facto de a derivada de Riemann-Liouville levar à definição de condições iniciais sem uma interpretação física associada.

A.3.2 DEFINIÇÃO DE CAPUTO

A derivada fraccionária de Caputo de ordem α com respeito à variável *t* da função $f(t)$ e ao ponto de início $t = a$ é definida como:

$$ {}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(\alpha - n)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha+1-n}} d\tau, \qquad n-1 < \alpha < n, n \in \aleph \tag{A.8} $$

onde Γ é a função Gama e $f^{(n)}(t)$ denota a derivada de ordem *n*.

Se $\alpha < 0$, temos o integral fraccionário de ordem $-\alpha$:

$$ {}_aI_t^{-\alpha} f(t) = {}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(-\alpha)} \int_a^t \frac{f(\tau)}{(t-\tau)^{1+\alpha}} d\tau, \qquad \alpha < 0 \tag{A.9} $$

A aproximação desenvolvida por Caputo permite a formulação das convencionais condições iniciais com interpretação física, tais como (considerando o limite inferior $t = a$) $D_y\big|_{x=a}$, $D_y^2\big|_{x=a}$, etc. (*i.e.* posição, velocidade, etc.). Considerando que os modelos matemáticos de determinados sistemas físicos levam à formulação de equações diferenciais de ordem fraccionária, o que vem sendo cada vez mais frequente em determinados domínios de aplicação, esta característica torna a definição de Caputo particularmente útil, pois permite a introdução das condições iniciais.

A.3.3 DEFINIÇÃO DE GRÜNWALD-LETNIKOV

A definição apresentada por Grünwald-Letnikov aqui apresentada é considerada a mais importante, visto que é a que impõe menos restrições nas funções em que é aplicada e reúne num único operador as noções de derivada e integral. A aproximação de Grünwald-Letnikov aborda o problema sob o ponto de vista da derivação, começando pela definição básica para a primeira derivada de uma função $f(t)$:

$$ \frac{df}{dt} \equiv f'(t) = \lim_{h \to 0} \frac{f(t) - f(t-h)}{h} \tag{A.10} $$

Aplicando novamente esta equação, pode-se obter a segunda derivada:

$$ \frac{d^2 f}{dt^2} \equiv f''(t) = \lim_{h \to 0} \frac{f'(t) - f'(t-h)}{h} $$

$$ = \lim_{h \to 0} \frac{1}{\mathrm{h}} \left(\frac{f(t) - f(t-h)}{h} - \frac{f(t-h) - f(t-2h)}{h} \right) $$

$$= \lim_{h\to 0}\left(\frac{f(t)-2f(t-h)+f(t-2h)}{h^2}\right) \tag{A.11}$$

e, do mesmo modo:

$$\frac{d^3f}{dt^3} \equiv f'''(t) = \lim_{h\to 0}\left(\frac{f(t)-3f(t-h)+3f(t-2h)-f(t-3h)}{h^3}\right) \tag{A.12}$$

Por indução obtém-se a generalização de derivada de ordem inteira *n*:

$$D^n f(t) \equiv \frac{d^n f}{dt^n} \equiv f^{(n)}(t) = \lim_{h\to 0}\frac{1}{h^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}f(t-kh) \tag{A.13}$$

onde,

$$\binom{n}{k} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!} = \frac{n!}{k!\,(n-k!)} \tag{A.14}$$

é a notação utilizada para os coeficientes do binómio.

Partindo da definição de integral como sendo o limite da soma de Riemann, e seguindo um processo iterativo semelhante ao anterior, chega-se à expressão final para um integral de ordem *n*:

$${}_aD_t^{-n}f(t) = \lim_{h\to 0} h^n \sum_{k=0}^{\left[\frac{t-a}{h}\right]}(-1)^k\binom{-n}{k}f(t-kh) \tag{A.15}$$

onde $\left[\frac{t-a}{h}\right]$ indica a parte inteira de $\frac{t-a}{h}$ e:

$$(-1)^k\binom{-n}{k} = \frac{n(n+1)(n+2)\cdots(n+k-1)}{k!} \tag{A.16}$$

Comparando as expressões (A.13) e (A.15), verifica-se que a derivada e o integral de ordem inteira *n* da função $f(t)$ podem ser definidos através de uma única expressão:

$$ {}_aD_t^q f(t) = \lim_{h\to 0} h^{-q} \sum_{k=0}^{\left[\frac{t-a}{h}\right]} (-1)^k \binom{q}{k} f(t-kh) \tag{A.17} $$

Esta expressão (A.17) representa a derivada e integral de ordem *n* para $q = n$ e $q = -n$, respectivamente. Deste modo, pode-se generalizar as noções de integração e diferenciação a uma ordem não inteira fazendo $q = \alpha$ ($\alpha \in \Re$), obtendo-se assim a definição de Grünwald-Letnikov para a derivada e integral fraccionário de ordem α:

$$ {}_aD_t^\alpha f(t) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{k=0}^{\left[\frac{t-a}{h}\right]} (-1)^k \binom{\alpha}{k} f(t-kh), \;\; \alpha \in \Re \tag{A.18} $$

onde,

$$ \binom{\alpha}{k} = \frac{\alpha(\alpha-1)(\alpha-2)\cdots(\alpha-k+1)}{k!} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \tag{A.19} $$

A.4 PROPRIEDADES DAS DERIVADAS E INTEGRAIS FRACCIONÁRIOS

As definições de derivadas e integrais fraccionários apresentadas em A.3 possuem as seguintes propriedades, onde α e β são dois valores arbitrários (positivos, negativos ou zero):

$$ \begin{aligned} & D^0 f(t) = f(t) \\ & D^\alpha D^{-\alpha} f(t) = f(t) \\ & D^\alpha \left[D^\beta f(t)\right] = D^\beta [D^\alpha f(t)] = D^{\alpha+\beta} f(t) \\ & D^n [D^\alpha f(t)] = D^\alpha [D^n f(t)] = D^{n+\alpha} f(t), n \in \aleph \end{aligned} \tag{A.20} $$

As propriedades apresentadas nem sempre se verificam para todas as definições. Portanto, deve-se ter em atenção qual a definição utilizada para a derivada e integral fraccionário. Existem outras propriedades importantes para a análise do CF, das quais se destacam a linearidade e a regra de Leibniz.

A.4.1 LINEARIDADE

De forma semelhante à diferenciação de ordem inteira, a diferenciação fraccionária é uma operação linear:

$$D^{\alpha}(\lambda f(t) + \mu g(t)) = \lambda D^{\alpha} f(t) + \mu D^{\alpha} g(t) \tag{A.21}$$

onde D^{α} representa qualquer uma das definições de diferenciação fraccionária referidas em A.3.

A.4.2 REGRA DE LEIBNIZ

A regra de Leibniz para a derivada de ordem inteira *n* do produto de duas funções $f(t)$ e $g(t)$ é dada por:

$$D^{n}[f(t)\, g(t)] = \sum_{k=0}^{n} \binom{n}{k} D^{n-k} f(t)\, D^{k} g(t) \tag{A.22}$$

Generalizando a expressão (A.22) para um valor α de ordem não inteira, a regra de Leibniz para a diferenciação fraccionária toma a forma:

$$D^{\alpha}[f(t)\, g(t)] = \sum_{k=0}^{\infty} \binom{\alpha}{k} D^{\alpha-k} f(t)\, D^{k} g(t) \tag{A.23}$$

A regra de Leibniz é especialmente útil para a determinação de derivadas fraccionárias do produto de uma função que é um polinómio com uma outra função em que a sua derivada fraccionária é conhecida.

A.5 TRANSFORMADA DE LAPLACE DE DERIVADAS FRACCIONÁRIAS

A Transformada de Laplace (TL) revela-se uma ferramenta extremamente útil na análise e projecto de sistemas de controlo, devido à transformação das operações integração e diferenciação no domínio dos tempos em operações algébricas no domínio das frequências (plano *s*). Nesta perspectiva, a TL é também importante na análise e projecto de sistemas fraccionários.

A TL de $f(t)$, denominada $L\{f(t)\} = F(s)$, é uma função de variável complexa $s = \sigma + j\omega$, onde:

$$F(s) = \int_0^{\infty} f(t)\, e^{-st} dt \tag{A.24}$$

A função original $f(t)$ pode ser obtida a partir de *F*(*s*) através da transformada de Laplace inversa L^{-1}:

$$f(t) = L^{-1}\{F(s)\} = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)\, e^{st} ds \tag{A.25}$$

onde *c* é um valor seleccionado para a direita de todas as singularidades de *F*(*s*) no plano *s*.

A TL da derivada ou integral de ordem inteira *n* da função $f(t)$ é dada por:

$$L\{D^n f(t)\} = s^n F(s) - \sum_{k=0}^{n-1} s^k\, D^{n-k-1} f(0), \qquad n = 0, \pm 1, \pm 2, \ldots \tag{A.26}$$

Normalmente, pode-se generalizar (A.26) a uma ordem α arbitrária do seguinte modo:

$$L\{D^{\alpha} f(t)\} = s^{\alpha} F(s) - \sum_{k=0}^{n-1} s^k\, D^{\alpha-k-1} f(0), \ \forall \alpha \tag{A.27}$$

Contudo, esta expressão sofre ligeiras alterações conforme a definição utilizada para a derivada fraccionária. Assim iremos ver a seguir a TL para as definições de Riemann-Liouville, Caputo e Grünwald-Letnikov.

A TL para a derivada fraccionária de Riemann-Liouville (RL) de ordem $\alpha > 0$ é:

$$L\left\{{}^{RL}D^{\alpha}f(t)\right\} = s^{\alpha}F(s) - \sum_{k=0}^{n-1} s^{k}\, D^{\alpha-k-1}f(t)|_{t=0}\,, n-1 < \alpha < n \qquad \textbf{(A.28)}$$

Dada a ausência de uma interpretação física dos valores limite das derivadas fraccionárias para $t = 0$, a sua aplicação prática é muito restrita.

A TL para a derivada fraccionária de Caputo (C) é dada por:

$$L\left\{{}^{C}D^{\alpha}f(t)\right\} = s^{\alpha}F(s) - \sum_{k=0}^{n-1} s^{\alpha-k-1}\, f^{(k)}(0), n-1 < \alpha < n \qquad \textbf{(A.29)}$$

Tal como se verifica na expressão (A.27), também aqui intervêm os valores da função $f(t)$ para $t = 0$. Isto permite a interpretação das convencionais condições iniciais, expressas em termos de derivadas de ordem inteira: $f(0) = f_0$ (posição inicial), $f'(0) = f_1$ (velocidade inicial), $f''(0) = f_2$ (aceleração inicial), etc. Esta característica torna esta definição particularmente útil para a resolução de problemas reais caracterizados por equações diferenciais fraccionárias lineares de coeficientes constantes.

A TL para a derivada fraccionária de Grünwald-Letnikov (GL) é dada por:

$$L\left\{{}^{GL}D^{\alpha}f(t)\right\} = s^{\alpha}F(s), \qquad 0 < \alpha < 1 \qquad \textbf{(A.30)}$$

A TL para o integral fraccionário das definições referidas acima pode ser obtida omitindo, nas expressões anteriores, a segunda parte (somatório) do lado direito das equações de derivadas fraccionárias.

De notar que o operador *L* permite a passagem do domínio dos tempos para o domínio das frequências (plano *s*), ou seja pode-se realizar no domínio das frequências toda a análise e projecto de sistemas, e de seguida voltar ao domínio dos tempos (operador L^{-1}) para verificar o seu comportamento.

A.6 TRANSFORMADA DE FOURIER DE DERIVADAS FRACCIONÁRIAS

Uma ferramenta também muito útil para a análise de sistemas no domínio das frequências é a Transformada de Fourier (TF), pelo que igualmente se torna importante nos sistemas fraccionários.

A TF da função contínua $f(t)$, denominada $F\{f(t)\} = F(\omega)$, é definida como sendo a função em frequência:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt, \qquad -\infty < \omega < \infty \tag{A.31}$$

A função $f(t)$ pode ser determinada a partir de $F(\omega)$ aplicando a transformada de Fourier inversa F^{-1}:

$$f(t) = F^{-1}\{F(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{j\omega t}\, d\omega \tag{A.32}$$

A TF da derivada ou integral de ordem inteira *n* da função $f(t)$ é dada por:

$$F\{D^n f(t)\} = (j\omega)^n\, F(\omega), \qquad n = 0, \pm 1, \pm 2, \ldots \tag{A.33}$$

Generalizando a expressão (A.33) para valores de α de ordem arbitrária, obtemos a definição geral:

$$F\{D^\alpha f(t)\} = (j\omega)^\alpha\, F(\omega), \qquad \forall \alpha \tag{A.34}$$

Assim, a TF do integral e da derivada fraccionária será dada por:

$$F\{D^{-\alpha}f(t)\} = (j\omega)^{-\alpha}\,F(\omega)$$
$$F\{D^{\alpha}f(t)\} = (j\omega)^{\alpha}\,F(\omega) \quad \textbf{(A.35)}$$

A expressão (A.35) é válida para qualquer uma das definições referidas para o caso em que estas coincidem, isto é, ${}_a^{RL}D(ft) = {}_a^{C}D(ft) = {}_a^{GL}D(ft)$ para $a = -\infty$.