FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores

Algoritmos Genéticos: Aplicação à Robótica

Eduardo José Solteiro Pires

Licenciado em Engenharia Electrotécnica pela Faculdade de Ciências e Tecnologia
da
Universidade de Coimbra

Dissertação realizada para satisfação parcial
dos requisitos de grau de mestre
em
Engenharia Electrotécnica e de Computadores
(Área de especialização de Sistemas)

Dissertação realizada sob a supervisão de
Professor Doutor José António Tenreiro Machado,
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto, Outubro de 1998

Resumo

Esta dissertação fornece uma visão global dos algoritmos evolutivos, nomeadamente sobre algoritmos genéticos, estratégias de evolução, programação evolutiva e programação genética. O trabalho inclui um conjunto de aplicações de sistemas robóticos baseados em algoritmos genéticos. Por fim, a dissertação apresenta os resultados de investigação na área do planeamento de trajectórias para manipuladores robóticos planares, de dois, três e quatro graus de liberdade. Assim, o estudo descreve e analisa técnicas, baseadas em algoritmos genéticos, no planeamento de trajectórias, utilizando a cinemática directa no espaço operacional.

O algoritmo, além de planear as trajectórias, optimiza ainda os deslocamentos das juntas do robô e minimiza a ondulação residual da cinemática diferencial das juntas. São também realizados estudos na sensibilidade de alguns parâmetros do algoritmo genético proposto.

Abstract

This thesis studies the theory of evolutionary computation and, in particular, the genetic algorithms (GAs). In a second phase it is analyzed the application of GAs in the area of robotic systems. Based on these concepts it is investigated the trajectory planning, through GAs, for planar manipulators with several degrees of freedom. In this line of thought several aspects are investigated, namely the influence of AG parameters. Finally, several improvements are introduced in the algorithm in order to reduce the ripple in the time evolution of the robot positions and velocities.

Resumé

Cette dissertation fournit une vision globale des algorithmes évolutifs, en particulier, sur des algorithmes génétiques, des stratégies d'évolution, de la programmation évolutive et de la programmation génétique. Ce travail comprend un ensemble d'applications de systèmes robotiques fondé sur des algorithmes génétiques. Finalement, cette disseration présente les résultats de la recherche dans le domaine du planning des trajectories pour des manipulateurs robotiques plans de deux, troits et quatre dégrés de liberté. Cet étude décrit et analyse, donc, des técniques, fondées sur des algorithmes génétiques dans le planning des trajectoires, en utilisant le modèle géométrique directe dans l'espace opérationnel.

Outre projeter les trajectories, l'algorithme optimise aussi les déplacements des liaisons du robot et minimise le *ripple* du modèle cinématique des liaisons. On a aussi réalisé des études sur la sensibilisation de certains paramètres de l'algorithme génétique proposé.

Palavras chave:

Algoritmos Genéticos
Algoritmos Evolutivos
Computação Evolutiva
Estratégias de Evolução
Optimização
Planeamento de Trajectórias
Programação Evolutiva
Programação Genética
Robótica

Keywords:

Evolution Strategies
Evolutionary Algorithms
Evolutionary Computation
Evolutionary Programming
Genetic Algorithms
Genetic Programming
Optimization
Robotics
Trajectory Planning

Mots clés:

Algorithmes Génétiques
Algorithmes Évolutifs
Ordination Évolutive
Optimisation
Planning de Trajectoires
Programmation Évolutive
Programmation Génétique
Robotique
Strategies d'Évolution

Agradecimentos:

Agradeço ao meu orientador Prof. J. A. Tenreiro Machado por todo o seu apoio, colaboração e disponibilidade manifestados ao longo deste trabalho.

Queria ainda agradecer a todas a pessoas que contribuíram directa ou indirectamente para a realização deste trabalho. Particularmente ao Paulo Salgado e ao Bonifácio Pires.

Por fim, queria agradecer à Secção de Engenharia Electrotécnica da UTAD pelas condições de trabalho proporcionadas.

Porto, Outubro de 1998

Eduardo Pires

Índice

1 INTRODUÇÃO 1

1.1 Introdução 1

1.2 Objectivos 1

1.3 Estrutura da dissertação 2

1.4 Referências 3

2 ALGORITMOS EVOLUTIVOS 5

2.1 Introdução 5

2.2 Introdução à teoria da selecção natural 5

2.3 Aplicação da teoria de selecção natural à ciência da computação 6

2.4 Estrutura de um algoritmo evolutivo 7

2.5 Analogia entre os termos utilizados nos algoritmos evolutivos e na natureza 8

2.6 Projecto de um algoritmo evolutivo 9
2.6.1 Introdução 9
2.6.2 Escolha do tipo e variações do algoritmo evolutivo 9
2.6.3 Espaço de pesquisa 10

2.7 Algoritmos evolutivos paralelos 11
2.7.1 Introdução 11
2.7.2 Classificação dos algoritmos evolutivos paralelos 11

2.8 Convergência do algoritmo 14

2.9 Classificação dos algoritmos evolutivos 15

2.10 Variantes dos algoritmos evolutivos 15

2.11 Algoritmos genéticos 15
2.11.1 Introdução 15
2.11.2 Comparação entre os algoritmos genéticos e os métodos clássicos 16
2.11.3 Projecto de um algoritmo genético 18
2.11.4 Representação e codificação dos algoritmos genéticos 19
2.11.5 Função de aptidão 21
2.11.6 Mecanismo de selecção 25
2.11.6.1 Introdução 25
2.11.6.2 Desempenho do mecanismo de selecção 25
2.11.6.3 Classificação dos operadores de selecção 26
2.11.6.4 Operadores de selecção 27
2.11.6.4.1 Introdução 27
2.11.6.4.2 Selecção por nível (posto) 27

2.11.6.4.3 Selecção por torneio (*tournament selection*) 28
2.11.6.4.4 Selecção de Bolzmann 29
2.11.6.4.5 Variações do modelo de selecção 29
2.11.7 Operadores genéticos 30
2.11.7.1 Introdução 30
2.11.7.2 Operador de cruzamento 31
2.11.7.2.1 Cruzamento uniforme 32
2.11.7.2.2 Recombinação com vários pais 32
2.11.7.2.3 Cruzamento análogo 32
2.11.7.2.4 Cruzamento de segregação 33
2.11.7.3 Operador de mutação 33
2.11.7.4 Operador de reordenação/inversão 34
2.11.7.5 Outros operadores 39
2.11.7.5.1 Introdução 39
2.11.7.5.2 Operador translocação 39
2.11.7.5.3 Operadores de duplicação e de remoção 39
2.11.7.5.4 Operadores sexuais 39
2.11.7.5.5 Restrição no casamento 40
2.11.7.5.6 Mecanismo de reinserção 41
2.11.7.5.7 Operador de dominância 41
2.11.8 Condição de finalização do algoritmo 43
2.11.9 Epistase 43
2.11.10 Problema *ilusório* 44
2.11.11 Desvio genético 45
2.11.12 Esquemas híbridos 45
2.11.13 Porque funcionam os algoritmos genéticos 46
2.11.14 Convergência prematura 50
2.11.15 Algoritmos genéticos com o número de indivíduos da população variável 51
2.11.16 Algoritmos genéticos desordenados 53
2.11.17 Algoritmos genéticos cíclicos 54
2.11.18 Manter a diversidade na evolução artificial 56
2.11.18.1 Introdução 56
2.11.18.2 Modelo AGs regime permanente (*steady state*) 56
2.11.18.3 Esquema de agrupamento 57
2.11.18.4 Esquema de partilha 57
2.11.18.5 Isolamento por distância 57
2.11.18.6 Prevenção de incesto 58
2.11.19 Algoritmos genéticos não binários 58
2.11.19.1 Introdução 58
2.11.19.2 Representação 59
2.11.19.3 Operador de cruzamento 59
2.11.19.4 Operador de mutação 60
2.11.19.5 Outros operadores de cruzamento 62
2.11.19.6 Outros operadores de mutação 62

2.12 Estratégias de evolução 63
2.12.1 Introdução 63
2.12.2 Representação 64
2.12.3 Operadores genéticos 65
2.12.3.1 Introdução 65
2.12.3.2 Operador de selecção 65
2.12.3.3 Operador de recombinação 66
2.12.3.4 Operador de mutação 67
2.12.4 Melhoramento da convergência 69
2.12.5 Comparação entre as estratégias de evolução e os algoritmos genéticos 69

2.13 Programação evolutiva **71**
2.13.1 Introdução 71
2.13.2 Comparação entre a programação evolutiva e as estratégias de evolução 71
2.13.3 Representação 72
2.13.4 Operadores genéticos 72
2.13.4.1 Introdução 72
2.13.4.2 Operador de selecção 72
2.13.4.3 Operador de mutação 73

2.14 Programação genética **74**
2.14.1 Introdução 74
2.14.2 Comparação entre a PG e os AGs 75
2.14.3 Representação 76
2.14.4 Função de aptidão 77
2.14.5 Algoritmo de programação genética 77
2.14.6 Operadores genéticos 78
2.14.6.1 Introdução 78
2.14.6.2 Reprodução 79
2.14.6.3 Operador de cruzamento 79
2.14.6.4 Operador de mutação 82
2.14.6.5 Operador de permutação 82
2.14.6.6 Operador de edição 83
2.14.6.7 Operador de encapsulamento 83
2.14.6.8 Operador de dizimação 84
2.14.7 Condição de finalização 84
2.14.8 Diplóide e dominância 85
2.14.9 Funções definidas automaticamente 85

2.15 Outros algoritmos evolutivos **86**
2.15.1 Introdução 86
2.15.2 Algoritmos genéticos baseados na ordem 86
2.15.3 Sistemas de classificação 86

2.16 Referências **87**

3 APLICAÇÃO DE ALGORITMOS GENÉTICOS À ROBÓTICA **89**

3.1 Introdução **89**

3.2 Planeamento de trajectórias para robôs móveis **89**
3.2.1 Introdução 89
3.2.2 Planeamento do movimento de um manipulador móvel 90
3.2.2.1 Introdução 90
3.2.2.2 Ambiente do robô 90
3.2.2.3 Representação 91
3.2.2.4 Função de aptidão 91
3.2.2.5 Operadores genéticos 92
3.2.3 Navegador/planeador evolutivo adaptativo 93
3.2.3.1 Introdução 93
3.2.3.2 Algoritmo do navegador/planeador 93
3.2.3.3 Representação das trajectórias 95
3.2.3.4 Função de aptidão 95
3.2.3.5 Operadores utilizados 97
3.2.3.6 Reprodução e selecção 97
3.2.3.7 Probabilidades adaptativas dos operadores 97

3.2.3.8 Experiências e resultados em modo *off-line* 98
3.2.3.9 Experiências e resultados em modo *on-line* 98
3.2.4 Planeamento de trajectórias com desvio dinâmico de obstáculos 99
3.2.4.1 Introdução 99
3.2.4.2 Representação das trajectórias 100
3.2.4.3 Função de aptidão 100
3.2.4.4 Resultados 101

3.3 Escolha e desenho de manipuladores robóticos 102
3.3.1 Introdução 102
3.3.2 Selecção do melhor robô 102
3.3.2.1 Introdução 102
3.3.2.2 Representação 102
3.3.2.3 Função de aptidão 103
3.3.2.4 Operadores genéticos 103
3.3.2.5 Problemas a resolver 103
3.3.3 Projecto de robôs modulares utilizando AGs 105
3.3.3.1 Introdução 105
3.3.3.2 Resolução do problema 105
3.3.3.3 Algoritmos genéticos utilizados 107
3.3.3.4 Função objectivo e função de aptidão 108
3.3.3.5 simulações 109
3.3.4 Projecto de um manipulador robótico através de AEs 111
3.3.4.1 Introdução 111
3.3.4.2 Algoritmo evolutivo multicromossoma 112
3.3.4.3 Operadores genéticos 114
3.3.4.4 Simulação 116
3.3.5 Desenho de manipuladores através de um AGMPs 118
3.3.5.1 Introdução 118
3.3.5.2 Operadores genéticos e representação 120
3.3.5.3 Função de aptidão 121
3.3.5.4 Projecto progressivo 122
3.3.5.5 Resultados 122

3.4 Estudo da locomoção de robôs 123
3.4.1 Introdução 123
3.4.2 Locomoção do “Stiquito” 124
3.4.2.1 Introdução 124
3.4.2.2 Representação 124
3.4.2.3 Operadores genéticos 125
3.4.2.4 Simulações e resultados 125
3.4.3 Geração de locomoção bípede usando um método hierárquico de geração de trajectórias 127
3.4.3.1 Introdução 127
3.4.3.2 Representação das *strings* 127
3.4.3.3 Operadores genéticos 129
3.4.3.4 Função de aptidão 130
3.4.3.5 Simulação 131
3.4.4 Metodologia auto-planeada 132
3.4.4.1 Introdução 132
3.4.4.2 Aproximação 132
3.4.4.3 Estudo de um caso 134
3.4.4.4 Resultados obtidos 135
3.4.4.5 Horizonte do robô 135
3.4.4.6 Aprendizagem do robô 136

3.5 Planeamento de trajectórias para manipuladores robóticos **137**
3.5.1 Introdução 137
3.5.2 Planeamento de trajectórias 137
3.5.2.1 Introdução 137
3.5.2.2 Representação da trajectória 137
3.5.2.3 Operadores genéticos 138
3.5.2.4 Função de aptidão 139
3.5.2.5 Mecanismo de selecção 140
3.5.2.6 Resultados 140
3.5.3 Planeamento de trajectória utilizando um AG com vírus 143
3.5.3.1 Introdução 143
3.5.3.2 Representação 145
3.5.3.3 Função de aptidão 146
3.5.3.4 Operadores relativos a infecção de vírus (camada de geração de posições) 147
3.5.3.5 Funções relativas à população de vírus 147
3.5.3.6 Operador de mutação (camada de geração de trajectórias) 148
3.5.3.7 Simulação 149
3.5.4 Optimização de trajectórias de dois robôs móveis 150
3.5.4.1 Introdução 150
3.5.4.2 Formulação do problema 150
3.5.4.3 Representação 151
3.5.4.4 Função de aptidão 152
3.5.4.5 Operadores genéticos utilizados 152
3.5.4.6 Resultados das simulações 153

3.6 Calibração de robôs **157**
3.6.1 Introdução 157
3.6.2 Planeamento óptimo de calibração de um robô 157
3.6.2.1 Introdução 157
3.6.2.2 Observabilidade do erro dos parâmetros 158
3.6.2.3 Formulação do problema 159
3.6.2.4 Função de aptidão 159
3.6.2.5 Representação 160
3.6.2.6 Operadores genéticos 160
3.6.2.7 Parâmetros do algoritmo genético 161
3.6.2.8 Experiências e resultados 161

3.7 Referências **163**

4 ALGORITMOS GENÉTICOS NO PLANEAMENTO DE TRAJECTÓRIAS PARA MANIPULADORES ROBÓTICOS **167**

4.1 Introdução **167**

4.2 Formulação do problema **167**

4.3 Variação da probabilidade de mutação **171**
4.3.1 Introdução 171
4.3.2 Resultado das experiências 172
4.3.3 Influência do operador de mutação 176

4.4 Variação da probabilidade de cruzamento **178**
4.4.1 Introdução 178
4.4.2 Resultados das experiências 178
4.4.3 Influência do operador de cruzamento 182

4.5 Variação dos coeficientes da função de aptidão **184**
4.5.1 Introdução 184
4.5.2 Resultado das experiências 184
4.5.3 Conclusões 187

4.6 Teste com um obstáculo simples **188**
4.6.1 Introdução 188
4.6.2 Resultado das experiências 188

4.7 Variação do comprimento do cromossoma **192**
4.7.1 Introdução 192
4.7.2 Ambiente sem obstáculos 192
4.7.2.1 Introdução 192
4.7.2.2 Resultado das experiências 192
4.7.3 Ambiente com obstáculos 196
4.7.3.1 Introdução 196
4.7.3.2 Resultado das experiências 196
4.7.3.3 Simulação para os robô de três e quatro elos 198
4.7.4 Conclusões 199

4.8 Simulação com dois obstáculos **199**
4.8.1 Introdução 199
4.8.2 Resultado da experiência com dois obstáculos 200

4.9 Ondulação residual **201**

4.10 Análise do tempo requerido pelo robô para planear as trajectórias **210**
4.10.1 Introdução 210
4.10.2 Tempos de simulação 210

5 CONCLUSÕES E PERSPECTIVAS PARA DESENVOLVIMENTOS FUTUROS **213**

5.1 Introdução **213**

5.2 Conclusões **213**

5.3 Perspectivas para desenvolvimentos futuros **214**

Índice de Figuras

FIGURA 2.1 AES GLOBAIS. 12
FIGURA 2.2 MIGRAÇÃO ENTRE VIZINHOS. 13
FIGURA 2.3 AES DIFUSOS. 13
FIGURA 2.4 CRUZAMENTO SIMPLES. 31
FIGURA 2.5 CRUZAMENTO DE PONTO DUPLO. 31
FIGURA 2.6 EXEMPLO DO CRUZAMENTO UNIFORME. 32
FIGURA 2.7 MUTAÇÃO DO BIT NÚMERO 5 DE UMA *STRING*. 34
FIGURA 2.8 OPERADOR INVERSÃO. 35
FIGURA 2.9 PROJECÇÃO ENTRE CROMOSSOMAS HOMÓLOGOS E O FENÓTIPO. 42
FIGURA 2.10 DOIS PROGRAMAS PAIS. 80
FIGURA 2.11 PROGRAMAS DESCENDENTES. 81
FIGURA 2.12 ÁRVORE APÓS A OCORRÊNCIA DE MUTAÇÃO NO NÓ 5. 82
FIGURA 2.13 ÁRVORES ANTES E DEPOIS DE OCORRER O OPERADOR DE PERMUTAÇÃO NO NÓ 2. 83
FIGURA 3.1 SISTEMA MANIPULADOR ROBÓTICO [13]. 90
FIGURA 3.2 CAMPO POTENCIAL NUMÉRICO [13]. 91
FIGURA 3.3 OPERADOR DE CRUZAMENTO [13]. 92
FIGURA 3.4 TRAJECTÓRIAS EM MODO *OFF-LINE* [4]. 98
FIGURA 3.5 NAVEGAÇÃO DO ROBÔ ON-LINE [4]. 99
FIGURA 3.6 ESTRUTURA DE CODIFICAÇÃO [21]. 100
FIGURA 3.7 TRAJECTÓRIA FINAL ENTRE DOIS PONTOS [21]. 101
FIGURA 3.8 AMBIENTE DA SIMULAÇÃO 1 [16]. 104
FIGURA 3.9 AMBIENTE DA SIMULAÇÃO 2 [16]. 104
FIGURA 3.10 SISTEMA DE COORDENADAS [5]. 106
FIGURA 3.11 RESULTADO DA SIMULAÇÃO DA TABELA 3.1 [5]. 110
FIGURA 3.12 RESULTADO DA TAREFA 2 [5]. 111
FIGURA 3.13 RESULTADOS DA TAREFA 2 [5]. 111
FIGURA 3.14 GENOMA HETEROGÉNEO [15]. 113
FIGURA 3.15 ESPECIFICAÇÃO DA TAREFA [15]. 117
FIGURA 3.16 COMPARAÇÃO ENTRE OS TRÊS ALGORITMOS [15]. 118
FIGURA 3.17 MANIPULADOR PLANAR COM 3 GRAUS DE LIBERDADE [6]. 119
FIGURA 3.18 TAREFA A EXECUTAR [6]. 119
FIGURA 3.19 PROJECTO PROGRESSIVO [6]. 122
FIGURA 3.20 $F_{1,MAX}$ V.S. NÚMERO DE GERAÇÕES [6]. 122
FIGURA 3.21 COMPRIMENTOS DOS ELOS V.S. NÚMERO DE GERAÇÕES [6]. 123
FIGURA 3.22 DIMENSÃO ÓPTIMA E A BASE DOS PONTOS DA TAREFA [6]. 123
FIGURA 3.23 RESULTADOS DO PROTÓTIPO E DO PLANEAMENTO NO ESPAÇO DAS JUNTAS [6]. 123
FIGURA 3.24 RESULTADO DO CONTROLO NO ESPAÇO DAS JUNTAS [6]. 123
FIGURA 3.25 NUMERAÇÃO DAS PERNAS DO “STIQUITO” [9]. 126
FIGURA 3.26 REPRESENTAÇÃO DOS ÂNGULOS DAS JUNTAS [19]. 128
FIGURA 3.27 TRAJECTÓRIA DA LOCOMOÇÃO BÍPEDE DO ROBÔ [19]. 132
FIGURA 3.28 ENERGIA CONSUMIDA PELOS ACTUADORES [19]. 132
FIGURA 3.29 MODULO DE ACÇÃO SIMPLES [18]. 133
FIGURA 3.30 REPRESENTAÇÃO DA *STRING* [18]. 133
FIGURA 3.31 HISTOGRAMA MÉDIO DO HORIZONTE DO ROBÔ [18]. 135
FIGURA 3.32 CONVERGÊNCIA DO AG [18]. 136
FIGURA 3.33 SOLUÇÃO DA MELHOR *STRING* PARA CINCO SIMULAÇÕES IGUAIS [22]. 141
FIGURA 3.34 ERRO MÉDIO DAS CINCO SIMULAÇÕES [22]. 142
FIGURA 3.35 ERRO MÉDIO E MELHOR ERRO DE UMA DAS CINCO EXPERIÊNCIAS [22]. 143
FIGURA 3.36 CODIFICAÇÃO NA CAMADA DE GERAÇÃO DE TRAJECTÓRIA [14]. 146
FIGURA 3.37 OPERADOR DE TRANSCRIÇÃO INVERSA [14]. 147
FIGURA 3.38 OPERADOR DE TRANSDUÇÃO: CÓPIA E REPOSIÇÃO [14]. 147

FIGURA 3.39 SIMULAÇÃO COM UM MANIPULADOR COM 7 GRAUS DE LIBERDADE [14]. 149
FIGURA 3.40 TRAJECTÓRIA RESULTANTE DA SIMULAÇÃO [14]. 149
FIGURA 3.41 (A) ESPAÇO OPERACIONAL DOS DOIS ROBÔS; (B) PONTOS DA TRAJECTÓRIA [2]. 151
FIGURA 3.42 APROXIMAÇÃO DOS ROBÔS POR CÍRCULOS [2]. 151
FIGURA 3.43 RESULTADOS DOS PONTOS TERMINAIS DOS ROBÔS PLANARES [2]. 154
FIGURA 3.44 BINÁRIOS APLICADOS AOS ROBÔS COM DOIS GRAUS DE LIBERDADE [2]. 154
FIGURA 3.45 REPRESENTAÇÃO DO ROBÔ PUMA 560 [2]. 155
FIGURA 3.46 TRAJECTÓRIAS FINAIS DOS ROBÔS COM 3 GRAUS DE LIBERDADE [2]. 156
FIGURA 3.47 PERCURSOS NO ESPAÇO DAS JUNTAS [2]. 156
FIGURA 3.48 DISTÂNCIA MÍNIMA ENTRE OS ROBÔS [2]. 156
FIGURA 3.49 BINÁRIO APLICADO AS JUNTAS DOS ROBÔS COM TRÊS GRAUS DE LIBERDADE [2]. 157
FIGURA 3.50 RESULTADO DAS EXPERIÊNCIAS [10]. 162
FIGURA 4.1 ROBÔ COM TRÊS ELOS. 168
FIGURA 4.2 REPRESENTAÇÃO DO CROMOSSOMA PARA O ROBÔ DE TRÊS ELOS. 168
FIGURA 4.3 RESULTADOS DO ROBÔ DE 2 ELOS (PC = 0,8; PM = 0,005). 173
FIGURA 4.4 RESULTADOS DO ROBÔ DE 2 ELOS (PC = 0,8; P_M = 0,03). 174
FIGURA 4.5 RESULTADOS DO ROBÔ DE 3 ELOS (PC = 0,8; P_M = 0,03). 175
FIGURA 4.6 RESULTADOS DO ROBÔ DE 4 ELOS (PC = 0,8; P_M = 0,03). 176
FIGURA 4.7 RESULTADO DE VÁRIAS EXPERIÊNCIAS. 177
FIGURA 4.8 RESULTADOS DO ROBÔ DE 2 ELOS (P_C = 0,1; P_M = 0,005). 179
FIGURA 4.9 RESULTADOS DO ROBÔ DE 2 ELOS (P_C = 0,8; P_M = 0,005). 180
FIGURA 4.10 RESULTADOS DO ROBÔ DE 3 ELOS (P_C = 0,8; P_M = 0,005). 181
FIGURA 4.11 RESULTADOS DO ROBÔ DE 4 ELOS (P_C = 0,8; P_M = 0,005). 182
FIGURA 4.12 RESULTADO DE VÁRIAS EXPERIÊNCIAS COM DIFERENTES PROBABILIDADES DE CRUZAMENTO. 183
FIGURA 4.13 RESULTADOS DO ROBÔ DE 2 ELOS (P_C = 0,8; P_M = 0,03). 185
FIGURA 4.14 RESULTADOS DO ROBÔ DE 3 ELOS (P_C = 0,8; P_M = 0,03). 186
FIGURA 4.15 RESULTADOS DO ROBÔ DE 4 ELOS (P_C = 0,8; P_M = 0,03). 187
FIGURA 4.16 RESULTADOS DO ROBÔ DE 2 ELOS (P_C = 0,6; P_M = 0,005). 189
FIGURA 4.17 RESULTADOS DO ROBÔ DE 3 ELOS (P_C = 0,6; P_M = 0,005). 190
FIGURA 4.18 RESULTADOS DO ROBÔ DE 4 ELOS (P_C = 0,6; P_M = 0,005). 191
FIGURA 4.19 RESULTADOS DO ROBÔ DE 2 ELOS (P_C =0,8; P_M = 0,03). 193
FIGURA 4.20 RESULTADOS DO ROBÔ DE 3 ELOS (P_C =0,8; P_M = 0,03). 194
FIGURA 4.21 RESULTADOS DO ROBÔ DE 4 ELOS (P_C =0,8; P_M = 0,03). 195
FIGURA 4.22 RESULTADO PARA O ROBÔ DE 2 ELOS (P_C = 0,6; P_M = 0,005). 197
FIGURA 4.23 RESULTADOS PARA O ROBÔ DE 3 E 4 ELOS (P_C = 0,6; P_M = 0,005). 198
FIGURA 4.24 RESULTADOS PARA ROBÔ COM 4 ELOS EM AMBIENTE COM 2 OBSTÁCULOS (P_C = 0,6; P_M = 0,005). 200
FIGURA 4.25 POSIÇÕES E VELOCIDADES PARA O ROBÔ DE 2 ELOS (RESOLUÇÃO DO PROBLEMA DA SECÇÃO 4.3, P_C = 0,8; P_M = 0,03). 201
FIGURA 4.26 POSIÇÕES E VELOCIDADES PARA O ROBÔ DE 3 ELOS (RESOLUÇÃO DO PROBLEMA DA SECÇÃO 4.3, P_C = 0,8; P_M = 0,03). 202
FIGURA 4.27 POSIÇÕES E VELOCIDADES PARA O ROBÔ DE 4 ELOS (RESOLUÇÃO DO PROBLEMA DA SECÇÃO 4.3, P_C = 0,8; P_M = 0,03). 203
FIGURA 4.28 POSIÇÕES E VELOCIDADES PARA O ROBÔ DE 2 ELOS (P_C = 0,8; P_M = 0,03). 205
FIGURA 4.29 POSIÇÕES E VELOCIDADES PARA O ROBÔ DE 3 ELOS (P_C = 0,8; P_M = 0,03). 207
FIGURA 4.30 POSIÇÕES E VELOCIDADES PARA O ROBÔ DE 4 ELOS (P_C = 0,8; P_M = 0,03). 208

Índice de Algoritmos

ALGORITMO 2.1 ESTRUTURA GERAL DE UM ALGORITMO EVOLUTIVO. 8
ALGORITMO 2.2 PSEUDO CÓDIGO DE AES DIFUSOS. 14
ALGORITMO 2.3 ALGORITMO GENÉTICO SIMPLES. 19
ALGORITMO 2.4 ALGORITMO AGSPV. 51
ALGORITMO 3.1 ALGORITMO NPE. 94
ALGORITMO 3.2 ALGORITMO HIERÁRQUICO. 129
ALGORITMO 3.3 ALGORITMO VEAG. 144
ALGORITMO 3.4 PROCEDIMENTO VÍRUS_INFECÇÃO. 145
ALGORITMO 4.1 GERADOR DE TRAJECTÓRIAS. 170
ALGORITMO 4.2 PROCEDIMENTO INICIALIZA_TRAJECTÓRIA. 171

Lista de Tabelas

TABELA 2.1 TERMINOLOGIA NATURAL/COMPUTACIONAL. 9
TABELA 2.2 SIMPLIFICAÇÃO DE PROGRAMAS EM LISP. 83
TABELA 3.1 ESPECIFICAÇÃO DA TAREFA 1 [5]. 109
TABELA 3.2 ESPECIFICAÇÃO DA TAREFA 2 (PONTOS E ORIENTAÇÃO DO PONTO TERMINAL) [5]. 110
TABELA 3.3 DIFERENÇAS ENTRE OS ALGORITMOS UTILIZADOS [15]. 116
TABELA 3.4 PARÂMETROS DA SIMULAÇÃO [19]. 131
TABELA 3.5 CARACTERÍSTICAS DOS ROBÔS RR [2]. 153
TABELA 3.6 PARÂMETROS DOS ROBÔS RRR [2]. 155
TABELA 3.7 PARÂMETROS DO AG INICIAL E FINAL [10]. 161
TABELA 4.1 VALORES DE APTIDÃO DAS SOLUÇÕES (P_C = 0,8). 172
TABELA 4.2 VALORES DE APTIDÃO DAS SOLUÇÕES (P_M = 0,005). 178
TABELA 4.3 VALORES DE APTIDÃO DAS SOLUÇÕES (P_C = 0,8; P_M = 0,03). 184
TABELA 4.4 VALORES DE APTIDÃO DAS SOLUÇÕES (P_C = 0,6; P_M = 0,005). 192
TABELA 4.5 VALORES DE APTIDÃO DAS SOLUÇÕES (P_C = 0,6; P_M = 0,03). 196
TABELA 4.6 TEMPO MÉDIO (SEGUNDOS) DE 4 SIMULAÇÃO COM 500 GERAÇÕES. 210

Lista de abreviaturas e símbolos

μ	–	Número de elementos (*strings*) da população
δ	–	Comprimento do *arranjo*
σ	–	Desvio padrão
λ	–	Número de descendentes
o	–	Ordem do *arranjo*
ρ	–	Pressão de selecção
π	–	Resolução do parâmetro
AE	–	Algoritmo evolutivo
AEs	–	Algoritmos evolutivos
AG	–	Algoritmo genético
AGC	–	Algoritmo genético cíclico
AGCs	–	Algoritmos genéticos cíclicos
AGDs	–	Algoritmos genéticos desordenados
AGs	–	Algoritmos genéticos
AGsPV	–	Algoritmos genéticos com o número de indivíduos variável
AGsRP	–	Algoritmos genéticos regime permanente
EE	–	Estratégia de evolução
EEs	–	Estratégias de evolução
f	–	Função de Aptidão
F	–	Soma das funções de aptidão da população
f.o.	–	Função objectivo
F_m	–	Média das funções de aptidão da população
F_{max}	–	Valor de aptidão máximo da população
l	–	Comprimento da *string*
M	–	Capacidade de manipulação (manipulabilidade)
M_r	–	Manipulabilidade relativa
O	–	Função objectivo
O_m	–	Média dos valores objectivo da população
p_c	–	Probabilidade de cruzamento
PE	–	Programação evolutiva
PF	–	Ponto final (da simulação)
PG	–	Programação genética
PP	–	Ponto que se pretende atingir (na simulação)
p_m	–	Probabilidade de mutação
p_r	–	Probabilidade de reprodução
q	–	ângulo da junta
SCs	–	Sistemas de classificação
t	–	Número da geração
t	–	Tempo
T	–	Número de gerações

Glossário

Clones	–	indivíduos idênticos.
Demes	–	subpopulações isoladas parcialmente, com pouca iteração entre as subpopulações, quando comparada com a iteração interna das subpopulações.
Diplóide	–	indivíduo onde os cromossomas são organizados em pares. Diz-se de um núcleo celular (célula, organismo, etc.) na fase em que apresenta duas séries de cromossomas.
Dominância	–	Processo em que um cromossoma é dominante relativamente ao seu par. As características apresentadas pelo indivíduo são provenientes do cromossoma dominante.
Evolvability	–	capacidade de esquemas de operadores/representação genético produzirem descendentes que são melhores do que os seus pais.
Esquemas *Panmictic*	–	Um indivíduo pode casar com um outro qualquer, *e.g*, esquema de agrupamento e esquema *Sharing*.
Genes	–	unidades que constituem um cromossoma.
Desvio genético (*Genetic drift*)	–	fenómenos que ocorrem em pequenas populações nas quais certas características se podem expandir, se bem que, eles não sejam particularmente mais aptos que os outros.
Haplóide	–	indivíduo que tem um só cromossoma.
Intron	–	porção do cromossoma que nunca é expresso por ser evidente e por providenciar espaço entre os genes.
Poliplóide (*polyploid*)	–	Organismo com vários (mais do que dois) cromossomas.

1 Introdução

1.1 Introdução

Nos últimos anos têm sido propostos algoritmos que fazem uma “mímica” de certos processos biológicos e que tomam o nome de algoritmos genéticos ou, de uma forma mais vasta, a designação de computação evolutiva. Esta área esta em forte expansão pelo que a sua sistematização não foi ainda totalmente realizada. Por outro lado, a aplicação destes conceitos no campo da robótica esta numa fase inicial pelo que existe um vasto campo de investigação e desenvolvimento de novos algoritmos.

A evolução é o principio de unificação principal da biologia moderna. A evolução Darwineana clássica juntamente com a selecção de Weismann e a genética de Mendel é a base para a teoria da evolução actual, ou, por outras palavras, a evolução é o resultado de processos estocásticos interactivos (reprodução, mutação, cruzamento e selecção) sobre populações ao longo de gerações. Todavia, a teoria da evolução estende-se para além dos sistemas biológicos, quando utilizada por máquinas computadorizadas com o fim de resolver problemas de optimização. A computação evolutiva é o termo que abrange uma serie de algoritmos entre os quais os algoritmos genéticos é um dos ramos principais.

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

1.2 Objectivos

O objectivo deste trabalho consiste em investigar a adopção de AGs no planeamento de trajectórias para manipuladores robóticos planares constituídos por 2, 3 e 4 juntas rotacionais. O espaço de trabalho pode incluir obstáculos e é restringido ao plano vertical.

Deste modo, é necessário desenvolver um programa, baseado num AG, para planear as trajectórias do manipulador robótico. Deve ser estudada a influência dos parâmetros do AG na determinação da solução final. Além disso, é estudado o comportamento das trajectórias, tanto a nível do deslocamento como a nível de velocidades das juntas do robô.

1.3 Estrutura da dissertação

O trabalho esta organizado em seis capítulos.

No capítulo 1 é elaborada uma introdução ao trabalho, aos seus objectivos e à estrutura da dissertação. No capítulo 2 são apresentados os aspectos principais dos algoritmos evolutivos, onde os algoritmos genéticos constitui um dos ramos principais. De seguida, no capítulo 3 é feito uma síntese de aplicações de sistemas robóticos que utilizam algoritmos evolutivos. No capítulo 4 é estudado o planeamento de trajectórias baseado na cinemática directa e no espaço operacional, para robôs de 2, 3 e 4 elos, utilizando AGs. Assim, é analisada a influência dos parâmetros no algoritmo e na determinação da solução final. São ainda desenvolvidas algumas modificações com vista a melhorar a ondulação residual da cinemática diferencial. Por último, no capítulo 5 apresenta-se um resumo dos pontos mais relevantes deste trabalho e são indicadas as perspectivas para desenvolvimentos futuros.

1.4 Referências

[1] David B. Fogel, *Editor-in-Chief* Natural Selection, Inc, La Jolla, CA 92037 USA, “Evolutionary Computation: A New Transations”, IEEE Transactions on Evolutionary Computation, Vol. 1, n. 1, pp. 1-2, April 1997.

[2] K. F. Man, K. S. Tang, S. Kwong, “Genetic Algorithms: Concepts and aplications”, IEEE Transations on Industrial, Electronics, Vol. 43, pp 519-534, October 1996.

2 Algoritmos Evolutivos

2.1 Introdução

Neste capítulo é feita uma abordagem dos principais algoritmos evolutivos e das suas propriedades. Em certos casos, para uma melhor compreensão, é feita uma analogia entre o processo natural e o algoritmo evolutivo "correspondente". Neste contexto são também apresentadas técnicas de forma a melhorar e a optimizar os algoritmos evolutivos propostos. Segundo esta ordem de ideias, nas secções 2.2 e 2.3 é feita uma introdução da selecção natural e dos tipos de algoritmos que existem. De seguida, nas secções 2.4, 2.5 e 2.6 é apresentada a estrutura geral de um algoritmo evolutivo, é estabelecida a correspondência entre as terminologias da genética natural e da ciência da computação e são referidos os parâmetros necessários a um algoritmo. Nas secções 2.7 a 2.10 são apresentados os diversos algoritmos evolutivos paralelos, é analisada a convergência de um algoritmo e são classificados os algoritmos evolutivos. Por último, nas secções 2.11 a 2.15 são apresentados, respectivamente, os algoritmos genéticos, as estratégias de evolução, a programação evolutiva, a programação genética e outros algoritmos evolutivos.

2.2 Introdução à teoria da selecção natural

A teoria da selecção natural, ou seja, a sobrevivência dos mais aptos, governa a adaptação evolutiva do mundo biológico, desde o vírus mais pequeno até ao mamífero mais complexo. A selecção natural opera em todo o organismo de modo a criar descendentes com as melhores características possíveis.

Os descendentes resultam, normalmente, do crescimento de uma célula que contém um exemplar de material genético dessa espécie. É através da criação desse material específico que os país influenciam a estrutura e funções hereditárias do descendente. Espécies diferentes criam material genético distinto para os respectivos descendentes, mas, em todas as espécies, os pais providenciam esse material. Em alguns casos, existem dois pais, sendo o material genético destes misturado para criar o descendente. Nos casos em que existe só um pai, o material genético hereditário sofre mutações (modificações aleatórias

do material genético) garantindo-se assim que o descendente não seja igual ao seu progenitor.

2.3 Aplicação da teoria de selecção natural à ciência da computação

A evolução da espécie através da capacidade de reprodução e a criação do plano genético dos descendentes, são temas centrais na adaptação evolutiva de organismos biológicos. Esta perspectiva pode ser aplicada à adaptação evolutiva de estruturas computacionais, resultando a área da computação evolutiva. Assim, a teoria da computação evolutiva consiste em aplicar os conceitos de selecção, baseados na aptidão de uma população de estruturas, num algoritmo de computador. A evolução é então um processo de optimização que pode ser simulado num computador adicionando-se eventualmente outros métodos comuns nas ciências de engenharia. O interesse destas simulações tem aumentado significativamente nos últimos anos. As aplicações deste novo tipo de algoritmos tem vindo a substituir os algoritmos clássicos em diversas áreas como, por exemplo, em sistemas de energia, reconhecimento de padrões, sistemas de controlo, escalonamento industrial, síntese farmacêutica e programas de análise económica e financeira.

Os algoritmos evolutivos imitam o processo de evolução natural que conduziu ao aparecimento de estruturas orgânicas complexas e bem organizadas. Por outras palavras, a evolução é o resultado da criação de material genético novo e da sua selecção. Com base num critério de desempenho, um indivíduo de uma população é afectado por outros indivíduos (*e.g.*, a competição por comida, a existência de predadores, e o acasalamento), e pelo ambiente (*e.g.* comida e clima existentes). Quanto melhor é a capacidade do indivíduo nestas funções maior é a possibilidade do indivíduo sobreviver e ter descendentes. Desta forma, a informação genética que normalmente passa para as gerações seguintes é a dos indivíduos mais aptos.

A natureza aleatória da reprodução conduz à introdução de nova informação genética e, consequentemente, à criação de descendentes com características diferentes.

Existem três ramos principais de estudo na evolução simulada: estratégias de evolução (EEs), programação evolutiva (PE) e algoritmos genéticos (AGs) (especialmente em programação genética (PG) e sistemas de classificação (SCs)). A computação evolutiva ou os algoritmos evolutivos (AEs) são os termos utilizados para denominar este tipo de técnicas. A diferença entre os algoritmos referidos consiste no tipo de alterações que são introduzidas nas soluções com vista a criar os respectivos descendentes, no método de selecção dos novos país e na estrutura de dados para representação das soluções.

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

2.4 Estrutura de um algoritmo evolutivo

Quando uma estratégia evolutiva é adoptada na resolução de problemas práticos, começa-se com uma população que contém soluções aleatórias. De seguida, são criadas aleatoriamente novas soluções modificando as soluções anteriores. Para estabelecer a aptidão ou inaptidão destas soluções é utilizada uma medida objectiva de desempenho. Um método de selecção determina quais são as soluções que devem ser pais da geração seguinte.

O algoritmo 2.1 descreve um método simples de um algoritmo evolutivo.

```
Programa_evolutivo
Início
t = 0
inicialização P(t)
avaliação P(t)
```

```
repetir enquanto não é verificada a condição de fim
   P'(t) = Variação P(t)
   Avaliação P'(t)
   P(t+1) = Selecção P'(t) ∪ Q
   t = t + 1
fim repetir
fim do algoritmo
```

Algoritmo 2.1 Estrutura Geral de um Algoritmo Evolutivo.

Neste algoritmo P(t) representa uma população de μ indivíduos na geração t. A população P(t), é inicializada aleatoriamente ou a partir de um processo heurístico. A variável Q representa um conjunto de indivíduos que podem ser considerados para selecção. Assim, o conjunto de indivíduos Q pode representar o conjunto P(t), um conjunto vazio ou ainda um conjunto dos melhores indivíduos. A população de λ descendentes (P'(t)) é gerada a partir de P(t) através de operadores tais como a recombinação (cruzamento), a mutação e inversão. A população de descendentes é avaliada através da função de aptidão, sendo calculado através desta o valor para cada indivíduo, distinguindo assim os indivíduos mais capazes. A escolha é feita com base no valor de aptidão de modo a conduzir o processo para a solução óptima.

2.5 Analogia entre os termos utilizados nos algoritmos evolutivos e na natureza

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

Tabela 2.1 Terminologia Natural/Computacional.

Termo natural	Termo utilizado nos AEs
Cromossoma, indivíduo	*String*
Gene	Caracter, parâmetro, característica
Alelo (*allele*)	Valor
Lugar (*locus*)	Posição na *string*
Genótipo	Estrutura
Fenótipo	Conjunto de parâmetros, soluções alternativas, estrutura de descodificação
Epistasis	Não-linearidade

2.6 Projecto de um algoritmo evolutivo

2.6.1 Introdução

Nesta secção são apresentadas as principais características de um AE.

2.6.2 Escolha do tipo e variações do algoritmo evolutivo

Para além das representações canónicas dos AGs, PE e EEs existe uma grande variedade na implementação dos algoritmos. As maiores diferenças entre as formas destes algoritmos tem haver com:

- A representação dos indivíduos (tipo de estruturas);
- O funcionamento dos operadores de variação (mutação e/ou recombinação);
- O mecanismo de selecção/reprodução.

Além disso, para cada tipo de algoritmo, podem existir variações consoante os:

- Métodos adoptados para criar a população inicial;

- Métodos de implementação das restrições do problema;
- Parâmetros utilizados (*i.e.* o número de *strings* da população, as probabilidades de ocorrência dos operadores, a percentagem de *strings* substituídas na população, etc.);
- Estratégias de selecção das *strings* (*i.e.* pura ou elitista).

2.6.3 Espaço de pesquisa

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

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

Existem muito resultados empíricos e teóricos que se encontram disponíveis para os algoritmos evolutivos normalizados. Este conhecimento constitui uma vantagem numa primeira aproximação e permite reutilizar os parâmetros e os operadores destes algoritmos. Por outro lado, funções de codificação complexas podem introduzir não linearidades e outras dificuldades matemáticas que podem prejudicar substancialmente o processo de pesquisa.

À *priori* não se sabe qual das soluções é a melhor para um determinado processo mas, diversas aplicações têm mostrado que os algoritmos modificados de acordo com o problema a resolver têm um desempenho superior aos algoritmos normalizados.

2.7 Algoritmos evolutivos paralelos

2.7.1 Introdução

Nesta secção são classificados os AEs paralelos de acordo com vários critérios.

Os AEs podem ser facilmente transportados para computadores de arquitecturas paralelas uma vez que as *strings* podem ser modificadas e avaliadas independentemente umas das outras. Deste modo deve-se esperar um aumento escalar da velocidade de resolução com o número de unidades de processamento *p*. A condição anterior é valida enquanto *p* não excede o valor máximo de *strings* da população (μ).Todavia, como a selecção opera sobre toda a população, o operador pode revelar um desempenho total inferior, especialmente para arquitecturas massivamente paralelas onde $p >> \mu$ .Esta característica motivou o desenvolvimento de algoritmos paralelos usando selecção local das subpopulações (de forma idêntica ao modelo de migração) ou como indivíduos vizinhos distribuídos espacialmente, de modo semelhante aos modelos difusos (também chamados algoritmos evolutivos celulares). Deve salientar-se que a selecção local não só aumenta consideravelmente a velocidade em arquitecturas paralelas, como também melhora a robustez do algoritmo.

2.7.2 Classificação dos algoritmos evolutivos paralelos

Existem uma série de métodos paralelos baseados nos AEs. Estes métodos paralelos podem ser classificados em: globais, de migração e difusos. Deste modo os algoritmos ficam agrupados de acordo com a natureza da estrutura da população e com o mecanismo de recombinação utilizado.

Os AEs globais aplicam em toda a população um mecanismo de reprodução simples. O algoritmo pode ser implementado num sistema multiprocessador com memória partilhada ou num sistema de memória distribuída. Num sistema de memória partilhada as

strings são guardadas nessa mesma memória onde cada processador avalia a *string* que lhe esta afectada. Num sistema de memória distribuída, o mecanismo é baseada na relação cliente-servidor. Uma desvantagem deste método é o facto dos clientes ficarem à espera do servidor enquanto este executa o seu trabalho. Este mecanismo pode ser visualizado na figura 2.1.

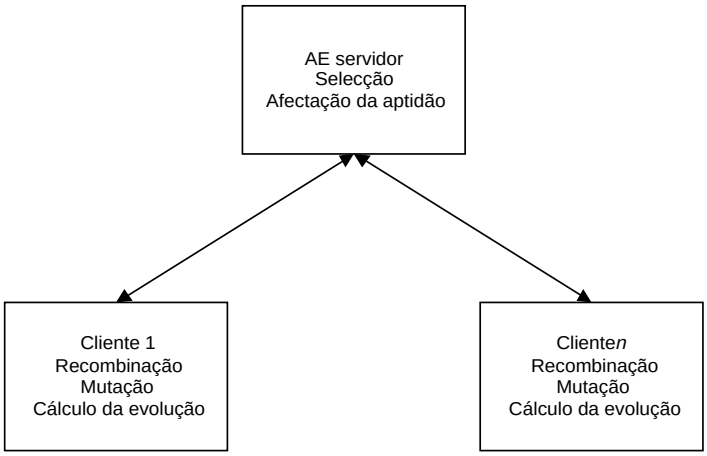


Figura 2.1 AEs globais.

O método de migração divide a população em várias subpopulações e cada subunidade é tratada de modo independente das restantes. Cada subpopulação executa um AE convencional seguido de uma emigração e uma imigração de *strings* entre as subunidades. A proliferação de material genético de boa qualidade, na população global, é feito através da migração entre as subpopulações de *strings* mais aptas (migração livre ou sem restrições). Uma estratégia semelhante é a migração entre vizinhos, ou seja, onde só é permitida a migração entre subpopulações vizinhas.

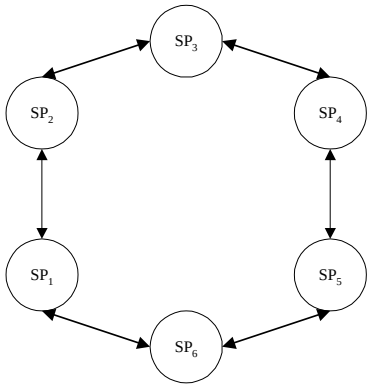


Figura 2.2 Migração entre vizinhos.

Por exemplo na figura 2.2 a migração da subpopulação 3 só ocorre entre as subpopulações 2 e 4 (uma seta entre duas subpopulações indicia a vizinhança destas). Se a migração é permitida num só sentido, então o algoritmo é classificado por migração em anel.

O caso dos AEs difusos é ilustrado na figura 2.3 onde a população é considerada única e continua (cada circunferência representa um processador). A cada *string* é afectada um processador, sendo permitido aos indivíduos procriar com os seus vizinhos de acordo com a arquitectura do computador paralelo.

No algoritmo 2.2 encontra-se um exemplo de um possível pseudo código de AEs difusos.

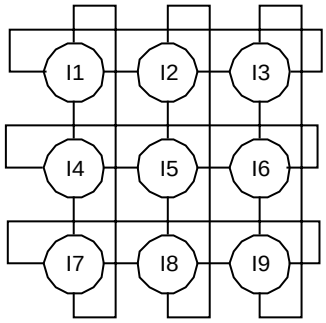


Figura 2.3 AEs difusos.

```
Procedimento AE_difuso
     Inicialização
     Repetir enquanto a condição de fim não é verificada
        Avaliar população
        Envio de copia da string para nós vizinhos
        Receber strings vizinhas
        Seleccionar parceiro para procriar
        Reprodução
     Fim repetir
retornar
```

Algoritmo 2.2 Pseudo código de AEs difusos.

2.8 Convergência do algoritmo

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

A analogia da convergência prematura nos AEs manifesta-se na natureza como o principio de preempção de nicho (*niche preemption*). De acordo com este principio, um nicho biológico na natureza tende a ficar dominados por uma espécie. Uma forma de minimizar os efeitos de preempção de nicho, convergência prematura, sensibilidade a condições iniciais e outros eventos aleatórios quando são utilizados AEs é fazer várias experiências independentes para o problema.

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

2.9 Classificação dos algoritmos evolutivos

Os problemas AEs são classificados em métodos fracos e métodos fortes. Um método fraco faz poucas suposições sobre o domínio do problema pelo que tem uma grande aplicabilidade. Todavia, quando transportado para problemas de grande dimensão as soluções, geradas através da explosão combinatória, vão aumentar o custo do método. Isto pode ser evitado através de um método forte considerando suposições relativas ao domínio do problema limitadas à aplicação em questão.

2.10 Variantes dos algoritmos evolutivos

Existe um grande número de variantes dos algoritmos evolutivos. Sendo de salientar os algoritmos genéticos baseados na ordem (*order-based*), os sistemas de classificação e a programação genética (PG). As variantes referidas nasceram como ramos dos algoritmos genéticos e desenvolveram as suas próprias direcções de pesquisa e de aplicação.

2.11 Algoritmos genéticos

2.11.1 Introdução

Os algoritmos genéticos, desenvolvidos por Holland em 1975 na Universidade de Michigan, só recentemente foram utilizadas para optimizações estruturadas. Os AGs são algoritmos de pesquisa baseados em mecanismos de selecção natural e na genética natural. Com estes mecanismos os indivíduos com melhores capacidades são normalmente os vencedores num ambiente altamente competitivo. Assim, os AGs combinam a sobrevivência dos indivíduos mais aptos com estruturas de *strings*. A informação criada aleatoriamente nestas estruturas é trocada para formar um algoritmo de pesquisa. Em cada geração é criada uma nova população artificial de *strings* desenvolvida a partir de *bits* e/ou conjunto de *bits* das *strings* mais aptas, substituindo a população anterior.

O tema central da investigação nos AG tem sido a robustez bem como o balanço entre a eficiência e a eficácia necessárias para a sobrevivência em diferentes ambientes. Os algoritmos genéticos funcionam bem num espectro largo do problema, enquanto que a maior parte dos algoritmos tradicionais funciona eficientemente apenas numa gama estreita de valores. Deve-se salientar que os AGs, tal como outros métodos heurísticos, não garantem que o valor óptimo seja encontrado.

Nas secções seguintes é feita a comparação entre os AGs e os métodos de optimização clássicos. São indicados os passos para criar um AG, é descrita a forma de representação, codificação e a função de aptidão. De seguida, são referidos os mecanismos de selecção, os operadores genéticos, as condições de finalização do algoritmo, o efeito da epistase e é descrito o problema ilusório. É também analisada a convergência do AGs e são apresentadas algumas variações dos AGs nomeadamente os AGs com o número de indivíduos da população variável, os AGs desordenados e os AGs cíclicos. Por último, são apresentados métodos para manter a diversidade da população e os AGs com representação não binária.

2.11.2 Comparação entre os algoritmos genéticos e os métodos clássicos

Os algoritmos genéticos 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;
- Os AGs usam informação de troca (funções objectivo), ao contrario do uso de derivadas e outros conhecimentos auxiliares;
- Os AGs usam regras de transição estocásticas, e não regras deterministas.

As vantagens dos AGs são:

- As restrições do problema são facilmente tratadas, inserindo-as na *string* de codificação.

- Os AGs é uma técnica capaz de resolver problemas multinodais, não diferenciáveis, não contínuos, não polinomiais, multidimensionais e redundantes;
- Os AGs estruturados providenciam uma ferramenta para optimizar a topologia, ou a estrutura paralela com os parâmetros da solução, para um problema particular.
- Os AGs são uma técnica simples de compreensão, requerendo poucos cálculos matemáticos.
- A *interface* com as simulações e com os modelos existentes é simples.
- Os AGs executam uma pesquisa multidireccional, mantendo uma população de soluções potenciais, encorajando a formação de informação e troca entre essas direcções.

Recentemente foram introduzidas nos AGs várias modificações com a finalidade de aumentar o desempenho e o espaço de pesquisa para um problema particular, nomeadamente, *strings* com comprimento variável, *strings* que incluem regras *se − então − senão* (PG) e estruturas mais ricas do que *strings* binárias (*e.g.* matrizes e alterações nos operadores genéticos).

Uma consequência dos AGs serem independentes do domínio é a sua incapacidade de resolver problemas com restrições não triviais. Estas restrições podem ser implementadas introduzindo penalidades nos indivíduos que não satisfazem essas restrições (ao nível da função de aptidão) ou criando funções de descodificação para representar as *strings*, de modo a evitar *strings* que não verifiquem essas restrições.

Se é introduzida uma penalidade muito grande na função de aptidão e se as restrições do domínio podem ser violadas pela representação dos indivíduos, então o algoritmo pode criar um grande número de *strings* ilegais. Por um lado pode acontecer que, quando é encontrada uma *string* que satisfaz essas restrições, esta conduza as restantes na sua direcção sem se encontrarem as melhores soluções. Por outro lado, se a penalidade introduzida é pequena, o sistema pode apresentar soluções que não verifiquem as restrições com um desempenho superior ao de outras *strings* que satisfazem as mesmas restrições. Isto

deve-se à parte da solução que satisfaz o domínio de uma *string* apresentar um desempenho superior as outras *strings*.

Se a representação é tal forma que a construção de indivíduos ilegais não seja possível, estão normalmente o problema é bastante intensivo computacionalmente e é difícil implementar todas as restrições.

2.11.3 Projecto de um algoritmo genético

Para desenvolver um AG é necessário:

- Escolher um *esquema* de representação considerando:
  i) o tamanho do alfabeto;
  ii) a selecção e projecção entre a *string* e os pontos do espaço.
- Definir a função de aptidão.
- Determinar os parâmetros e as variáveis necessários para controlar o algoritmo entrando em linha de conta com:
  i) o número de *strings* da população;
  ii) o número de gerações;
  iii) a probabilidade de reprodução (p_r);
  iv) a probabilidade de cruzamento (p_c);
  v) a probabilidade de mutação (p_m);
  vi) a percentagem de população que vai ser substituída.
- Determinar o modo de guardar a *string* que representa o melhor resultado.
- Estabelecer o critério para terminar o processo.

Após a escolha dos requisitos necessários para a implementação do AG, a estrutura de um AG simples é a seguinte:

```
Inicio
t = 0
inicializar aleatoriamente P(t)
```

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

Algoritmo 2.3 Algoritmo genético simples.

Onde P(*t*) é a população na geração *t*.

As funções do algoritmo serão explicadas em subsecção seguinte.

2.11.4 Representação e codificação dos algoritmos genéticos

Os AGs canónicos utilizam uma representação binária nas *strings* (que têm comprimento fixo) podendo cada caracter da *string* ter o valor "0" ou "1". Os AGs utilizam frequentemente funções de codificação e descodificação, de modo a facilitar a projecção entre as soluções do problema e as *strings* binárias. O sucesso dos AGs é muitas vezes dependente da função de codificação. No caso de problemas de optimização com parâmetros contínuos, os algoritmos genéticos representam normalmente o conjunto de parâmetros reais por uma *string*, sendo esta dividida em vários segmentos (um por cada parâmetro (gene)) do mesmo comprimento. Cada segmento contém um valor inteiro que é projectado linearmente no intervalo que o parâmetro real correspondente pode tomar. Este valor pode ser calculado através de *l* caracteres com uma resolução π de acordo com a seguinte formula:

$$\pi = \frac{U_{\max} - U_{\min}}{2^l - 1}$$

onde:

l é o número de caracteres para cada parâmetro;

U_{max} é o valor máximo do parâmetro a codificar (o valor de U_{max} codificado é $11...1_b$);

U_{min} é o valor mínimo do parâmetro a codificar (o valor de U_{min} codificado é $00...0_b$).

Um determinado valor para o parâmetro é codificado através da expressão:

$$V_c = \pi \times V_r$$

onde:

V_c é o parâmetro codificado;

V_r é o parâmetro real;

π é a resolução do parâmetro codificado.

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

O argumento fundamental que justifica o uso do alfabeto binário nos algoritmos genéticos é devido ao facto do número de *arranjos* ser máximo para um conjunto de pontos de pesquisa num algoritmo binário. Consequentemente, o teorema do *esquema* favorece a representação binária das soluções. Além da sua simplicidade o uso do alfabeto binário facilita a análise teórica e permite operadores genéticos estruturalmente elegantes. Todavia a representação binária tem a desvantagem da função de codificação poder introduzir multimodalidade adicional, fazendo com que a função binária de optimização se torne mais complexa do que a função real inicial.

2.11.5 Função de aptidão

A função de aptidão é o único meio de quantificar a utilidade (*i.e.* o valor para atingir o objectivo) de uma *string*. Esta função constitui uma ligação importante entre os AGs e o sistema a tratar. A função de aptidão deve ser semi-definida positiva. Contudo, existem problemas em que interessa minimizar o custo da função pois maximizar um problema não garante que a função de utilidade tenha valores não negativos para algumas *strings*. Assim, é frequente projectar a função objectivo numa função de aptidão.

Para transformar um problema de minimização num problema de maximização pode utiliza-se a seguinte função:

$$O_i = \begin{cases} C_{\max} - G_i & \text{para } G_i < C_{\max} \\ 0 & \text{para os casos restantes} \end{cases}$$

onde:

a função G_i é a função objectivo inicial;

a função O_i é a função objectivo;

o valor $C_{\max}$ é o valor máximo observado em G_i.

Se o problema é de maximização então:

$$O_i = \begin{cases} C_{\min} + G_i & \text{para } G_i + C_{\min} > 0 \\ 0 & \text{para os casos restantes} \end{cases}$$

onde:

a função G_i é a função objectivo inicial;

a função O_i é a função objectivo;

o valor $-C_{\min}$ é o valor negativo mínimo observado em G_i.

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

i) o modelo é executado;

ii) as funções objectivo são avaliadas;

iii)verifica-se se existe alguma restrição que não é satisfeita;

iv)se existir alguma restrição que não é verificada, então a solução não é admissível e, consequentemente, a sua função de aptidão tem o valor zero.

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

No método das penalidades um problema com restrições é transformado noutro sem restrições. Esta transformação é feita associando um custo, ou penalidade, a todas as restrições que não são satisfeitas. Este custo é incluído na função objectivo. Assim, o seguinte problema com restrições:

$$\text{Min } \{g(x)\}$$
$$\text{Sujeito a. } b_i(x) \geq 0 \; i = 1,2,\ldots,n$$

é transformado no seguinte problema sem restrições:

$$\text{Min } \{g(x)\} + r \times \sum_{i=1}^{n} \phi\,[b_i(x)]$$

onde:

x é um vector de dimensão m;

$g(x)$ é a função a minimizar;

b_i é a restrição i do problema;

ϕ é a função de penalidade;

r é o coeficiente de penalidade.

A função de penalidades pode ter muitas representações como, por exemplo:

$$\phi\,[b_i(x)] = b_i^2(x)$$

A função de aptidão varia de acordo com o problema a resolver, nomeadamente com os parâmetros que se pretendem optimizar e com as restrições do problema. Para manter a uniformidade entre os diversos problemas, o valor objectivo (O_i) é projectado no valor de aptidão.

As escalas mais frequentemente utilizadas são: escalonamento linear, escalonamento por potência e truncação sigma. As suas características são:

- Escalonamento linear (*linear scaling*): o valor de aptidão (f_i) da *string i* tem a seguinte relação linear com o valor objectivo:

$$f_i = a \times O_i + b$$

  onde os coeficientes *a* e *b* podem ser escolhidos de diversos modos. Um exemplo consiste em assegurar a igualdade entre a média da função objectivo (O_m) e da média dos valores de aptidão. Assim, com o uso do procedimento de selecção, espera-se que cada membro da população em média contribua com um descendente para a próxima geração. A aptidão máxima pode ser um múltiplo da média da aptidão, para controlar o número de descendentes de uma *string* na população com a função objectivo máxima. Quando se utiliza esta escala deve-se ter o cuidado com os valores de aptidão negativos que são introduzidos. Se os parâmetros *a* e *b* forem constantes ao longo das gerações este escalonamento é independente do problema.

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

$$f_i = O_i^k$$

onde *k* varia de acordo com o problema ou, eventualmente, durante a sua execução (normalmente o valor de *k* utilizado é próximo de 1 (*e.g.* $k = 1{,}005$)).

- Truncação sigma: O valor de aptidão (f_i), da *string i*, é calculada de acordo com a seguinte expressão:

$$f_i = O_i - (O_m - c \times \sigma)$$

onde *c* é um valor inteiro pequeno (normalmente $1 \leq c \leq 5$), O_m é a média dos valores objectivos e σ é o desvio padrão da população.

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

A convergência do problema depende da função de aptidão. Por exemplo, considere-se as funções de aptidão $f_1(x)$ e $f_2(x)$:

$$f_1(x) = f_2(x) + c$$

onde *c* é uma constante. Se $c >> F_{m1}(x)$ (F_m – média da função de aptidão) então a função $f_2(x)$ vai ter uma convergência muito mais pequena que a função $f_1(x)$. No caso extremo a função $f_2(x)$ terá uma pesquisa aleatória enquanto que a função $f_1(x)$ terá uma convergência prematura.

2.11.6 Mecanismo de selecção

2.11.6.1 Introdução

Para gerar descendentes com boas características é necessário um bom mecanismo de selecção de pais proficientes. Este processo é adoptado para determinar o número de casamentos, utilizadas na reprodução, para um indivíduo em particular. A probabilidade de escolher uma *string* como pai deve ser directamente proporcional ao número de descendentes produzidos.

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

2.11.6.2 Desempenho do mecanismo de selecção

De acordo com [6] existem três medidas de desempenho no mecanismo de selecção: direcção (*bias*), extensão (*spread*), e eficiência. A medida direcção indica a diferença absoluta entre as probabilidades actuais e esperadas das *strings* serem seleccionadas. A medida extensão é o intervalo entre zero e o número possível de casamentos que um indivíduo pode realizar. A eficiência está relacionado com o tempo de execução do algoritmo.

O método de selecção do tipo roleta redonda tem uma direcção próxima de zero, mas tem uma extensão ilimitada. O algoritmo pode ser implementado com um tempo na ordem de $\mu \times \log\mu$, onde μ é o número de *strings* da população. Outro algoritmo de amostragem de fase simples consiste na amostragem estocástica universal (s*tochastic universal sampling*) com direcção nula, extensão mínima e um tempo de execução do algoritmo na ordem de μ. Existem ainda outros métodos como, por exemplo, a selecção baseada no posto. Esta selecção introduz uma alternativa da afectação da aptidão pois as *strings* são seleccionadas proporcionalmente ao posto.

Existem dois temas importantes na pesquisa genética: a diversidade da população e a pressão da selecção (medida proporcional correspondente à diferença de valores de apti-

dão que as *strings* podem apresentar). Se a função de selecção é muito apurada (*i.e*. selectiva) a diversidade da população tende a diminuir. Se, por outro lado, o método é pouco selectivo então a diversidade da população tende a aumentar. Assim, se o método de selecção é muito apurado o método tende a convergir prematuramente.

2.11.6.3 Classificação dos operadores de selecção

Existem várias classificações dos métodos de selecção, entre os quais:

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

- métodos *preservativos* versus métodos *extintivos*. Os métodos *preservativos* requerem que a probabilidade de selecção de uma *string* seja não nula. Os métodos *extintivos* podem ainda subdividir-se em selecção *esquerda* e *direita*. No método *extintivo esquerdo* as melhores *strings* são proibidas de se reproduzirem, evitando eventuais convergências prematuras. Na selecção pelo método *extintivo direito* a melhor *string* tem uma reprodução normal.

- uma selecção diz-se *pura* se o tempo de vida das *strings* é de uma geração.

- métodos geracional versus métodos substituição imediata. Quando são fixados os pais, até a nova geração seja completa, a selecção designa-se por *geracional* (*generational*). Contrariamente à selecção de *substituição imediata* onde os descendentes são inseridos na população após a sua criação.

- os modelos *elitistas* permitem que alguns pais passem para a geração seguinte.

2.11.6.4 Operadores de selecção

2.11.6.4.1 Introdução

O operador de selecção é baseado exclusivamente nos valores de aptidão dos indivíduos. Nos AGs a selecção é normalmente implementada como um operador estocástico, utilizando a aptidão relativa, assim a probabilidade de selecção (proporcional) do indivíduo a_i de uma população com μ elementos é dada por: $p(a_i) = f(a_i) / \sum_{j=1}^{\mu} f(a_i)$. Este método requer valores de aptidão positivos e maximização da tarefa. Nesta perspectiva, são utilizadas projecções para transformar a aptidão na gama de valores desejados.

Nesta secção são descritos alguns operadores e algumas variações no modelo de selecção.

2.11.6.4.2 Selecção por nível (posto)

Este operador utiliza os índices dos indivíduos, quando ordenados de acordo com os valores de aptidão, para calcular a probabilidade de selecção correspondente. Neste caso foram propostas projecções:

lineares:

$$p(nível) = q - (nível - 1) \times r$$

não lineares:

$$p(nível) = q \times (1 - q)^{nível - 1}$$

Ambas as funções determinam a probabilidade de um indivíduo ser escolhido em uma amostra. Considera-se que o indivíduo se encontra cotado numa escala de nível (*nível* = 1 (*nível* = μ) quando o indivíduo em questão é o melhor (pior) da população).

As duas funções satisfazem a equação:

$$\sum_{i=1}^{\mu} p_i = 1$$

ou seja:

$$q = 0{,}5 \times r \times (\mu - 1) + \mu^{-1}$$

onde:

r é um parâmetro escolhido pelo utilizador.

μ é o número de indivíduos da população.

O parâmetro r, escolhido pelo utilizador, diz respeito à pressão do método de selecção (para $r = 0$ não existe pressão no método, isto é, todos os indivíduos têm a mesma probabilidade de serem seleccionados, contrariamente para $r = 2 / (n^2 - n)$ a selecção do método é máxima).

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

- A responsabilidade de utilizar este método é deixada ao cargo do utilizador;
- O algoritmo ignora a informação relativa às outras *strings*;
- O método trata todos os casos de modo uniforme, independente do tipo de problema;
- O algoritmo não verifica o teorema do *esquema*.

2.11.6.4.3 Selecção por torneio (*tournament selection*)

Este tipo de selecção retira uma amostra de q ($q > 1$) *strings* da população, segundo uma lei aleatória uniforme. A melhor *string*, da amostra, é copiada para a geração seguinte e a operação é repetida até que o número de *strings* seleccionadas preencha a nova geração. Este método tem bastante popularidade devido à sua fácil implementação, do

ponto de vista de eficiência computacional, e permite apurar o peso da selecção pelo aumento ou diminuição do tamanho do torneio (q).

2.11.6.4.4 Selecção de Bolzmann

Esta selecção tem semelhanças com o anterior, para $q = 2$, onde as duas *string* competem entre si de acordo com a fórmula:

$$\frac{T}{1+e^{f(i)-f(j)}}$$

onde:

T é a temperatura;

f(k) é a função objectivo da *string* k.

Esta fórmula é utilizada na minimização de problemas.

2.11.6.4.5 Variações do modelo de selecção

Existem variações do modelo de selecção nomeadamente:

- Modelo elitista. Este método força a melhor *string* a estar presente na geração seguinte.

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

- Modelo elitista do valor esperado . Este modelo é idêntico ao anterior excepto para a melhor *string* é sempre passada para a geração seguinte.

- Modelo do factor de agrupamento (*crowding*). Neste algoritmo por cruzamento dos pais é gerada uma *string*, que substitui uma das *strings* que lhe deu origem.

- Modelo de amostragem estocástica (*remainder stochastic sampling with replacement*). O algoritmo selecciona *strings* de acordo com a parte inteira do valor esperado do número de ocorrências destas. A população é completada pela competição da parte decimal do valor de aptidão das *strings*. Este método é o que apresenta melhores resultados de todos os métodos.

- Modelo de amostragem estocástica universal (*stochastic universal sampling*). Este algoritmo utiliza uma roleta redonda (esta roleta é construída da mesma forma que no método da roleta redonda), com um número de marcas igualmente espaçadas, que é rodada até atingir o tamanho da população (em oposição ao caso de ser rodada apenas uma vez).

2.11.7 Operadores genéticos

2.11.7.1 Introdução

Existem três grandes tipos de operadores de variação nos algoritmos genéticos: recombinação, mutação e inversão. Dentro destes principais tipos existe ainda uma grande variedade de operadores mas todos estes operadores devem obedecer a várias propriedades, como por exemplo, o teorema do *esquema*.

Nos AGs canónicos são utilizados apenas os operadores de cruzamento binário e de mutação binária.

Nesta secção são apresentados e estudados os principais operadores.

2.11.7.2 Operador de cruzamento

O algoritmo canónico utiliza o cruzamento (recombinação) de ponto simples. Para esse efeito são escolhidos, aleatoriamente, dois indivíduos da população. De seguida é escolhido, aleatoriamente, um ponto nas *strings*, que consiste no ponto de cruzamento. Desta forma, um descendente é criado a partir do cruzamento da parte esquerda de uma *string* com a parte direita da outra *string*. As partes restantes serão utilizadas para formar um segundo descendente ver Figura 2.4. Este operador permite várias extensões tais como o número de pontos de cruzamento ser superior a um (cruzamento multiponto, ver Figura 2.5) e o cruzamento ser uniforme. Este operador é aplicado à população com uma probabilidade p_c (normalmente $p_c \approx 0{,}6$ para populações grandes $\mu \geq 100$ e $p_c \approx 0{,}9$ para populações pequenas $\mu \leq 30$). O aumento da probabilidade de cruzamento provoca o aumento da recombinação na construção de blocos.

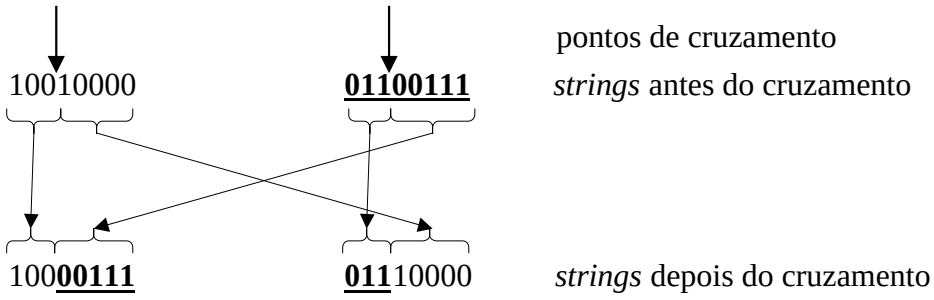


Figura 2.4 Cruzamento simples.

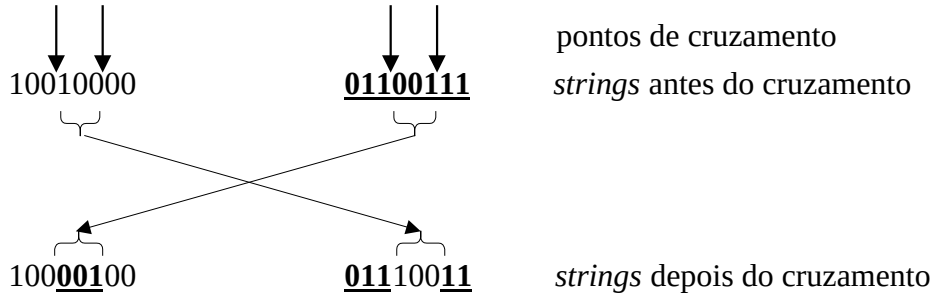


Figura 2.5 Cruzamento de ponto duplo.

2.11.7.2.1 Cruzamento uniforme

No cruzamento uniforme cada *bit* do descendente é escolhido aleatoriamente de um *bit* dos pais. Para esse fim é utilizada uma máscara gerada aleatoriamente. Um exemplo de um cruzamento deste tipo é exemplificada na figura 2.6.

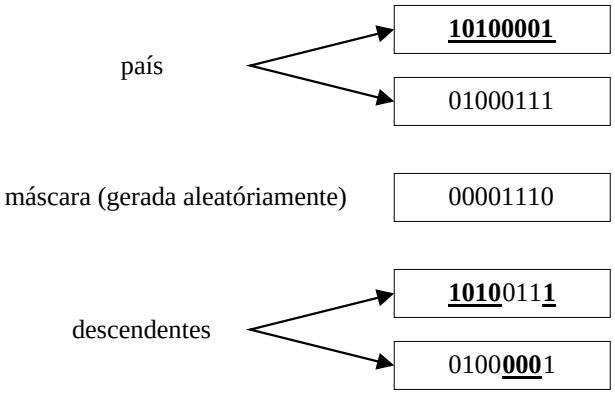


Figura 2.6 Exemplo do cruzamento uniforme.

O número de pontos de cruzamento não é fixo mas tem, em média, $l / 2$ pontos de cruzamento (onde l é o comprimento de uma *string*).

2.11.7.2.2 Recombinação com vários pais

A recombinação com vários pais consiste em criar um descendente a partir de vários pais (*i.e*. mais do que dois pais).

2.11.7.2.3 Cruzamento análogo

Em contraste com o cruzamento tradicional, onde o ponto de cruzamento é escolhido de acordo com a posição da *string*, no cruzamento análogo [15] o local de cruzamento é baseado em semelhanças na características do genótipo. Este operador usa uma função fenotípica de parâmetros como critério do ponto de cruzamento. O cruzamento análogo

não só preserva a ordem da característica da *string* como também tem uma justificação biológica (*i.e*. orgão análogos são orgãos ou partes de corpos adaptados para servir o mesmo propósito). O cruzamento de parâmetros de acordo com o seu carácter genotípico é superior ao cruzamento de acordo com a sua posição genotípica, para *strings* onde o número, o tamanho e a posição dos parâmetros não tem uma estrutura rígida. Neste tipo de estruturas é importante que o cruzamento ocorra entre locais que controlam a mesma função ou uma função similar no espaço fenotípico.

2.11.7.2.4 Cruzamento de segregação

Na natureza o local de cruzamento pode ocorrer entre genes ou até dentro de genes. O cruzamento entre genes (*i.e*. o cruzamento que não divide genes) é chamado cruzamento de segregação [15]. Quando os parâmetros com vários símbolos têm significado natural, o cruzamento dentro destes grupos pode ter um efeito de ruptura no genótipo e, consequentemente, no fenótipo.

2.11.7.3 Operador de mutação

O operador de mutação foi introduzido como um operador secundário e de menor importância. O operador canónico de mutação inverte os *bits* com uma probabilidade p_m bastante baixa (normalmente $p_m \approx 0{,}001$ para grandes populações $\mu \geq 100$ indivíduos e $p_m \approx 0{,}01$ para populações pequenas, $\mu \leq 30$ indivíduos, $p_m \in [0{,}005;\ 0{,}01]$ ou $p_m = 1 / l$ (onde l é o comprimento da *string*) de acordo com vários autores [12]). A mutação é responsável pela perca de alelos (*alleles*), prevenindo assim possíveis convergências para óptimos locais, e introduzir uma pequena pesquisa aleatória. Probabilidades de mutação elevadas aumentam o tempo do algoritmo, mas ajudam na sua convergência. De facto o aumento da probabilidade de mutação transforma o AG num algoritmo de pesquisa aleatório e permite a reintrodução de material genético perdido.

Este operador é aplicado a todos os descendentes (filhos) depois da recombinação (cruzamento). Cada *bit* pode ser modificado com uma probabilidade p_m. O *bit* que sofre a mutação é modificado para o valor complementar (no caso do alfabeto ser binário). Deste

modo garante-se que todos os pontos de pesquisa têm probabilidade não nula de serem examinados.

Na figura 2.7 está ilustrado a alteração do *bit* 5 (a negro) de uma *string* quando sofre uma mutação nesse *bit*. Neste caso o *bit* que tem o valor "1", após a mutação passa a ter o valor "0".

string inicial:

1 0 1 1 **1** 0 1 1

1 2 3 4 5 6 7 8

string depois de sofrer a mutação:

1 0 1 1 **0** 0 1 1

1 2 3 4 5 6 7 8

Figura 2.7 Mutação do bit número 5 de uma *string*.

2.11.7.4 Operador de reordenação/inversão

A finalidade da reordenação é tentar encontrar ordenações dos genes que tenham um potencial evolutivo superior ao normal. O principal mecanismo que deu origem à reordenação foi o operador de inversão. A inversão consiste em:

i) escolher dois pontos aleatórios na *string*;
ii) os *bits* entre esses dois pontos são retirados da *string*;
iii) os *bits* são recolocados por ordem inversa.

Na figura 2.8 está ilustrado o funcionamento do operador inversão.

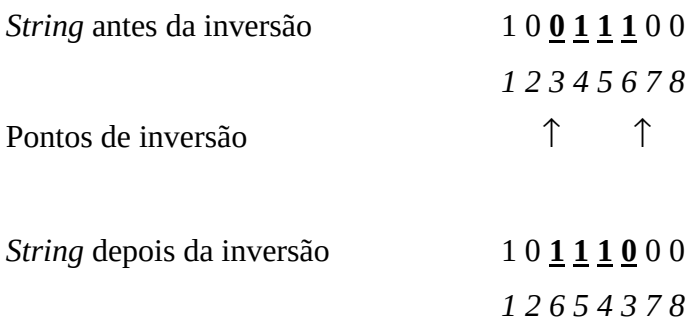


Figura 2.8 Operador inversão.

Na natureza os genes são independentes do lugar que ocupam. Para ter uma certa flexibilidade na representação utilizam-se nome para os genes (na figura 2.8 são utilizados os números de 1 a 8). Com esta representação o valor dos símbolos é independente do seu lugar. Assim, o operador de inversão não tem qualquer influência na função de descodificação e na função de aptidão.

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

Os seguintes operadores de reordenação combinam características de inversão e cruzamento: cruzamento parcialmente semelhante (*Partially matched crossover* – PMX), cruzamento ordenado (*order crossover* – OX) e cruzamento cíclico (*cycle crossover* – CX).

Por exemplo, num problema TSP onde cada cidade (1...8) é visitada por ordem ascendente tem-se a representação:

1 2 3 4 5 6 7 8

Com esta representação a função de aptidão do problema TSP só depende da posição dos alelos. 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. O poder da pesquisa genética incide no efeito da selecção e da estrutura. Para criar operadores com o poder de cruzamento, devem ser operadores binários que devem combinar a construção ordenada de blocos a partir dos pais.

O operador PMX escolhe aleatoriamente dois pontos de cruzamento para as duas *strings* pais. Esses dois pontos definem a secção de casamento que é usada para efectuar operações de trocas de posição. Considerando as duas *strings*:

A = 8 4 | 5 6 7 | 1 3 2
B = 7 1 | 2 3 8 | 6 5 4

De seguida procede-se à troca de cidades fazendo uma projecção entre as cidades que se encontram dentro da secção de casamento. Assim, dentro das *strings*, A e B são feitas as seguintes trocas: as cidades 5, 6 e 7 trocam, respectivamente, com as cidades 2, 3 e 8. Deste modo as *strings* descendentes A’ e B’ serão as seguintes:

A’ = 7 4 | 2 3 8 | 1 6 5
B’ = 8 1 | 5 6 7 | 3 2 4

O operador OX começa por fazer a mesma projecção que o operador PMX.

A = 8 4 | 5 6 7 | 1 3 2
B = 7 1 | 2 3 8 | 6 5 4

De seguida, o operador utiliza um movimento deslizante para preencher os lugares deixados vagos (*i.e*. os pares de cidades iguais aos da zona de casamento) pela transferência das zonas de casamento.

A’ = - 4 | 2 3 8 | 1 - -
B’ = - 1 | 5 6 7 | - - 4

Os lugares vazios deslizam para a direita a partir do segundo ponto de cruzamento.

A’ = - - | 2 3 8 | 1 4 -
B’ = - - | 5 6 7 | 4 1 -

As cidades que foram trocadas vão ocupando os lugares vazios a partir do segundo ponto de cruzamento. Por último, obtêm-se as *strings* descendentes:

A’ = 6 7 | 2 3 8 | 1 4 5
B’ = 3 8 | 5 6 7 | 4 1 2

O operador CX executa a recombinação sob a restrição de cada cidade provém de um pai para outro. Para exemplificar este operador consideremos os caminhos A e B dados por:

A = 8 4 5 6 7 1 3 2
B = 7 1 2 3 8 6 5 4

Começa-se por escolher, aleatoriamente, uma cidade do primeiro pai resultando:

A’ = 8 - - - - - - -

De seguida, escolhe-se a cidade 7 da *string* A, (já que a cidade 7 da *string* B corresponde à cidade 8 da *string* A) resultando:

A' = 8 - - - 7 - - -

A selecção da cidade 7 significa que a cidade seguinte a ser escolhida é a cidade 8. Como isto não é possível, pois a cidade 8 já faz parte da *string* A', conclui-se o ciclo. As cidades restantes são seleccionadas a partir da *string* B.

A' = 8 1 - - 7 - - -
A' = 8 1 - - 7 - - 4
A' = 8 1 2 - 7 - - 4
A' = 8 1 2 - 7 - 5 4

Ficando a *string* A' da forma:

A' = 8 1 2 3 7 6 5 4

Se, entretanto, fosse concluído outro ciclo, as cidades passariam a ser retiradas na *string* A.
De modo semelhante obtém-se a outra *string*:

B' = 7 4 5 6 8 1 3 2

Em geral o operador PMX tende a respeitar a posição absoluta das cidades, enquanto que o operador OX tende a respeitar a posição relativa das cidades.

2.11.7.5 Outros operadores

2.11.7.5.1 Introdução

Além dos operadores principais já apresentados, existem outros a referir tais como: translocação, duplicação, remoção, sexuais, restrição no casamento, reinserção e dominância. Estes operadores são descritos em seguida.

2.11.7.5.2 Operador translocação

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

2.11.7.5.3 Operadores de duplicação e de remoção

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

2.11.7.5.4 Operadores sexuais

Nos AGs canónicos os indivíduos podem casar com um indivíduo qualquer. Todavia, na natureza, por vezes é necessário que os indivíduos sejam de dois (ou mais) sexos distintos para a espécie sobreviver. O estabelecimento de sexos diferentes divide as espécies em dois (ou mais) grupos cooperativos. Esta bifurcação permite que cada grupo se especialize em características (subdomínios) diferentes do problema. Assim, cobre-se uma maior quantidade de comportamentos necessários à sobrevivência, do que o atingido com a competição de, apenas, um grupo.

2.11.7.5.5 Restrição no casamento

Existem vários técnicas que indicam com quem (*i.e.* qual ou com quais parceiros) um indivíduo pode casar.

Técnica reprodução de linhagem (*linebreeding*): Onde o indivíduo mais apto é o único que pode casar com os outros. Este método é bom para funções unimodais.

Técnica reprodução interna com cruzamento intermitente (*inbreeding with intermittent crossbreeding*): somente os indivíduos bastantes chegados podem casar entre si (enquanto a família existir). Esta técnica é boa para funções multimodais.

Técnica do casamento por padrão (*mating templates*): Apenas é permitido o casamento para espécies diferentes. Esta técnica é utilizada nas estratégias de evolução.

Para ilustrar como esta técnica funciona, considerem as seguintes *strings*:

<Padrão>:<Funcional>
#10#:1010
#01#:1100
#00#:0000

Nestas strings o padrão (*template*) de casamento é construído pelo alfabeto {0, 1, #}. Onde, ‘0’ casa com um ‘0’, ‘1’ casa com um ‘1’, e ‘#’ casa com ‘0’ e ‘1’.

Para determinar quais as strings que podem casar, os padrões são casados contra as substrings funcional. Podem ser implementadas várias regras de casamento:

- Casamento bidireccional
- Casamento unidireccional
- Melhor casamento parcial

No casamento bidireccional só é permitido os casamentos entre duas *strings* se a parte do padrão de uma *string* encaixa na parte funcional da outra e vice-versa. Assim, no exemplo indicado só podem casar as *strings* 1 (#01#:1100) e 2 (#10#:1010).

No casamento unidireccional, basta apenas que um padrão de uma *string* verifique a parte funcional da outra *string*.

No melhor casamento parcial, a *string* escolhida para casar é a que obtém a maior semelhança entre a sua parte funcional e o padrão da primeira *string*.

2.11.7.5.6 Mecanismo de reinserção

Depois de escolhida a subpopulação de descendentes, existem várias estratégias de substituição da antiga população.

Existe o caso onde a antiga população é reposta pela nova geração. Nesta situação é gerado um número de descendentes igual ao número de indivíduos existentes na população anterior. Esta reposição tem um aspecto negativo uma vez que a melhor *string* pode não ser seleccionada para gerar novas *strings* da geração seguinte. Assim, esta estratégia é normalmente combinada com uma estratégia elitista, de modo a que seja introduzida a melhor *string*, ou um conjunto das melhores *strings*, da geração presente na geração futura. A estratégia elitista pode levar ao domínio da população por uma *string* mas, em contrapartida, aumenta o desempenho do algoritmo.

Nos processos onde é utilizado um número pequeno de *strings* é substituída somente uma parte da população pela nova geração de *strings*. Normalmente a pior *string* é também inserida na nova população juntamente com as escolhidas para o efeito. A substituição directa das *strings* pais pelos descendentes é também utilizada neste caso. Desta forma evita-se a convergência prematura do algoritmo.

2.11.7.5.7 Operador de dominância

Nos algoritmos genéticos canónicos considera-se que os seres vivos são haploídes (*i.e.* têm um só cromossoma). Contudo na natureza existem seres vivos com mais do que um cromossoma (*i.e.* seres diploídes), podendo cada genótipo ter mais que um cromossoma (cromossomas homólogos), onde cada cromossoma contém informação relativa a mesma função. Considerando como exemplo uma estrutura diploíde, onde cada letra corresponde a um alelo:

ABCde

aBcdE

As letras maiúsculas ou minúsculas representam os valores alternativos que um alelo pode ter numa posição (*i.e.* num gene).

Na natureza cada alelo pode representar uma característica fenotípica diferente (*e.g.* a posição 1 pode representar a cor do cabelo, sendo a letra "A" utilizada para representar a cor "castanho" e a letra "a" utilizada para representar a cor "preto"). De facto existem dois alelos que representam a mesma função. Assim, deve existir um mecanismo que estabeleça qual o valor que prevalece (*i.e.* um indivíduo não pode ter cabelo preto e castanho). O mecanismo que elimina este conflito é o operador de dominância. Para cada lugar do cromossoma existe uma lista de dominância que indica qual o alelo dominante e o recessivo entre as duas alternativas. Os alelos dominantes serão representados no fenótipo. Na figura 2.9 está representado o fenótipo onde se considera que as letras maiúsculas são os alelos dominantes.

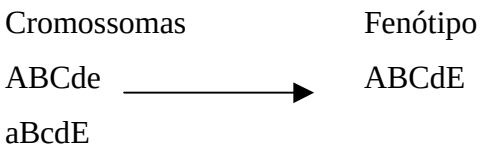


Figura 2.9 Projecção entre cromossomas homólogos e o fenótipo.

Na figura 2.9 a representação de um gene é expressa pelo alelo dominante quando heterozigoto ($Aa \rightarrow A$) ou homozigoto ($BB \rightarrow B$) e representado pelo gene recessivo quando homozigoto ($dd \rightarrow d$).

A dominância pode ser vista como uma projecção entre o genótipo e o fenótipo ou a redução do genótipo.

A representação diploíde tem raízes na ideia que especialmente em ambientes dinâmicos, AGs robustos, devem incorporar um termo longo de memória, mais comprido do que a política corrente.

2.11.8 Condição de finalização do algoritmo

Um algoritmo termina quando excede o número máximo de gerações (normalmente utilizam-se 10.000 gerações) ou quando são satisfeitas certas características de pesquisa. Existem duas características de pesquisa: uma baseada nas estruturas das *strings* (genótipo) e outra baseada no significado de uma *string* em particular (fenótipo). A condição de paragem do algoritmo, de acordo com a estrutura, mede a convergência da população através da verificação do número de alelos que convergem. A condição de finalização, de acordo com o significado de uma *string*, mede o progresso feito pelo algoritmo num número predefinido de gerações.

2.11.9 Epistase

O termo epistase (*epistasis*) é utilizado para descrever a influência de um gene na função de aptidão de uma *string*, dependendo do valor de qualquer outro gene noutro *lugar*. Um gene é dito epistático (*epistatic*) quando a sua presença elimina o efeito de outro gene noutro *lugar*. Um gene epistático também é normalmente denominado por gene inibidor devido ao seu efeito nos outros genes (hipostático).

A epistase pode ter origem de duas formas distintas: como um problema de codificação ou como problema de teoria dos AGs. Para resolver o primeiro caso, basta arranjar uma nova codificação (representação) e descodificação, onde não se verifique a epistase. No segundo caso o problema pode resolver-se utilizando operadores de cruzamento e de mutação apropriados (operadores modificados de acordo com o problema). Deste modo, pode-se representar um problema com pouca ou mesmo sem epistase.

O problema epistático pode ser evitada:

- Se se conhecer à *priori* a função objectivo de modo a codificá-la de forma apropriada.
- Utilizando um operador de inversão.
- Utilizando mGA.

2.11.10 Problema *ilusório*

Um princípio fundamental dos AGs é o de que as *strings* que incluem *arranjos*, incluídos no óptimo global, devem aumentar a sua frequência (isto verifica-se especialmente para *arranjos* pequenos e de baixa ordem). Este aumento é devido ao processo de cruzamento das *strings* e os *arranjos* óptimos devem juntar-se de modo a construir a *string* correspondente ao óptimo global. Todavia, se os *arranjos* que não estão contidos no óptimo global aumentam mais depressa do que aqueles que realmente estão, então o AG será desviado da direcção correcta, afastando-se assim do óptimo global. Por outras palavras, uma *string* muito má pode ser gerada a partir de duas strings muito boas. Esta situação é denominada por problema *ilusório*. O problema *ilusório* é um caso particular da epistase.

Estatisticamente, um *arranjo* deve aumentar a sua frequência numa população, se a correspondente função de aptidão (*i.e.* a média das funções de aptidão das *strings* que contêm esse *arranjo*) é maior do que a média das funções de aptidão de todos os *arranjos* que existem na população. Um problema é denominado por ilusório se a média da função de aptidão de qualquer *arranjo* que não está contido no óptimo global é maior do que a média das funções de aptidão dos *arranjos* que se encontram na população. Um problema é dito totalmente *ilusório* se os *arranjos*, de todas as ordens, contêm uma solução subóptima que é melhor do que a dos outros *arranjos* da população.

O seguinte exemplo ilustra um problema *ilusório*:

$$f(0,0) > f(1,1) > f(0,1) > f(1,0)$$

O *arranjo* (#, 1) não contém a solução óptima (0, 0) e os descendentes do cruzamento desta *string* com qualquer outra qualquer terão sempre um valor de aptidão inferior, pelo que, o algoritmo é levado na direcção contrária à da solução óptima.

2.11.11 Desvio genético

O desvio genético (*genetic drift*) é um fenómeno que ocorre quando o algoritmo se estabelece em óptimos diferentes no espaço fenotípico. O desvio genético é uma característica própria dos AGs quando estes não incorporarem uma função partilhada (ou mecanismo similar). Em geral, o desvio genético é conhecido na biologia, e é um fenómeno frequente em populações genéticas onde indivíduos diferentes, dentro da mesma espécie ou em espécies diferentes, não competem pelos mesmos recursos e, consequentemente, não existe competição e pressão selectiva. Nestas situações, a evolução não é guiada pela adaptação e a mera natureza estocástica da evolução dita quais os indivíduos que sobrevivem. A força que conduz à selecção não é a característica actual de um organismo mas, outrossim, a sua aptidão efectiva.

O desvio genético pode ser visto como uma evolução com uma função de aptidão larga. Nestes casos os AGs não conseguem distinguir soluções diferentes uma vez que essas têm um erro pequeno em relação as outras e não conseguem ser detectadas

2.11.12 Esquemas híbridos

Os AGs canónicos são métodos de pesquisa “cegos” pois eles exploram apenas a codificação e o valor da função objectivo, para criar descendentes plausíveis na geração seguinte. Por outro lado, a sua pequena sensibilidade face à informação especifica do problema origina, em grande parte, dos AGs a grande robustez e generalidade (um procedimento que funciona bem sem o conhecimento peculiar do problema especifico tem uma maior hipótese de transferência para outro domínio). Por outro lado, a não utilização de todo o conhecimento disponível para um problema particular, coloca os AG numa competição desigual relativamente a outros métodos que fazem uso dessa informação.

Quando se conhece informação especifica sobre o problema, pode ser vantajoso utilizar AGs híbridos. Os AGs podem ser utilizados juntamente com outros métodos de pesquisa específicos do problema, formando uma técnica híbrida que explore a perspectiva global dos AGs e a convergência do método especifico do problema.

Um exemplo desta técnica é a determinação de óptimos globais. Assim, utiliza-se um AG para determinar alguns pontos e, de seguida, adopta-se o método do gradiente para as melhores *strings*. Este processo repete-se as vezes que forem necessárias.

As técnicas híbridas são utilizadas para aumentar a velocidade da pesquisa dos AGs. O aumento de velocidade pode ser conseguida também através dos operadores de mutação e de cruzamento, com um conhecimento especifico do problema (*e.g.* operador PMX para o problema TSP).

2.11.13 Porque funcionam os algoritmos genéticos

Os fundamentos teóricos dos AGs incidem na representação binária das *strings* e a notação de *arranjo*. Um *arranjo* permite explorar as semelhança entre os cromossomas. Um *arranjo* (*schemata*) é construído pela introdução do símbolo # (este caracter pode substituir qualquer outro símbolo existente) no alfabeto dos genes. Um *arranjo* representa qualquer *string* (um hiperplano, ou um subconjunto do espaço de pesquisa), que seja igual em todas as posições, excepto naquelas onde existe o símbolo #. Por exemplo, o arranjo {#01} representa as *strings* {001, 101} e o *arranjo* {0##} representa as *strings* {000, 001, 010, 011}. Um *arranjo* representa 2^k *strings*, onde k é o número de símbolos #. Por outro lado, uma *string* com comprimento l pode ser representada por 2^l *arranjos*. Obviamente *arranjos* diferentes têm propriedades diferentes.

A ordem de um *arranjo* (o (S)é o número de símbolos existentes na *string* diferentes do caracter #.

O comprimento de um *arranjo* (δ (*S*)é distância entre o primeiro e o último símbolos fixos que ocorrem na *string*.

Para as *strings* {101##10#, 01##1001} a ordem e o comprimento do *arranjo* são, respectivamente, {(o = 5, δ = 7 – 1 = 6), (o = 6, δ = 8 – 1 = 7)}.

A ordem e o comprimento têm interesse para calcular as probabilidades de continuidade de um *arranjo* após a mutação e o cruzamento.

Seja $\xi(S, t)$ o número de *strings* que o *arranjo S* representa. A função de aptidão do *arranjo S*, f(*S*, *t*), isto é, a média do valor de aptidão de todas as *strings* que o *arranjo* representa é dado por:

$$f(S, t) = \sum_{j=1}^{p} \frac{f(S_j, t)}{p}$$

Durante o passo de selecção é criada uma população intermédia. Assim, cada *string* é copiada de acordo com a sua função de aptidão.

$$p_i = \frac{f(S_i)}{F(t)}$$

onde F(*t*) é a soma das funções de aptidão de todas *strings* da população na geração *t*.

Após a selecção espera-se que o número de *strings* representadas pelo *arranjo* seja

$$\xi(S, t + 1) = \mu \times \frac{f(S, t)}{F(t)}$$

se

$$F_m(t) = \frac{F(t)}{\mu}$$

onde μ é o número de *strings* na população, então:

$$\xi(S, t + 1) = \xi(S, t) \times \frac{f(S,t)}{F_m(t)}$$

Por outras palavras, o número de *strings* que o *arranjo* representa na população cresce de acordo com a taxa de aptidão do *arranjo* e com a média da função de aptidão da população. Assim, se o número de *strings* que um *arranjo* representa tem um valor de aptidão superior (inferior) à média dos valores de aptidão da população, o número de *strings* representadas pelo arranjo aumenta (diminui) na geração seguinte.

Se a média do valor de aptidão do *arranjo* se mantém superior à média ε, então:

$$f(S, t) = F(t) + \varepsilon \times F(t)$$

$$\xi(S, t) = \xi(S,0) \times (1 + \varepsilon)^t$$

Assim, a representação do *arranjo* aumento exponencialmente de geração para geração.

De seguida é ilustrado o efeito que a recombinação introduz no número de *arranjos* que são esperados na geração seguinte.

O comprimento do *arranjo* tem um papel bastante importante para a sua sobrevivência após o cruzamento uma vez que *arranjos* com grande comprimento têm uma maior probabilidade de serem destruídos.

Se o ponto de cruzamento é escolhido uniformemente então a probabilidade de destruição do arranjo é

$$p_d = \frac{\delta(S)}{l-1}$$

Consequentemente, a probabilidade do arranjo sobreviver é dada por:

$$p_s = 1 - \frac{\delta(S)}{l-1}$$

Sendo a probabilidade de cruzamento p_c então vem:

$$p_s(S) = 1 - p_c \times \frac{\delta(S)}{l-1}$$

Mesmo que o ponto de cruzamento seja seleccionado entre posições fixas existe a possibilidade do arranjo sobreviver. Isto acontece quando a segunda *string* tem os mesmos valores que a primeira *string* para aquelas posições fixas. Deste modo:

$$p_s(S) \geq 1 - p_c \times \frac{\delta(S)}{l-1}$$

Considerando a probabilidade de ocorrer uma mutação p_m, a probabilidade de alterar um simples símbolo é p_m e a probabilidade do símbolo sobreviver é $1 - p_m$. Assim, a probabilidade de um *arranjo*, de *k* símbolos fixos ($o(S)$), sobreviver é

$$p_s(S) = (1 - p_m)^{o(S)}$$

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

$$p_s(S) \approx 1 - p_m \times o(S)$$

Combinando a selecção, o cruzamento e a mutação, o crescimento de um *arranjo* pode ser expresso pela seguinte fórmula:

$$\xi(S, t+1) \geq \xi(S, t) \times \frac{\mathrm{f}(S,t)}{\mathrm{F_m}(t)} \times [1 - \mathrm{p_c} \times \frac{\delta(S)}{l-1} - o\ (S) \times \mathrm{p_m}]$$

Esta expressão é válida para funções de aptidão sempre positivas.

O crescimento exponencial das *strings* que o *arranjo* representa deve-se à selecção. O efeito destrutivo do cruzamento e da mutação num *arranjo* não é significativo para *arranjos* de comprimento pequeno e de baixa ordem.

O teorema do *esquema* mostra que *arranjos* pequenos, de baixa ordem e com valor de aptidão superior à média, têm um aumento exponencial no número de *strings* que representam, nas gerações seguintes.

A hipótese de construção de blocos é enunciada da seguinte forma: um algoritmo pesquisa com desempenho próximo do óptimo se utilizar *arranjos* de comprimento pequeno, ordem baixa e com alto desempenho.

2.11.14 Convergência prematura

A convergência prematura é um problema comum nos AGs. Se esta ocorre muito rápido, perde-se muita informação genética. Existem algumas estratégias para evitar este problema. Como por exemplo:

- Prevenção de incesto
- Cruzamento uniforme
- Detecção de *strings* duplicadas

2.11.15 Algoritmos genéticos com o número de indivíduos da população variável

Como o próprio nome indica, a população deste algoritmo (AGsPV) pode variar ao longo das gerações. A escolha do número de elementos da população pode ser crítica para o algoritmo. Populações muito grandes consomem bastante tempo de cálculo até ser alcançado um melhoramento substancial. Em contrapartida, populações pequenas podem levar à convergência prematura do algoritmo.

O algoritmo AGsPV utiliza a noção de *idade* para indicar o número de gerações que uma *string* pode permanecer na população. A *idade* de um elemento é atribuída de acordo com a sua aptidão e a selecção é feita por esse mecanismo.

O algoritmo deste tipo de AGs é mostrado a seguir:

```
Procedimento AGsPV
t = 0
Inicializa P(t)
Avalia P(t)
Repetir enquanto condição de fim não é verificada
  t = t + 1
  Incrementa idade para cada string
  Recombinação P(t)
  Retira strings com idade superior ao seu tempo de vida
Fim de repetir
Retornar
```

Algoritmo 2.4 Algoritmo AgsPV.

Existem vários métodos para calcular o *tempo de vida* de uma *string*. Este cálculo deve beneficiar os indivíduos com aptidão acima da média e regular o tamanho da popu-

lação de acordo com o estado da pesquisa. De seguida são apresentados três tipos de cálculo:

- *tempo de vida* proporcional

$$\text{mínimo } (tv_{min} + \eta \times \frac{f_i}{F_m}; tv_{max})$$

- *tempo de vida* linear

$$tv_{min} + 2 \times \eta \times \frac{f_i - |F_{min}|}{|F_{max}| - |F_{min}|}$$

- *tempo de vida* bilinear

$$\begin{cases} tv_{min} + \eta \times \dfrac{f_i - F_{min}}{F_m - F_{min}}, & F_m \geq f_i \\ \dfrac{tv_{min} + tv_{max}}{2} + 2 \times \eta \times \dfrac{f_i - |F_{min}|}{|F_{max}| - |F_{min}|}, & F_m \leq f_i \end{cases}$$

onde:

F_m é o valor médio da função de aptidão da população;

F_{max} é o valor de aptidão máximo existente na população;

F_{min} é o valor de aptidão mínimo existente na população;

tv_{max} é o *tempo de vida* máximo;

tv_{min} é o *tempo de vida* mínimo;

$\eta = (tv_{max} - tv_{min}) / 2$.

O cálculo proporcional inspira-se no método de selecção da roleta redonda e tem como desvantagem não utilizar como referência o tempo de vida da melhor *string*. Na estratégia linear o valor é calculado de acordo com os valores de aptidão da *string* em questão e da melhor *string* da população. Por último, a estratégia bilinear estabelece um compromisso entre as duas estratégias anteriores.

2.11.16 Algoritmos genéticos desordenados

Os AGs *desordenados* (AGDs) (*messy genetic algorithms*) diferem dos AGs clássicos na representação, nos operadores, na selecção e fases do processo de selecção. As *strings* são de tamanho variável podendo ser especificadas tanto por defeito como por excesso com respeito ao problema em questão. Os operadores utilizados são menos rígidos do que os adoptados nos AGs clássicos. Além disso, a representação da informação deve ser robusta, flexível e adaptativa. Nesta perspectiva, a representação é implementada no algoritmo através da introdução de códigos e operadores desordenados (*messy*) para as expressões do Gene.

Cada alelo da *string* é rotulada com um código. Por seu lado as *strings* tem um comprimento variável, podendo ter tanto alelos redundantes como alelos em contradição.

Por exemplo, para a *string*:

$$s_1 = ((7,1)(1,0)(2,0)(1,0)(9,1)(2,1))$$

onde cada par de valores indica a posição do alelos e o valor respectivo. A *string* s_1 especifica o valor dos *bits* 1, 2, 7 e 9. O *bit* 1 e 2 encontram-se repetidos, estando o *bit* 2 em contradição com ele próprio.

Para manusear estas *strings* devem existir funções para lidar com problemas de sobre e sub-informação. A sobre-informação pode ser resolvida pelo maior número de valores idênticos ou através da precedência. A sub-informação é mais difícil de tratar.

Os operadores de cruzamento utilizados são a concatenação (*splice*) e o corte (*cut*). O operador concatenação junta duas *strings* previamente escolhidas. O operador corte divide uma *string* em duas num local determinado aleatoriamente. O operador de mutação é idêntico ao utilizado nos AGs canónicos.

A selecção utilizada nos AGDs é idêntica à selecção por torneio. O processo de evolução está dividido em duas partes: a selecção das *strings* e o uso dos operadores genéticos.

Os AGDs tem um bom desempenho para problemas difíceis (*e.g*. problemas ilusórios) e o tempo computacional cresce de forma polinomial de acordo com as variáveis de decisão.

2.11.17 Algoritmos genéticos cíclicos

Nos AGs cíclicos [5] (AGCs) o cromossoma é constituído por 3 partes: a parte inicial, a parte interactiva e a parte final. A parte inicial contém as tarefas requeridas para inicializar a secção interactiva que é necessária no ciclo continuo. Cada gene (representa uma tarefa) pode ser uma primitiva, subprograma ou ciclo. Existem alguns genes que podem ter como tarefa a coordenação ou inibição de certos comportamentos.

Os AGCs podem ter comprimento fixo ou variável. De qualquer forma, o sistema deve ser capaz de atribuir o número apropriado de tarefas em cada fase e ser suficientemente flexível para permitir que o AGC forme um ciclo completo na secção interactiva.

Quando o comprimento do cromossoma é fixo uma tarefa (gene) pode ser repetida, e este número de repetições é codificado no gene. Desta forma, os cromossomas de tamanho fixo podem guardar as características desejadas dos cromossomas de comprimento variável mantendo fixo o controlo do treino.

Quando o cromossoma tem um tamanho variável, os operadores genéticos (normalmente o cruzamento) devem fornecer um meio para a variação do cromossoma. São usu-

almente requeridos alguns meios de controlo devido a poderem existir mudanças drásticas no comprimento do cromossoma impedindo-o de convergir.

Existem dois tipos de operador de cruzamento:

- Cruzamento para AGCs com tamanho fixo. É usado o método de ponto simples para as partes inicial e final, e de ponto duplo para a parte iterativa.
- Cruzamento para AGCs com a parte iterativa variável (gene-a-gene). O cruzamento é feito em entre os genes correspondentes. No caso em que em que os genes são representados por listas, os pontos de cruzamento podem ocorrer entre os membros individuais da lista ou com os *bits* dos números específicos da lista.

Por exemplo, considere-se os cromossomas iniciais (um gene é formado por quatro *bits*):

(0000 0000 ((0000 0001) (0000 0011))
(1111 1111 ((1111 0111) (1101 0100))

O cromossoma após cruzamento gene-a-gene vem:

(0011 0001 ((0000 0011) (0001 0100))

O operador de mutação pode ser de dois tipos:

- Reposição aleatória do gene. O valor do gene é substituído por outro valor aleatório.
- Mutação do Gene. Um *bit* dentro do gene é mudado para o seu valor complementar.

O operador de avaliação gene-a-gene consiste em escolher aleatoriamente um ou dois indivíduos da população e avaliar nestes um gene de cada vez. Este operador remove os genes que são inferiores a um determinado limiar, reduz o número de repetições de um gene (*i.e.* quando inicialmente este tem bom desempenho e após algumas repetições o desempenho diminui) e coloca os genes com zero repetições no fim da zona iterativa.

Os AGCs são utilizados para resolver problemas onde existam acontecimentos repetitivos que requerem acções sequenciais contínuas.

2.11.18 Manter a diversidade na evolução artificial

2.11.18.1 Introdução

Nesta secção são apresentados alguns métodos para manter a diversidade da população nos AGs.

2.11.18.2 Modelo AGs regime permanente (s*teady state*)

O modelo normalmente utilizado nos AGs e na programação genética pode ser considerado um processo periódico, onde a nova população é criada de uma só vez. Em contraste com a maioria das populações reais onde os nascimentos e mortes ocorrem continuamente e em intervalos aleatórios. Além disso, uma vez nascidos os filhos da geração seguinte, a geração antiga é eliminada. Assim, é possível, e bastante provável, que os indivíduos mais aptos não sobrevivem na nova geração, apesar de muito do seu material genético ser retido.

A ideia por detrás do modelo AGs regime permanente (AGsRP) é executar a selecção individual e os casamentos como é descrito no parágrafo anterior, mas, com a modificação de dois novos descendentes substituírem os dois indivíduos menos aptos existentes da população.

O método AGsRP confere imortalidade ao melhor indivíduo da população pois este nunca é seleccionado para substituição até nascer um indivíduo melhor. Numa perspectiva mais geral este modelo permite aos genótipos variarem o tempo de vida com base na sua aptidão, relativamente a outros membros da população. Consequentemente, o método AGsRP é um bom método para manter a diversidade na população.

2.11.18.3 Esquema de agrupamento

Este esquema de agrupamento (*crowding*) é usado entre gerações, onde os novos indivíduos vão substituir aqueles que genotipicamente são similares com estes (em alternativa à substituição dos indivíduos menos aptos). Para prevenir um número impraticável de comparações quando um indivíduo é criado, indivíduos recém-nascidos vão ser comparados apenas a um certo número (factor de agrupamento – FA) de indivíduos. Este esquema foi desenvolvido para funções multinodais, isto é, para funções com vários picos na função de aptidão.

2.11.18.4 Esquema de partilha

O esquema de partilha (*sharing*) baseia-se no principio de que um ambiente contém recursos limitados e que indivíduos fenotipicamente similares devem partilhar esses recursos (uma ideia razoável do ponto de vista da biológico). Indivíduos similares sofrem penalidades na sua função de aptidão dependendo da sua similaridade com outros indivíduos na população.

Este esquema foi desenvolvido para funções multinodais, não colocando ênfase numa solução única.

2.11.18.5 Isolamento por distância

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

- casamento espacial (*spacial mating*);
- *stepping stone* ou modelo das ilhas (*island model*).

O modelo casamento espacial coloca os indivíduos numa grelha redonda e só permite casamentos entre os vizinhos. Este método implica a criação dinâmica de *demes* com tamanho variável de acordo com a sua aptidão relativamente aos *demes* vizinhos.

O modelo das ilhas divide a população em vários *demes*. Cada um evolui com a sua própria velocidade e direcção existindo uma certa migração entre as ilhas.

2.11.18.6 Prevenção de incesto

A prevenção de clones ajuda a evitar a convergência prematura na evolução artificial. Eshelman [9] sugeriu a prevenção de incesto, nomeadamente onde apenas pais geneticamente diferente podem acasalar, a fim de prevenir a formação de clones. No entanto, existe o problema de saber até que ponto dois pais são diferentes. A estratégia de Eshelman foi a de calcular a distância de Hamming entre os pais escolhidos. Se a distância for superior a um dado valor fixo então os pais podem casar. Todavia, à medida que o algoritmo converge os pais ficam mais semelhantes entre si e o número de casamentos rejeitados aumenta. À medida que o número de casamentos rejeitados é aumentada a distância pode ser relaxada ligeiramente. Este método reduz significativamente a criação de *clones* com baixo custo.

2.11.19 Algoritmos genéticos não binários

2.11.19.1 Introdução

Esta secção aborda os AGs com representação não binária. Assim, é explicado a sua representação e o funcionamento de alguns operadores de cruzamento e mutação.

2.11.19.2 Representação

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

2.11.19.3 Operador de cruzamento

Para o operador de cruzamento podem ser utilizados os algoritmos seguintes:

- Cruzamento simples

Definido de modo semelhante ao operador utilizado nos AGs canónicos mas com a restrição: para um dado cromossoma X, o ponto de cruzamento deve estar entre dois parâmetros. Assim, sendo:

$$X_t = <x_1, x_2,\ldots, x_q>$$
$$Y_t = <y_1, y_2,\ldots, y_q>$$

se o cruzamento é feito entres as posições $k - 1$ e k ($1 \leq k \leq q$), os descendentes são:

$$X_{t+1} = <x_1, x_2,\ldots,x_{k-1}, y_k,\ldots,y_q>$$
$$Y_{t+1} = <y_1, y_2,\ldots,y_{k-1}, x_k,\ldots,x_q>$$

- Cruzamento aritmético

O operador é definido com uma combinação linear de dois vectores. Assim, sendo X_n e Y_n os vectores progenitores, os vectores resultantes do cruzamento são:

$$X_{t+1} = a \times Y_t + (1 - a) \times X_t$$
$$Y_{t+1} = a \times X_t + (1 - a) \times Y_t$$

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

2.11.19.4 Operador de mutação

Para o operador de mutação podem ser utilizados os algoritmos seguintes:

- Mutação uniforme

Este operador é semelhante ao utilizado nos AGs. Para o cromossoma $S_t = <s_1, s_2,\ldots,s_n>$, onde cada parâmetro tem a mesma probabilidade de ser escolhido para sofrer a mutação, se o elemento s_k é seleccionado para a mutação, então o cromossoma após a operação fica $S_t = <s_1,\ldots,s'_k,\ldots,s_n>$ onde s'_k tem um valor aleatório do domínio compreendido entre s_{kmin} e s_{kmax}.

- Mutação não uniforme

Este operador é utilizado para pesquisas locais do sistema. Para o cromossoma $S_t = <s_1, s_2,\ldots,s_n>$, se o elemento s_k é seleccionada para a mutação, então o cromossoma fica $S_t = <s_1,\ldots,s'_k,\ldots,s_n>$ onde

$$s'_k = \begin{cases} s_k + \Delta(t, s_{kmax} - s_k) & \text{se o número aleatório é 0} \\ s_k - \Delta(t, s_k - s_{kmin}) & \text{se o número aleatório é 1} \end{cases}$$

A função $\Delta(t, y)$ tem contradomínio $[0; y]$ e é tal que a probalidade do seu valor estar próximo de zero aumenta com *t*. Desta forma, inicialmente o operador pesquisa o espaço uniformemente (quando *t* é pequeno) e à medida que *t* aumenta a pesquisa torna-se, pro-

gressivamente, local. Assim, para grandes valores de *t* o valor gerado é muito próximo de *y*. Um exemplo para a função $\Delta(t, y)$ é

$$\Delta(t, y) = y \times (1 - r^{(1-\frac{t}{T})^b})$$

onde:

r é um número aleatório entre 0 e 1;

T é o número total de gerações;

b é um parâmetro do sistema de acordo com o grau de não uniformidade.

A mutação não uniforme tem um desempenho, em termos de tempo, superior à mutação uniforme.

- Mutação proposta por Srinivas e Patnaisk [16]

Este tipo de mutação utiliza probabilidades adaptativas e juntamente com o operador de cruzamento (também proposto pelos mesmos autores) mantêm a diversidade na população e a capacidade de convergência do algoritmo. As probabilidades destes operadores variam de acordo com os valores de aptidão das soluções: as soluções boas são protegidas enquanto que as soluções más são destruídas. Os operadores de cruzamento e mutação são os seguintes:

$$p_c = \begin{cases} k_1 \times \dfrac{f_{max} - f'}{f_{max} - f_m} & \text{se } f' \leq f_m \\ k_3 & \text{se } f' > f_m \end{cases}$$

$$p_m = \begin{cases} k_1 \times \dfrac{f_{max} - f'}{f_{max} - f_m} & \text{se } f' \leq f_m \\ k_4 & \text{se } f' > f_m \end{cases}$$

onde:

as constante k_1, k_2, k_3 e k_4 têm valor compreendido entre 0 e 1;

a variável f_{max} é o valor máximo do valor de aptidão da população;

a variável f_m é o valor médio dos valores de aptidão da população;

a variável f' é o maior valor de aptidão das duas *strings* seleccionadas para o cruzamento.

O valor de $f_{max} - f_m$ é essencial na fórmula anterior e é também importante na medida da convergência do algoritmo. As probabilidades p_c e p_m são nulas para as soluções com o valor de aptidão máxima. Para soluções com a função de aptidão igual à média das funções de aptidão da população (*i.e.* $f = f_m$) os valores das probabilidades são $p_c = k_3$ e $p_m = k_4$.

2.11.19.5 Outros operadores de cruzamento

Para além dos operadores de cruzamento referidos em 2.11.19.3 podem referir-se ainda os seguintes:

- Média – o descendente é criado a partir da média aritmética dos pais.
- Geométrico – o descendente é criado a partir da média geométrica (*i.e.* da raiz quadrada do produto) dos pais.
- Extensão – o descendente é criado através da diferença dos valores dos dois pais. Esta diferença é posteriormente adicionada ao valor mais alto ou subtraída ao valor mais baixo.

2.11.19.6 Outros operadores de mutação

Para além dos operadores de mutação referidos em 2.11.19.4 podem referir-se ainda os seguintes:

- Reposição aleatória – substitui o símbolo pelo valor de um dos pais, escolhido aleatoriamente.
- *Creep* – adiciona ou subtrai um pequeno valor gerado aleatoriamente.
- *Creep* geométrico – o valor é multiplicado por um número, próximo da unidade, gerado aleatoriamente.

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

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

2.12 Estratégias de evolução

2.12.1 Introdução

Os AGs são úteis para resolver problemas de optimização de difícil formulação. Deste modo, não é de estranhar que as EEs, incorporando conhecimento específico do problema nos cromossomas, apresentem um desempenho superior aos AGs.

As restrições impostas por um problema nas EEs são satisfeitas através da escolha "da melhor" representação para o cromossoma juntamente com operadores genéticos correspondentes. Um operador deve passar certas características do progenitor para o descendente, pelo que a estrutura de representação é muito importante na elaboração dos operadores genéticos. Consequentemente, estas duas componentes influenciam-se uma à outra.

Nesta secção é descrita a representação e o funcionamento dos operadores nas EEs e é indicado um meio de melhorar a convergência do algoritmo. Por último, são comparados os AGs e as EEs.

2.12.2 Representação

Nas estratégias de evolução a representação adopta directamente valores reais (através de números em virgula flutuante) quando o problema de optimização lida com parâmetros contínuos (note-se que as EEs são também utilizadas para resolver problemas discretos). Por outro lado, os problemas do mundo real têm espaços de pesquisa bastante complexos que não podem ser projectados nos algoritmos canónicos. Devido a este facto surgiram muitas variantes com valores inteiros, com "misturas" (*mixing-integer*) e com a optimização de estruturas.

Quando as EEs foram inicialmente desenvolvidas a população era constituída apenas por um indivíduo (*i.e.* (1 + 1) – população com dois membros) e era munido apenas do operador de mutação.

Para EEs multimembros (μ , 1), ou seja, com o número de indivíduos da população maior do que um membro, foram introduzidos operadores de recombinação com mais do que um pai onde todos os membros da população têm a mesma probabilidade de acasalamento. Sendo, em cada geração produzida apenas uma *string*.

As estratégias (1 + 1) e (μ , 1) deram origem as estratégias (μ , λ) e (μ + λ). Estas estratégias serão explicadas numa subsecção seguinte.

A representação de um indivíduo é constituída por um par de valores reais (ponto no espaço de pesquisa, desvio padrão):

$$X = (x, \sigma)$$

2.12.3 Operadores genéticos

2.12.3.1 Introdução

Existe uma grande variedade de operadores genéticos mas todos devem obedecer às propriedades matemáticas da representação escolhida. De seguida são apresentados quatro tipos principais.

2.12.3.2 Operador de selecção

A estratégia de evolução (μ, λ) usa um esquema de selecção determinista. A notação (μ, λ) indica que μ pais criam λ $(\lambda > \mu)$ descendentes através dos operadores de recombinação e mutação. De seguida, os melhores μ descendentes são escolhidos, deterministicamente, para substituírem os pais (neste caso Q = 0 no algoritmo 2.1). Assim, o tempo de vida de um indivíduo é exactamente uma geração. Este mecanismo permite que o melhor membro da população, na geração $t + 1$, possa ter um desempenho inferior ao do melhor indivíduo da geração t. Como se pode ver este método não é elitista. Consequentemente, a estratégia permite deteriorações temporárias que podem ajudar a abandonar um óptimo local e deslocar-se para um óptimo superior. Nesta perspectiva, a estratégia tem melhor desempenho nos problemas onde o valor óptimo varia com o tempo e para problemas com ruído.

A estratégia $(\mu + \lambda)$ selecciona os μ sobreviventes do conjunto constituído pelos pais e seus descendentes. Desde modo é garantido um curso monótono na evolução (neste caso Q = P(n)) no algoritmo 2.1).

Ambos os esquemas podem ser interpretados como consequências da estratégia geral (μ, k, λ), onde k $(1 \leq k \leq \infty)$ é o número máximo de vidas de um indivíduo. Para $k = 1$, o método de selecção leva ao método (μ, λ) enquanto que para $k = \infty$ o método leva à estratégia $(\mu + \lambda)$.

Quando a população é constituída apenas por uma *string*, esta só é substituída se o valor de aptidão do descendente for superior (*i.e.* $f(x_{t+1}) > f(x_t)$).

2.12.3.3 Operador de recombinação

A recombinação das EEs é incorporada como primeiro operador no ciclo principal (ver algoritmo 2.1) e gera uma população intermédia de λ indivíduos a partir de um conjunto de μ aplicações feitas à população de pais. Por cada aplicação é criado um indivíduo a partir de ρ $(1 \leq \rho \leq \mu)$ indivíduos. O valor de ρ utilizado normalmente é 2 ou μ (recombinação global). Nas EEs o tipo de recombinação das variáveis objecto e os parâmetros estratégicos diferem frequentemente uns dos outros. Exemplos típicos são a recombinação discreta (escolhas aleatórias de uma variável dos pais comparável com o cruzamento uniforme nos AGs) e a recombinação intermédia (frequentemente médias aritméticas, mas é possível outras variantes como o cruzamento geométrico). O operador de recombinação nas EEs é utilizado na criação de todos os descendentes da população.

A escolha do operador de recombinação depende da topologia da função objectivo, da dimensão da função objectivo e dos parâmetros que constituem um indivíduo.

Este operador é utilizado para populações onde o número de *strings* é superior a um e podem apontar-se dois tipos principais.

- Operador de cruzamento uniforme.

Para EEs funciona do seguinte modo: retiram-se aleatoriamente duas *strings* da população

$$(X, V) = (x_1,\ldots,x_n, v_1,...,v_n)$$
$$(Y, S) = (y_1,\ldots,y_n, s_1,...,s_n)$$

A *string* (descendente) resultante é

$$(Z,\sigma) = (z_1,\ldots,z_n,\sigma_1,...,\sigma_n)$$

onde:

z_i é um valor (x_i ou y_i) escolhido aleatoriamente nas *strings* X ou Y;

σ_i é um valor (v_i ou s_i) escolhido aleatoriamente nas *strings* V ou S.

- Operador de cruzamento utilizando aprendizagem.

Os operadores utilizados nas estratégias (μ, λ) e $(\mu + \lambda)$ utilizam dois níveis de aprendizagem. O parâmetro de controlo σ deixa de ser constante, e ao invés de ser mudado através de regras deterministas (como a regra 1/5 – ver secção 2.12.4) passa a ser incorporado nas estruturas dos indivíduos e, assim, submetido ao processo evolutivo.

Considerem-se as duas *string* (escolhidas aleatoriamente) do exemplo anterior.

O cruzamento <u>discreto</u> apresenta o seguinte resultado:

$$(Z,\sigma) = (z_1,\ldots,z_n,\sigma_1,\ldots,\sigma_n)$$

onde:

z_i é um valor (x_i ou y_i) escolhido aleatoriamente nas *strings* X ou Y;

σ_i é um valor (v_i ou s_i) escolhido aleatoriamente nas *strings* V ou S.

O cruzamento <u>intermédio</u> apresenta o seguinte resultado:

$$(Z, \sigma) = (0{,}5 \times (x_1 + y_1)),\ldots, 0{,}5 \times (x_n + y_n), 0{,}5 \times (v_1+s_1),\ldots, 0{,}5 \times (v_n + s_n))$$

2.12.3.4 Operador de mutação

Nas EEs a mutação é executada independentemente em cada indivíduo, adicionando um valor obtido a partir de uma distribuição aleatória normal.

- Mutação normal.

A função de distribuição tem média nula e desvio padrão σ, resultando:

$$x'_t = x_t + N_i(0,\sigma)$$

onde $N_i(0,\sigma)$ é um vector de números aleatórios Gauseanos de média zero e desvio padrão σ.

- Mutação utilizando aprendizagem.

O desvio padrão da função N_i é escolhido aleatoriamente para cada *string* e para cada geração. Aplicando o operador mutação ao descendente (x, σ), a *string* resultante é (x', σ'), onde

$$\sigma' = \sigma \times e^{N(0, \Delta\sigma)}$$
$$x' = x + N(0, \sigma')$$

sendo $\Delta\sigma$ um parâmetro.

- Mutação introduzida por Schwfel.

Este operador é idêntico ao anterior mas apresenta um melhoramento na convergência. Assim, uma *string* é representada da forma:

$$(x, \sigma, \theta)$$

A mutação da *string* anterior é dada por (x', σ', θ') com:

$$\sigma' = \sigma \times e^{N(0, \Delta\sigma)}$$
$$\theta' = \theta + N(0, \Delta\theta)$$

$$x' = x + \mathrm{C} \times (0, \sigma', \theta')$$

sendo $\Delta\sigma$ e $\Delta\theta$ parâmetros.

2.12.4 Melhoramento da convergência

Com o fim de melhorar a convergência Rechenberg [16] propôs a regra 1/5. Esta regra é aplicada após *k* gerações (sendo *k* um parâmetro do método). Assim, o desvio padrão σ é dado pela seguinte expressão:

$$\sigma^{t+1} = \begin{cases} c_d \times \sigma^t, & \text{se } \varphi(k) < \frac{1}{5} \\ c_i \times \sigma^t, & \text{se } \varphi(k) > \frac{1}{5} \\ \sigma^t, & \text{se } \varphi(k) = \frac{1}{5} \end{cases}$$

sendo $\varphi(k)$ a taxa de sucesso do operador de mutação durante as *k* gerações anteriores e as constantes $c_i > 1$ e $c_d < 1$ regulam o aumento e a diminuição da variação da mutação.

2.12.5 Comparação entre as estratégias de evolução e os algoritmos genéticos

A principal diferença entre as EEs e os AGs incide no seu domínio pois as EEs foram desenvolvidas como método de optimização numérica. Os EEs adoptam um procedimento especial de melhoria (hill-climbing) auto adaptativo com um passo σ e ângulo de inclinação θ. Por outro lado, os AGs têm sido formulados como técnicas de pesquisa adaptativas, que aumentam exponencialmente o número de experiências para *arranjos* com um valor de aptidão superior à média.

Apenas recentemente as EEs têm sido aplicadas a problemas de optimização discretos. Os AGs são aplicados numa variedade de domínios, e uma optimização de parâmetros (real) é só um campo das suas aplicações.

A maior semelhança entre os AGs e as EEs consiste em ambos manterem uma população de soluções potenciais e usarem o principio da selecção do indivíduo mais apto.

Assim, quatro grandes diferenças entre as EEs e os AGs clássicos consistem:

- No modo como as *strings* são representadas. As EEs utilizam vectores de números em virgula flutuante enquanto os AGs operam sobre vectores binários.

- Na forma como as *strings* são seleccionadas. Nas EEs são escolhidos as melhores μ *strings* do conjunto formado pelos pais e seus descendentes, enquanto que nos AGs o processo é aleatório, seleccionando-se μ *strings* (com possíveis repetições), sendo a selecção proporcional à aptidão das *strings*.

- Na ordem relativa do processo de selecção e recombinação. Nas EEs o processo de selecção segue a aplicação dos operadores de recombinação, enquanto que nos AGs estes passo ocorrem em ordem inversa. Nas EEs um descendente é o resultado do cruzamento de dois pais e de uma mutação posterior. Nos AGs é seleccionada uma população intermédia sendo o cruzamento e a mutação aplicados a algumas *strings*. Nos AGs as probabilidades dos operadores mantêm-se constantes durante o processo de evolução, enquanto que nas EEs os parâmetros σ e θ variam (auto adaptam-se) durante todo o processo.

- No modo como são tratadas as restrições. As EEs consideram um conjunto de $q \geq 0$ inequações:

$$g_i \geq 0,\ i = 1,\ldots,\ q,$$

como parte do problema de optimização. Se, durante alguma iteração, um descendente não satisfaz todas as restrições, então este é desqualificado, isto é, não é colocado na nova população. Se a taxa desses descendentes é alta, então as EEs ajustam

esses parâmetros de controlo, através da diminuição da componente do vector σ . Nos AGs o tratamento das restrições é feito através de penalidades.

2.13 Programação evolutiva

2.13.1 Introdução

A PE é usada para resolver problemas de optimização numérica.

Nesta secção é descrita a PE. Nessa perspectiva, é estabelecida uma comparação entre a PE e as EEs e é descrita a representação e o funcionamento dos operadores da PE.

2.13.2 Comparação entre a programação evolutiva e as estratégias de evolução

A PE e a EE têm muitas semelhanças, mas os pontos fundamentais que distinguem a PE são os seguintes:

- A PE não utiliza operadores de recombinação. Assim, o operador de mutação é o operador fundamental na PE.
- A PE usa uma selecção estocástica (selecção por torneio), em contraste com a EE que selecciona os melhores μ indivíduos para a geração seguinte.
- Na PE os valores de aptidão são obtidos através da função objectivo projectando-os e possivelmente impondo-lhes uma alteração aleatória.
- Na EE, para o operador de mutação de uma *string*, o desvio padrão é calculado como a raiz quadrada da transformação linear dos seus próprios valores de aptidão (secção 2.13.4.3).

2.13.3 Representação

Na programação evolutiva a representação utiliza directamente valores reais (representação em virgula flutuante) quando o problema de optimização lida com parâmetros contínuos. Assim, uma solução é formada por vectores de valores reais (cromossomas), todos do mesmo comprimento. Cada elemento é inicializado de modo a pertencer ao domínio e os operadores são criados de forma a preservar esta restrição.

A precisão do algoritmo depende da máquina a utilizar, mas é superior à da representação binária. Além disso, a utilização de valores reais tem uma capacidade de representação de domínios superior à da representação binária.

2.13.4 Operadores genéticos

2.13.4.1 Introdução

Os operadores utilizados na PE e nos AGs diferem entre si. Assim, para a PE podem mencionar-se os operadores de selecção e de mutação.

A PE não tem qualquer tipo de operador de recombinação. A evolução das soluções incide apenas no poder do operador de mutação.

2.13.4.2 Operador de selecção

Para este operador existe uma diferença mínima nos casos das EEs. A diferença na PE consiste na escolha de uma variante estocástica de selecção ($\mu + \lambda$), onde cada solução de indivíduos, pais e filhos, é avaliada contra q ($q > 1$, tipicamente, $q \leq 10$) outras soluções escolhidas aleatoriamente da união de indivíduos, pais e filhos, ($\mu \cup \lambda$) (*i.e.* Q = P(t) no algoritmo 2.1). Por cada comparação é afectado um ganhador se o valor de aptidão é melhor ou igual à do seu oponente. Os μ indivíduos com mais vitórias são retidos para formarem a geração seguinte. Este método de selecção é uma versão estocástica

de selecção $(\mu + \lambda)$ que vai ficando cada vez mais determinista, à medida que o número q de competidores aumenta.

2.13.4.3 Operador de mutação

Originalmente, na programação evolutiva, o operador de mutação foi implementado como uma ou mais alterações aleatórias na descrição de máquinas de estado finitas. Estas alterações consistiam em: mudar um símbolo de saída, mudar um estado de transição, adicionar um estado, apagar um estado ou mudar o estado inicial. A mutação era normalmente executada de acordo com uma função de probabilidade uniforme e o número de mutações para um descendente era fixo ou escolhido de acordo com uma determinada distribuição de probabilidade. Hoje em dia a mutação que é mais utilizada, na aplicação de representações de valores reais, é muito semelhante à utilizada nas estratégias de evolução.

Na PE a mutação é feita da seguinte maneira:

Considere-se uma solução X

$$X = \{x_1, x_2, \ldots, x_n, \sigma_1, \ldots, \sigma_n\}$$

A nova solução após ocorrer o operador de mutação é

$$X' = \{x'_1, x'_2, \ldots, x'_n, \sigma_1, \ldots, \sigma_n\}$$

onde

$$x'_i = x_i + \sqrt{\beta_i \times f(X) + \gamma_i} \times N_i(0, 1)$$

A constante de proporcionalidade β_i e o valor de *offset* γ_i são $2 \times n$ parâmetros endógenos que devem ser ajustados para uma tarefa em particular.

Os valores β_i e γ_i são utilizados com o valor 1 e 0, respectivamente, tal que:

$$x'_i = x_i + \sqrt{f(X)} \times N_i(0, 1)$$

Esta redução apresenta os seguintes problemas:

- Se o valor de aptidão do mínimo global é diferente de zero, então não é possível uma aproximação exacta ao mínimo global.

- Se o valor de aptidão é muito grande, então é esperado um comportamento aleatório na pesquisa. Uma diminuição do desvio padrão é como regredir através de mutações grandes.

- Ajustes simultâneos e eficientes, tanto na transformação linear entre a aptidão e a variância, como na projecção da junção, são impossíveis, quando o utilizador do algoritmo de PE não conhece o valor aproximado do ponto mínimo local.

2.14 Programação genética

2.14.1 Introdução

A programação genética (PG) utiliza uma pesquisa evolutiva no espaço de estruturas de árvores. Estas estruturas podem ser interpretadas como programas de computador, numa linguagem apropriada, que são modificadas pelos operadores de mutação e recombinação. A linguagem mais utilizada na PG é o LISP para implementar o espaço genótipo. Além desta linguagem são também utilizadas outras linguagens tais como C++ e por vezes assembly.

Nesta secção compara-se a PG com os AGs. De seguida estuda-se a representação, a função de aptidão, os operadores genéticos e a condição de finalização da PG. Por último,

é analisado como a diploidia e a dominância podem ser encarados na PG e qual a finalidade das funções definidas automaticamente.

2.14.2 Comparação entre a PG e os AGs

A PG é um ramo dos AGs, mas em certos aspectos é mais uma generalização do que uma especificação dos AGs.

Na PG não há correspondência entre o genótipo e fenótipo. As maiores diferenças entre a PG e os AGs são:

- Não linearidade, uso de árvores como estruturas e material genético.

Em geral os AGs usam material genético linear, apesar de existirem alguns AGs que utilizam material genético não linear. A PG opera quase sempre em material genético não linear e é usualmente estruturado em árvore.

- Material genético de tamanho variável.

A PG utiliza material genético que pode variar em tamanho. Por razões práticas são utilizadas limitação no comprimento das estruturas. Nos AGs o comprimento das *strings* é constante.

- Material genético executável.

A PG é uma evolução directa de programas para computadores, isto é, em quase todos os casos o material genético que está envolvido é executável de certa forma. As estruturas normalmente são “compreendidas” por um interpretador (algumas vezes numa linguagem idêntica ou semelhante a uma linguagem de computador existente, outras vezes numa linguagem desenhada para o problema). De qualquer maneira, em quase todos os casos, existe um conceito de executabilidade do material genético com o fim de realizar a função desejada da qual a função de aptidão é derivada.

2.14.3 Representação

Uma parte muito importante na representação esquemática dos AGs é a projecção que representa um ponto possível no espaço de pesquisa. A *string* tem um comprimento fixo de caracteres e cada cromossoma é um ponto no espaço de pesquisa do problema. A selecção de um *esquema* de representação, que facilite a resolução do problema através dos AGs, requer frequentemente um conhecimento considerável do problema e uma boa avaliação das dificuldades.

A PG é normalmente representada por meio de árvores dinâmicas compostas por *funções* (nós da árvore – não folhas) e *terminais* (folhas das árvores). Uma árvore com altura máxima pode ser representada por uma *string* de tamanho fixo. Na PG há sempre uma limitação prática na altura da árvore (normalmente 17) de modo a manter o problema tratável computacionalmente. O tamanho da limitação pode ser imposto pela altura máxima da árvore ou do tamanho de memória virtual do computador.

O espaço de pesquisa da PG é o espaço de todas as possibilidades de programas de computador, composto por *funções* e *terminais* apropriados ao domínio do problema. As funções podem ser operadores aritméticos, operações de programação e funções matemáticas convencionais e/ou funções lógicas e funções especificas do domínio.

Os parâmetros que controlam os PG são:

- Número de indivíduos da população;
- Número de gerações;
- Conjunto de terminais;
- Conjunto das funções primitivas;
- Função de aptidão;
- Método de guardar a solução que representa o melhor resultado;
- Critério para terminar uma execução;
- Probabilidade de cruzamento;
- Probabilidade de reprodução;
- Probabilidade de selecção dos pontos de cruzamento;

- Altura máxima da população;
- Altura máxima das árvores da população inicial;
- Probabilidade de mutação;
- Frequência da aplicação do operador de edição (*editing*);
- Frequência da aplicação do operador de encapsulamento;
- Condição para invocar o operador de dizimação;
- Percentagem de dizimação.

2.14.4 Função de aptidão

Na PG a interpretação de uma expressão em árvore é feita sem ter em consideração as posições dos alelos. Desta forma, tal como nos programas de computador, a interpretação do alelo "if" num programa genético é a mesma independentemente da posição na árvore.

Tipicamente, cada programa da população é executado para um número diferente de funções de aptidão. Assim, a sua aptidão é medida através da soma ou da média de várias representações diferentes. Os casos de aptidão podem ser escolhidos aleatoriamente ou de uma maneira estruturada (*i.e*. em intervalos regulares ou sobre uma grelha regular).

A variação dinâmica dos programas que são desenvolvidos ao longo da solução é uma característica importante da PG. É sempre difícil, e pouco natural, tentar especificar e restringir à *priori* o tamanho e a forma da eventual solução.

Outra característica importante da PG é a ausência ou pequeno número de regras de pré-processamento das entradas e pós-processamento das saídas. As entradas, os resultados intermédios e as saídas são expressos directamente em termos da terminologia natural no domínio do problema.

2.14.5 Algoritmo de programação genética

Os passos necessários num algoritmo de PG são:

- Gerar uma população inicial de composições aleatórias de funções e terminais do problema (*i.e*. programas de computador).

- Executar interactivamente os seguintes subespaços até o critério de conclusão ser satisfeito.
  i) Executar cada programa da população e atribuir um valor de aptidão utilizado a função de aptidão.
  ii) Criar uma nova população de programas de computador aplicando os dois operadores primários seguintes:
    1) reprodução de um programa, copiando-o para a nova população.
    2) criar dois novos programas de computador, a partir de dois programas existentes, por recombinação genética aleatória escolhendo partes de dois programas existentes, utilizando o operador de cruzamento, aplicado a um ponto de cruzamento escolhido aleatoriamente em cada programa.
  iii) programa que é identificado pelo método como resultado (*i.e*. o indivíduo melhor até ao momento) é guardado como o resultado do algoritmo genético para a execução. Este resultado pode representar a solução (ou a solução aproximada) para o problema.

2.14.6 Operadores genéticos

2.14.6.1 Introdução

Esta secção descreve os operadores utilizados para modificar as estruturas. São apresentados dois operadores primários (reprodução e cruzamento) e cinco operadores secundários (mutação, permutação, edição, encapsulamento e dizimação).

2.14.6.2 Reprodução

Na PG o operador de reprodução baseia-se na lei de selecção e de sobrevivência do indivíduo mais apto. O operador consiste em escolher uma *string* (a escolha é baseado num método de selecção) da população, sendo aquela colocada na nova geração. Assim, o operador de selecção é assexual.

Os métodos de escolha baseados nos métodos de selecção são:

- Reprodução proporcional à aptidão – a escolha é feita através do método de selecção proporcional.
- Reprodução pelo posto – a escolha é feita através da selecção pelo posto.
- Reprodução por torneio – a escolha é feita pela selecção por torneio

Pode ser poupado um tempo apreciável se não for calculado o valor de aptidão de uma *string* que aparece na geração presente como resultado da reprodução da geração anterior. A aptidão da *string* da geração anterior não será modificada e não será necessário recalcula-la (a menos que a aptidão varie de geração para geração). Se a reprodução for aplicada a 10% da população, em cada geração, esta técnica resulta em menos de 10% de cálculos de aptidão, em cada geração. O cálculo da aptidão consome uma grande quantidade de tempo computacional para problemas não triviais pelo que esta técnica simples poupa, de imediato, 10% de tempo do algoritmo.

2.14.6.3 Operador de cruzamento

O operador de cruzamento é sensível à estrutura de árvores que são manipuladas. O esquema de cruzamento simples preserva as restrições sintácticas da representação pela troca completa de sub-árvores entre os pais. O operador deve ter a capacidade de identificar sub-árvores completas e assegure a construção de programas sintacticamente válidos, removendo todos os programas sintacticamente inválidos.

Actualmente a PG usa operadores de cruzamento que preservam as restrições de sintaxe da linguagem de representação, tratando-se da única distinção entre os AG canónicos e a PG. Mais importante é a conveniência conceptual da representação dinâmica, a interpretação não-posicional e a manipulação da representação da PG, que mostram o ponto de vista mais restrito da investigação prévia dos AGs.

Considere-se dois pais da figura 2.10. Se forem escolhidos, respectivamente, os nós 6 e 3 da árvore esquerda e direita então os descendentes são obtidos trocando essas sub-árvores (figura 2.11).

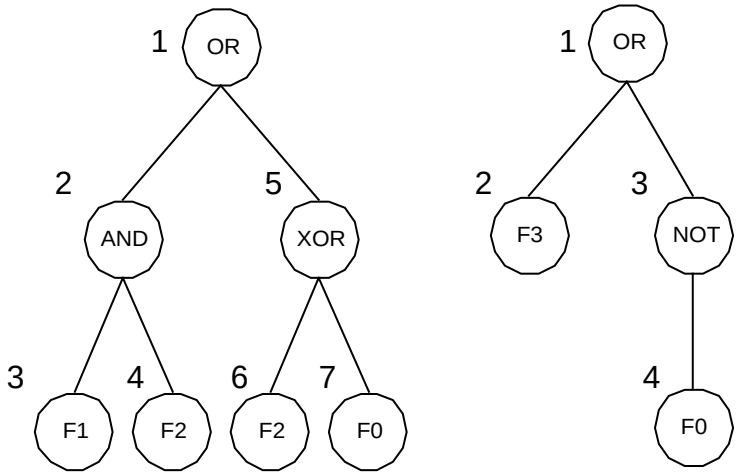


Figura 2.10 Dois programas pais.

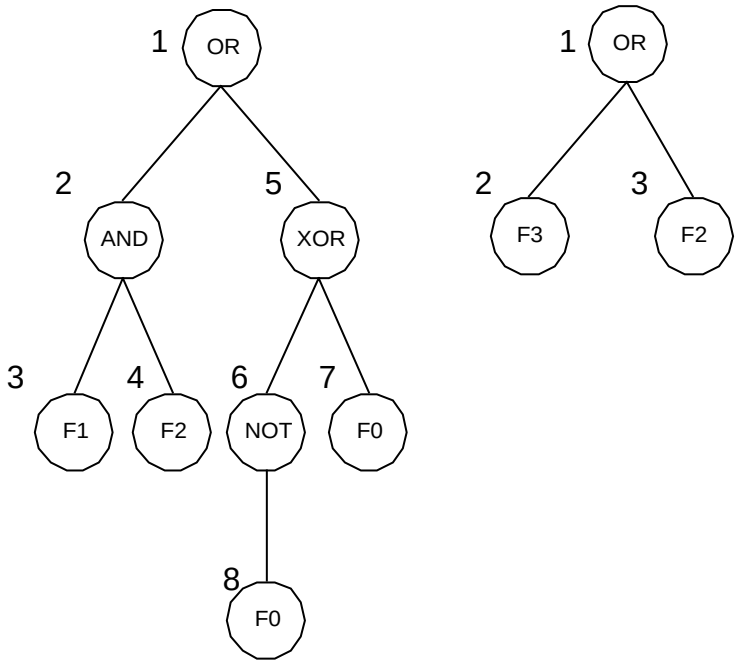


Figura 2.11 Programas descendentes.

Os nós escolhidos para cruzamento podem ser as raízes. Neste caso, os descendentes são iguais aos pais.

Quando um indivíduo (incestuosamente) casa consigo mesmo, isto é, casa com cópias do mesmo, os dois descendentes resultantes serão diferentes excepto quando o ponto de cruzamento é o mesmo.

O operador de reprodução cria uma tendência no sentido de convergência mas o operador de cruzamento exerce um contrapeso no sentido contrário.

No cruzamento pode verificar-se que a altura de uma das árvores seja muito grande. Neste caso, o cruzamento é abortado podendo o primeiro pai casar com um elemento da nova geração.

2.14.6.4 Operador de mutação

O operador de mutação introduz mudanças aleatórias na população. O operador começa por seleccionar um nó aleatório (*e.g*. uma função ou um terminal) da árvore. O operador de mutação elimina esse nó, ou uma sub-árvore, e substitui–o(a) por uma sub-árvore criada aleatoriamente. Por exemplo, se for escolhido o nó 5 da árvore esquerda da figura 2.11, o programa após a mutação vem:

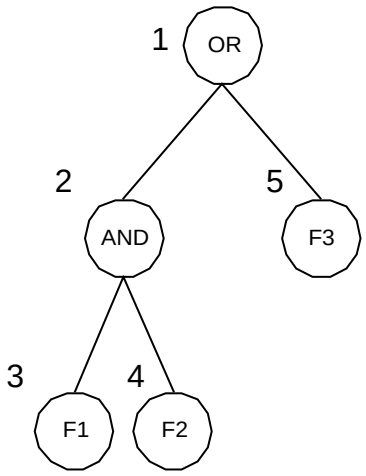


Figura 2.12 Árvore após a ocorrência de mutação no nó 5.

2.14.6.5 Operador de permutação

O operador de permutação é uma generalização do operador de inversão dos AGs. O operador é assexual, sendo escolhido o pai do mesmo modo que para os operadores de cruzamento e recombinação. O operador começa por determinar um nó *função* da árvore. De seguida, é escolhida uma permutação de *k*! (sendo *k* o número de filhos do nó respectivo) para a nova disposição da sub-árvore. Um exemplo que ilustra a permutação no nó 2 é apresentado na figura 2.13.

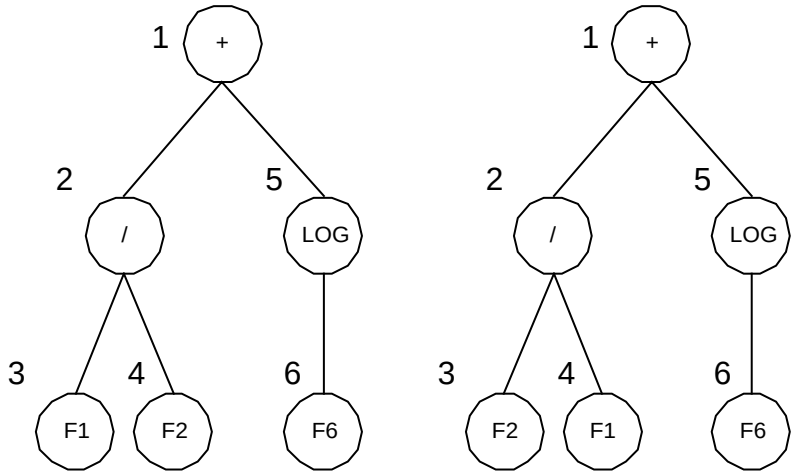


Figura 2.13 Árvores antes e depois de ocorrer o operador de permutação no nó 2.

2.14.6.6 Operador de edição

O Operador de edição (*editing*) permite editar programas e simplificá-los. Na tabela 2.2 encontram-se alguns exemplos.

Tabela 2.2 Simplificação de programas em LISP.

Programa inicial	Programa final
AND X X	X
OR X X	X
+ 1 1	2
/ 3 3	1

2.14.6.7 Operador de encapsulamento

O operador de encapsulamento permite identificar automaticamente uma sub-árvore com um potencial desempenho, sendo referenciada e usada noutros programas. Este operador é utilizado para resolver problemas de grande dimensão quando estes podem ser divididos em vários subproblemas. O operador de encapsulamento é assexual e o indivíduo é seleccionado de forma análoga à reprodução e cruzamento.

A operação de encapsulamento começa por seleccionar um nó *função* da árvore (do código) resultando a mesma árvore (código) e a definição de uma sub-árvore (subfunção). O operador de encapsulamento retira a sub-árvore e define uma função que permita referenciar essa sub-árvore. Esta função não tem argumentos e facilita a compilação do programa.

O operador de encapsulamento tem como efeito a sub-árvore seleccionada no indivíduo novo deixar de estar sujeita ao efeito disruptivo do operador de cruzamento, pois ela é agora representada, na árvore, por um nó indivisível que referencia a função.

2.14.6.8 Operador de dizimação

O operador de dizimação fornece um meio rápido para lidar com o seguinte problema: quando existe uma distribuição de valores de aptidão muito enviesada, os indivíduos que têm um valor ligeiramente superior, começarão a dominar a população e, consequentemente, a diversidade da população começa a diminuir.

O operador de dizimação é controlado por dois parâmetros: a percentagem de uso e a condição que especifica quando o operador deve ser evocado. Por exemplo, seja a percentagem ser 10% e o operador seja invocado na geração 0. Nessa situação, após ser calculada a função de aptidão da geração 0, a função é invocada e apaga 90% da população. Se o operador é chamado na iteração 0 então o programa deve começar com 10 vezes a população desejada para o resto da execução do problema. Se não existirem indivíduos duplicados na população inicial, e se é aplicado o operador de dizimação na segunda geração, continuam a não existir indivíduos duplicados.

2.14.7 Condição de finalização

A PG termina quando o número de gerações foi atingido ou quando se verifica alguma especificação do problema. Esta especificação pode ocorrer quando é encontrada 100% da população de soluções correctas. Alternativamente utiliza-se um critério quando não se espera a solução correcta ou interactivamente, após algumas gerações analisa-se a

solução e se esta satisfaz o utilizador o processo pára, senão são executadas mais um conjunto de gerações.

2.14.8 Diplóide e dominância

Quando o resultado lógico da *condição* (if) é independente do estado do ambiente a condição representa uma forma estática de diploidia, com a acção dominante determinada pela *condição*. Por exemplo, com a *condição* que leva sempre ao resultado falso, a *acção 2* será a parte dominante da sub-árvore. A sub-árvore recessiva nunca é executada mas, em poucas gerações pode ficar dominada se for requerido num descendente. Contrariamente às formas normais de dominância na computação, dominância na PG não está na representação mas surge dinamicamente.

A *condição* pertence ao estado condicional e só é mutável pelo processo evolutivo. Consequentemente, a dominância das acções é especifica ao indivíduo. Além do mais, o cálculo da dominância não precisa de ser estático mas pode ser dependente do contexto da situação. Quando uma *condição* usa elementos de estado do ambiente isso permite uma forma especifica de dominância.

2.14.9 Funções definidas automaticamente

Está demostrado que a dificuldade do problema visto pela PG cresce muito rapidamente quando a escala do problema é aumentada. O uso de funções definidas automaticamente (FDAs) é uma técnica que permite subprogramas envolverem-se com muitos programas na PG, reduzindo quer o esforço computacional requerido para resolver o problema quer a taxa de aumento da dificuldade com o aumento da escala do problema.

2.15 Outros algoritmos evolutivos

2.15.1 Introdução

Nesta secção apresentam-se sucintamente alguns algoritmos derivados dos AGs nomeadamente o método baseado na ordem (*order-based*) e os sistemas de classificação.

2.15.2 Algoritmos genéticos baseados na ordem

Os algoritmos baseados na ordem (*order-based*) foram propostos para pesquisar directamente o espaço de permutações $\pi:\{1, \ldots,n\}\rightarrow\{1,\ldots,n\}$ evitando o uso de funções de descodificação complexas para projectar *strings* binárias em permutações e assim preservá-las no domínio quando sofrem mutações e cruzamentos. Para preservar as permutações os algoritmos têm operadores especiais de mutação (*e.g.* trocas aleatórias de permutações de dois elementos) e de cruzamento (*e.g.* OX e PMX).

2.15.3 Sistemas de classificação

Os sistemas de classificação usam um algoritmo evolutivo para pesquisar o espaço das regras produzido (normalmente codificado por *strings* sobre um alfabeto ternário, mas que por vezes usam regras simbólicas) de um sistema de aprendizagem capaz de induzir e generalizar. Tipicamente existem duas aproximações: Michigan e Pittsburgh que são diferenciadas de acordo com um dos seguintes casos respectivamente:

- a um indivíduo corresponde uma regra simples do sistema de regras;
- a um indivíduo corresponde uma base completa de regras.

2.16 Referências

[1] David B. Fogel, *Editor-in-Chief* Natural Selection, Inc, La Jolla, CA 92037 USA, "Evolutionary Computation: A New Transations", IEEE Transactions on Evolutionary Computation, Vol. 1, n. 1, pp. 1-2, April 1997.

[2] David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison – Wesley Publishing Company, Inc.

[3] D. Beasley, D. R. Bull, R.R. Martin, "An overview of genetic algorithms: part 1, fundamentals", University Computing, pp 58-69, 1993, Inter Committee on Computing.

[4] D. Beasley, D. R. Bull, R.R. Martin, "An overview of genetic algorithms: part 2, fundamentals", University Computing, pp170-180, 1993, Inter Committee on Computing.

[5] Gray B. Parker, "Generating Arachnid Robot Gaits With Cyclic Genetic Algorithms", http://www.cs.indiana.edu/hyplan/gaparker.html, 29-5-1998.

[6] J. J. Grefenstette, Ed., Lawrence Erlbaum Associates, "Reducing bias and ineficienty in the selection algorithm", Proc. 2nd Int. Conf. Genetic Algorithms, pp 14-21, 1995.

[7] Jin-Oh Kim and Pradeep K. Khosla, "A Multi-Population Genetic Algorithm And Its Application to Design Of Manipulators, IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol 1, pp 279-286, 7-10 July 1992.

[8] John R. Koza, "Genetic Programming On the Programming of Computers by Means of Natural Selection", A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

[9] Kenneth E. Kinnear, Jr "Advances in Genetic Programming". The MIT Press, Cambridge, Massachusetts, London, England.

[10] K. F. Man, K. S. Tang, S. Kwong, “Genetic Algorithms: Concepts and aplications”, IEEE Transations on Industrial, Electronics, Vol. 43, pp 519-534, October 1996.

[11] L. Darrel Whitley, “Foundations os Genetic Algorithms 2”, Morgan Kaufmann Publishers, San Mateo, California, ISBN 1-55860-263-1, 1993.

[12] Thomas Bäck, “Evolutionary Algorithms in theory and Practice, Evolutionary Strategies – Evolutionary Programming – Genetic Algorithms”, New York – Oxford, Oxford University Press, ISBN 0-19-509971-0, 1996.

[13] Thomas Bäck, Ulrich Hammel, Hans-Paul Schwefel, “Evolutionary Computation: Comments on the History and Current State”, IEEE Transations on Evolutionary Computation, Vol. 1, n. 1, pp. 3-17, April 1997.

[14] Yaval Davidor, “Genetic Algorithms and Robotics, a heuristic strategy for optimization”, World Scientific, Singapure, ISBN 9810202172, 1991.

[15] Yuval Davidor, “Analogous Crossover”, Procedings of the third International Conference on Genetic Algorithms, pp. 98-103, George Manson University, 1989.

[16] Zbigniew Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs”, ISBN 3-540-60676-9, Third Edition, Springer, 1996.

3 Aplicação de Algoritmos Genéticos à robótica

3.1 Introdução

Neste capítulo são apresentadas aplicações dos AGs à robótica. Assim, na secção 3.2 são descritas algumas aplicações na geração de trajectórias para robôs móveis. Na secção 3.3 são apresentados algoritmos para a construção ou selecção do robô que melhor se adapta a um determinado ambiente. De seguida, na secção 3.4, são estudados algoritmos de locomoção para robôs. Na secção 3.5 são descritos algoritmos para a geração de trajectórias para manipuladores robóticos. Por último, na secção 3.6 é apresentada a estimação de parâmetros de robôs com vista a sua calibração.

3.2 Planeamento de trajectórias para robôs móveis

3.2.1 Introdução

Os AGs são utilizados na geração de trajectórias e na navegação de robôs móveis [1],[4],[11],[13],[21],[23]. Num esquema de navegação é desejável que se encontre o ponto de destino sem ocorrer uma eventual colisão com os obstáculos. Assim, conhecendo o robô e o respectivo ambiente, o planeamento de trajectórias consiste em determinar uma trajectória entre dois locais específicos, livre de colisões.

Existem dois tipos de planeamento: o planeamento em *off-line* e o planeamento em *on-line*.

Os algoritmos propostos inicialmente eram concebidos para ambientes perfeitamente conhecidos e estacionários. Utilizavam AEs canónicos, sem a utilização de conhecimento especifico do domínio, adoptavam mapas discretos e revelavam-se pouco flexíveis em relação a adaptabilidade de mudanças no ambiente.

Nas secções seguintes são apresentados três exemplos de aplicações de AGs a geração de robôs móveis.

3.2.2 Planeamento do movimento de um manipulador móvel

3.2.2.1 Introdução

Chen e Zalzala [13] apresentam um algoritmo para um manipulador robótico montado em cima de um veículo móvel (figura 3.1). O algoritmo tem como finalidade optimizar o percurso do robô móvel, a posição e a configuração do manipulador, sem ocorrer qualquer colisão.

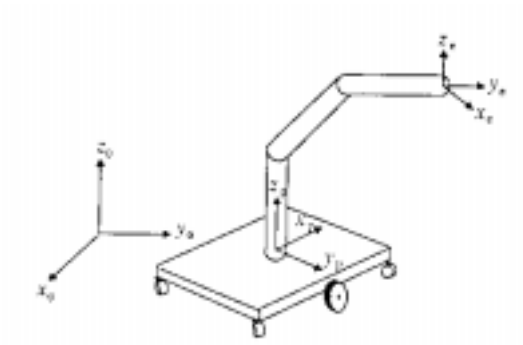


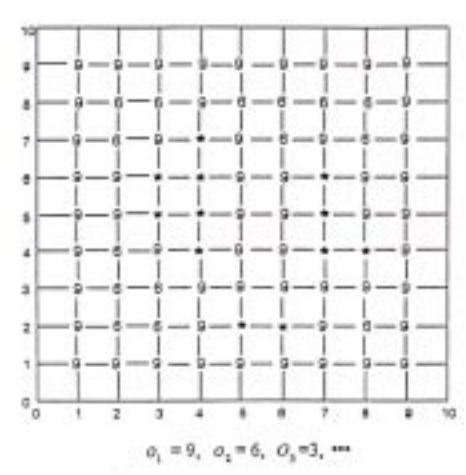
Figura 3.1 Sistema manipulador robótico [13].

3.2.2.2 Ambiente do robô

O ambiente do robô é representado por um conjunto de células (figura 3.3) e por duas grelhas numéricas, uma para os campos potenciais e para a meta, e a outra para os obstáculos.

O campo potencial numérico (figura 3.2) é construído da seguinte forma:

- $U = o_0$ para cada ponto que coincide com as fronteiras dos obstáculos;
- $U = o_1$ ($o_0 > o_1$) para o ponto para qualquer ponto na vizinhança do obstáculo;
- $U = o_2$ ($o_1 > o_2$) para qualquer ponto na vizinhança dos pontos anteriores (o_1);
- Até que todos os pontos da grelha serem afectados de um valor.



Legenda: * – obstáculo.

Figura 3.2 Campo potencial numérico [13].

A construção do campo potencial para a meta é idêntico. Começa-se por atribuir o valor zero para a meta, de seguida atribui-se um peso a cada célula vizinha da meta, com um valor superior a esta, e assim por diante.

3.2.2.3 Representação

As *strings* (de comprimento constante) para o robô móvel são codificadas em valores inteiros e são representado da seguinte forma:

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}$$

3.2.2.4 Função de aptidão

Para a função de optimização é considerada: a distância aos obstáculos, o binário aplicado, a capacidade de manipulação e a distribuição óptima do binário.

3.2.2.5 Operadores genéticos

Os operadores genéticos adoptados são:

- Operador de reprodução – método de amostragem estocástica.

- Operador de cruzamento (figura 3.3) – método do cruzamento análogo.

  Quando existem vários pontos que coincidem nas duas trajectórias é escolhido aleatoriamente um destes pontos para o cruzamento. No caso contrario escolhem-se dois pontos relativamente próximos, um de cada trajectória, e gera-se um caminho para cada trajectória entre um dos pontos seleccionado e o ponto seguinte da outra trajectória.

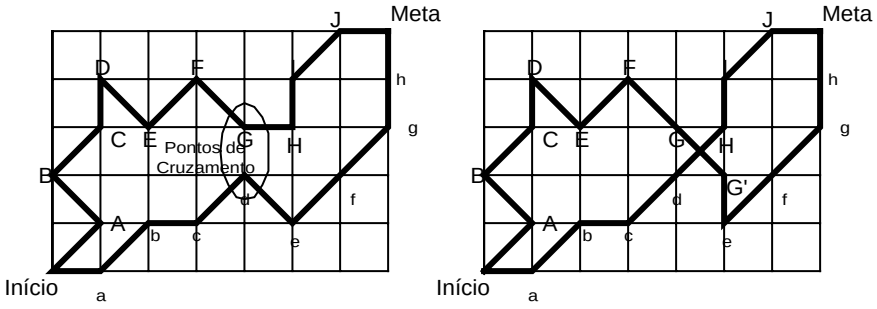


Figura 3.3 Operador de cruzamento [13].

- Operador de mutação. Escolhe-se aleatoriamente um ponto da trajectória e, a partir deste ponto até ao final da *string* os pontos são gerados aleatoriamente.

3.2.3 Navegador/planeador evolutivo adaptativo

3.2.3.1 Introdução

O navegador/planeador evolutivo adaptativo (NPE) [4] foi desenvolvido de modo a ser flexível e adaptativo. Utiliza métodos de planeamento em *off-line* e em *on-line* que permitem:

- Modificar os critérios de optimização;
- Incorporar vários tipos de conhecimento específico do domínio;
- Incluir opções entre caminhos curtos, planeamentos com alta eficiência e lidar com ambientes apresentando obstáculos desconhecidos.

3.2.3.2 Algoritmo do navegador/planeador

O navegador NPE pode regular o seu desempenho de acordo com o ambiente e com suas modificações, adaptando permanentemente as probabilidades dos seus operadores e ajustando os percursos, mesmo quando o robô se encontra em movimento.

O NPE encontra-se explanado no algoritmo 3.1

```
Procedimento NPE
t ← 0
se known_path então
      input P(t)
senão
      inicializa P(t)
fim se
avaliar P(t)
repetir enquanto cond <> fim
      t ← t + 1
      seleccionar operador o_j com probabilidade p_j
```

```
      seleccionar pai(s) de P(t)
      produzir um descendente aplicando o operador o aos
                                        pai(s) escolhidos
      avaliar os novos descendentes
      substituir os novos descendentes pelos piores
                                  elementos da população
      escolher o melhor elemento p de P(t)
      se on_line e p é admissível e resto ( t / n ) = 0
                                                 então
         mover um passo k_max no percurso determinado por p
                             enquanto observa ambiente
         modificar os valores em todos os indivíduos de
                        acordo com a nova posição inicial
         se existir alguma mudança observada então
           actualiza mapa
         fim se
         avalia P(t)
      fim se
fim repetir
retornar
```

Algoritmo 3.1 Algoritmo NPE.

O algoritmo do navegador NPE usa o método dos AGs em regime permanente. A variável *Know_path* indica se é necessário criar uma população inicial P(0). No caso afirmativo, a população P(0) utilizada pode ser a população final da última pesquisa ou então pode ser escolhida através de outro método. Por seu lado a variável *on_line* indica qual o tipo de planeamento utilizado.

O algoritmo NPE em modo *on-line* corre dois processos em paralelo:

- Navegação do robô, ao longo da melhor trajectória, enquanto observa o ambiente com o fim de detectar novos objectos.

- Execução do AE para encontrar eventuais trajectórias melhores, tendo em conta a nova posição do robô e os novos objectos detectados (se existirem).

Quando é detectada uma nova trajectória mais eficiente, o percurso em uso é substituído por essa nova trajectória.

Como o algoritmo permite executar o planeamento da trajectória tanto em *on-line* como em *off-line*, uma técnica comum consiste em usar no algoritmo *on-line* a população final após a execução do algoritmo em *off-line*.

3.2.3.3 Representação das trajectórias

Uma trajectória é representada por uma *string*, correspondente a segmentos rectos, como consequência dos pontos que constituem as *strings*. Cada *string* é composta por parâmetros e cada um deste é constituído por três valores reais. Os dois primeiros valores são as coordenadas de um ponto e o terceiro indica se este ponto é ou não admissível.

x_1	y_1	b_1		x_n	y_n	b_n

O número de pontos intermédios da *string* é variável.

3.2.3.4 Função de aptidão

Neste algoritmo são considerados dois tipos de funções de aptidão, consoante as *strings* são ou não admissíveis.

Para as *strings* admissíveis (todos os pontos da *string* são admissíveis) a função de aptidão é

$$f = w_d \times \text{dist}(p) + w_s \times \text{smooth}(p) + w_c \times \text{clear}(p)$$

onde:

w_d, w_s e w_c são os pesos de cada critério de acordo com a sua importância.

- $\text{dist}(p) = \sum_{i=1}^{n-1} \text{d}(m_i, m_{i+1})$, é a função que calcula a distância total da trajectória;

- $\text{smooth}(p) = \max_{i=2}^{n-1} \text{s}(m_i)$, é a função que calcula a curvatura máxima dos pontos intermédios, sendo *s* definida por:

$$\text{s}(m_i) = \frac{\theta_i}{min\{\text{d}(m_{i-1}, m_i), \text{d}(m_i, m_{i+1})\}};$$

o parâmetro $\theta_i \in [0, \pi]$ é o ângulo entre os segmentos de recta ligando a trajectória m_{i-1} e m_i e o segmento de recta m_i e m_{i+1}.

- $\text{clear}(p) = \max_{i=1}^{n-1} c_i$, onde

$$c_i = \begin{cases} g_i - \tau, & \text{se} \quad g_i \geq \tau \\ e^{a \times (\tau - g_i)} - 1, & \text{outros} \quad \text{casos} \end{cases}$$

o parâmetro g_i é a distância mais pequena do segmento $m_i - m_{i+1}$ para todos os objectos, a distância τ define a distância mínima de segurança relativa a um objecto e *a* é um coeficiente. Quando a distância diminui abaixo da distância de segurança a penalização cresce exponencialmente. A função clear(*p*) é definida como o máximo dos c_i que indica se uma trajectória é perigosa devido ao seu percurso passar perto de um obstáculo.

A função de aptidão para as trajectórias não admissíveis tem em conta os seguintes factores:

- Número de intersecções da trajectória com os obstáculos;
- A profundidade da intersecção do percurso com a trajectória;

- A taxa de percurso admissível *versus* o percurso não admissível;
- Comprimento da trajectória;
- A pior trajectória admissível é sempre melhor do que uma trajectória não admissível.

3.2.3.5 Operadores utilizados

Os operadores implementados no algoritmo são:

- Cruzamento de ponto simples;
- Mutate_1 – muda ligeiramente um ponto intermédio. Este operador é utilizado para fazer pequenos ajustamentos na trajectória;
- Mutate_2 – utilizado para fazer grandes modificações num ponto intermédio;
- Inserir-remover – inserir/remover nós intermédios em trajectórias não admissíveis;
- Swap – troca dois parâmetros (genes) seguidos;
- Smooth – Suaviza uma trajectória admissível substituindo um ponto por dois de modo a remover o canto do percurso;
- Repair – utilizado em trajectórias não admissíveis. Troca uma sequência de pontos não admissíveis (que cruzam um obstáculo) por um conjunto de pontos de modo a que a trajectória contorne esse obstáculo.

3.2.3.6 Reprodução e selecção

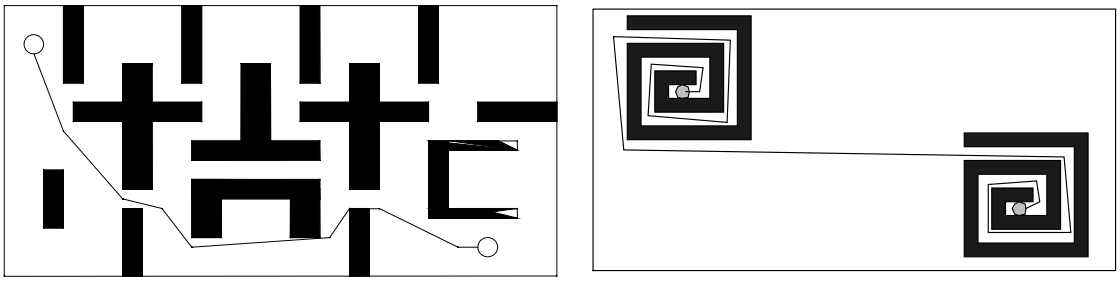
A selecção é feita por torneio e o operador de reprodução é proporcional à aptidão.

3.2.3.7 Probabilidades adaptativas dos operadores

As probabilidades dos operadores começam todos com a mesmo valor, após um intervalo o algoritmo usa um mecanismo automático para medir os desempenhos dos operadores genéticos e adaptar as probabilidades p_i de acordo com esse desempenho.

3.2.3.8 Experiências e resultados em modo *off-line*

Alguns resultados obtidos no modo *off-line* estão apresentados na figura 3.4. Os resultados obtidos estão perto da solução ideal, com um número médio de gerações para obtenção de resultados de T = 400 gerações. Normalmente são necessárias T = 600 gerações para obter resultados satisfatórios.



T = 800 T = 400

Figura 3.4 Trajectórias em modo *off-line* [4].

3.2.3.9 Experiências e resultados em modo *on-line*

Nesta simulação o robô conhece parte do ambiente (o robô desconhece alguns objectos) e tem uma visão de raio *R*. O robô detecta um objecto desconhecido quando se aproxima deste de uma distância *R* sendo então o objecto anotado no ambiente do robô.

Na navegação *on-line* o estado desta é verificado em cada *n* gerações e, deste modo, é fornecido ao robô o melhor percurso admissível. O robô move-se em passos de comprimento k_{max} ($k_{max} < R$ – denota o comprimento máximo desses passos). Se o segmento do percurso tem um comprimento inferior ao valor de k_{max} então o robô executa-o de uma só vez. No caso contrário, o robô move-se ao longo do segmento através de vários passos. O processo de evolução é executado em paralelo com o movimento do robô, pelo que quando é encontrado um percurso melhor o robô passa a seguir este no passo seguinte. Quando o robô detecta a presença/ausência de um objecto desconhecido/conhecido insere-o/remove-o do ambiente. Consequentemente, a trajectória pode ser modificada e as funções de aptidão serem recalculadas, pois os percursos podem tornar-se não admissíveis.

A figura 3.5 apresenta o ambiente do robô com os objectos desconhecidos representados a “transparentes”. Inicialmente o robô segue a trajectória encontrada em *off-line* (figura 3.5-a). Quando o robô encontra um obstáculo (figura 3.5-b e figura 3.5-c) a trajectória é recalculada de acordo com o novo ambiente (figura 3.5-c e figura 3.5-d).

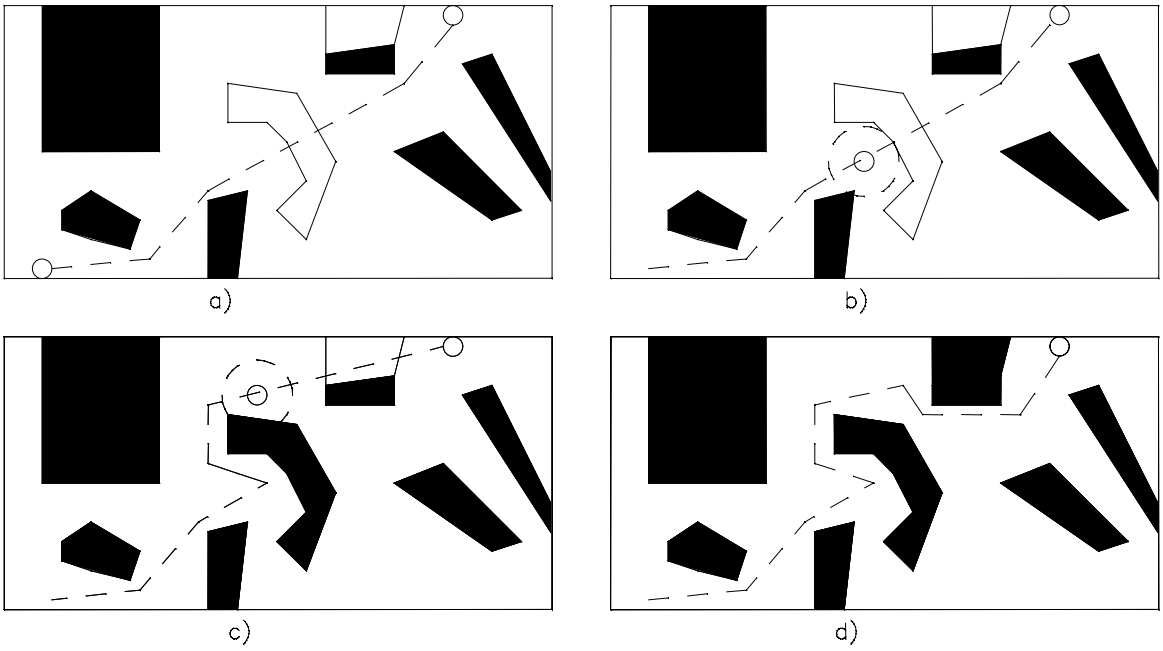


Figura 3.5 Navegação do robô on-line [4].

a) percurso do robô, com dois obstáculos desconhecidos (objectos transparentes);
b) instante em que o robô descobre o primeiro obstáculo desconhecido, neste ponto o objecto é inserido no ambiente e geradas novas trajectórias;
c) robô a seguir o novo percurso;
d) percurso que o robô seguiu.

3.2.4 Planeamento de trajectórias com desvio dinâmico de obstáculos

3.2.4.1 Introdução

Han *et al*. [21] propõem também um algoritmo de navegação para um robô móvel em tempo real com a capacidade de evitar obstáculos. O algoritmo funciona em ambien-

tes dinâmicos. A função de custo toma em consideração a distância e a segurança do percurso.

O robô é munido de um sistema de visão (tanto pode estar incluído neste como em qualquer parte do ambiente) e de um algoritmo que planeia o percurso. Num sistema dinâmico o algoritmo deve identificar em tempo real o movimento de obstáculos e a geração de percursos admissíveis deve ser também efectuada em tempo real.

3.2.4.2 Representação das trajectórias

Uma trajectória é constituída pelas coordenadas (*x, y*) de todos os pontos intermédios entre o ponto inicial e o ponto final. Por forma a diminuir o comprimento da *string* e o tempo computacional, cada ponto intermédio (*x, y*) é projectado na semi-recta que liga o ponto de origem ao ponto de destino. Se as distâncias entre as projecções forem iguais é necessário guardar apenas a distância entre esses pontos e as sua projecções (figura 3.6) (espaço dos g's). A *string* guarda os valores dos g's em modo binário.

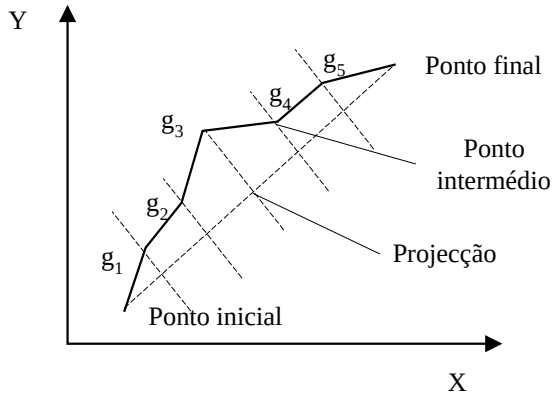


Figura 3.6 Estrutura de codificação [21].

3.2.4.3 Função de aptidão

A função de aptidão é medida de acordo com a distância do percurso e de acordo com a distância aos obstáculos dada por:

$$f = \alpha \times \sum_{i=1}^{n} \frac{L_i}{l_i} + \beta \times \sum_{i=1}^{n} d_{\min}$$

onde:

l_i – é a distância entre o ponto intermédio *i* e o ponto final;

L_i – é a distância entre a projecção do ponto *i* e o ponto final;

$d_{\min}$ – é a distância mínima entre os pontos intermédios e os obstáculos;

α eβ – são os pesos de cada factor na função de aptidão.

3.2.4.4 Resultados

O algoritmo apresenta bons resultados enquanto a velocidade do robô permanecer inferior a 1,2 m/s. Na figura 3.7 está ilustrado o efeito do peso de cada componente na função de aptidão.

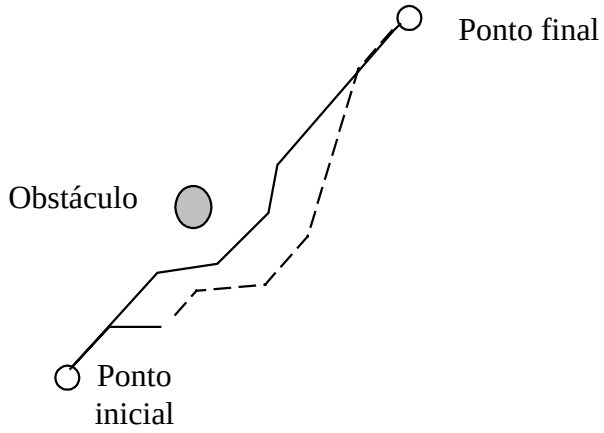


Figura 3.7 trajectória final entre dois pontos [21].

Legenda:

linha continua – $\alpha = 300, \beta = 0{,}01$;

linha descontinua – $\alpha = 100, \beta = 0.3$.

3.3 Escolha e desenho de manipuladores robóticos

3.3.1 Introdução

A escolha e desenho de um manipulador robótico deve ter em atenção a tarefa a ser executada e os obstáculos que se encontram no ambiente de trabalho [5],[6],[15],[16]. Nesta secção é apresentado um exemplo onde é escolhido o robô que tem o melhor desempenho para uma tarefa entre vários robôs possíveis. Nas subsecções seguintes são apresentados diversos métodos para projectar o melhor manipulador robótico que satisfaz determinadas restrições.

3.3.2 Selecção do melhor robô

3.3.2.1 Introdução

Chedmail e Ramstein [16] apresentam um algoritmo para escolher o robô que desempenha melhor uma tarefa, dentro de um conjunto de robôs com determinadas morfologias e posições. A tarefa consiste em seguir uma determinada trajectória num ambiente com obstáculos.

3.3.2.2 Representação

A *string* é constituída por:

$$X = \begin{bmatrix} T \\ L \\ x \end{bmatrix}$$

onde:

T – indica o tipo de mecanismo da *string* X, T pertence ao conjunto de robôs;

L_i – coordenada i do vector que guarda os comprimentos dos elos;

x_i – coordenada i da posição/orientação da base do mecanismo.

3.3.2.3 Função de aptidão

O objectivo do problema consiste em minimizar a trajectória seguida pelo ponto terminal do mecanismo, sem ocorrerem colisões.

3.3.2.4 Operadores genéticos

- O operador de selecção é baseado no método da roleta redonda.

- O operador de cruzamento consiste em trocar o tipo ou a posição da base entre as duas *strings* (manipuladores).

- O operador de mutação tem como função mudar o tipo de mecanismo ou a posição da base.

3.3.2.5 Problemas a resolver

Nesta secção são apresentados dois exemplos, sendo um bidimensional e o outro tridimensional.

A primeira simulação está representada na figura 3.8, sendo os robôs a utilizar: um robô RR e um robô RP. A limitação das juntas rotacionais é de [0º; 360º] e das juntas prismáticas é de [0; L]. A base do robô deve estar localizada no eixo conforme indicado na figura 3.8.

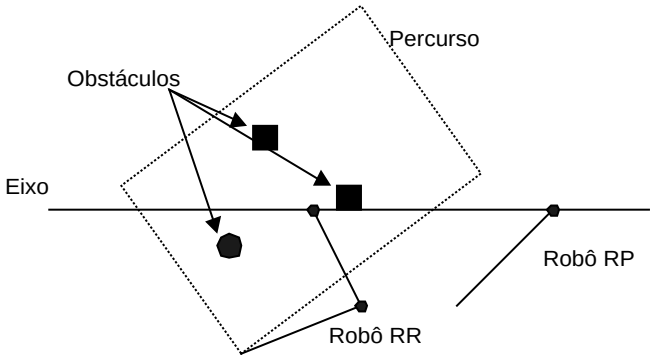


Figura 3.8 Ambiente da simulação 1 [16].

O mecanismo escolhido pelo algoritmo foi o mecanismo RR ao fim da geração 40. Neste exemplo o robô escolhido consegue seguir toda a trajectória.

A 2º simulação é baseada num ambiente real (figura 3.9), onde são comparados quatro tipos de mecanismos:

- Robô cilindrico th8
- Robô Scara
- Robô Puma
- Robô Silbot

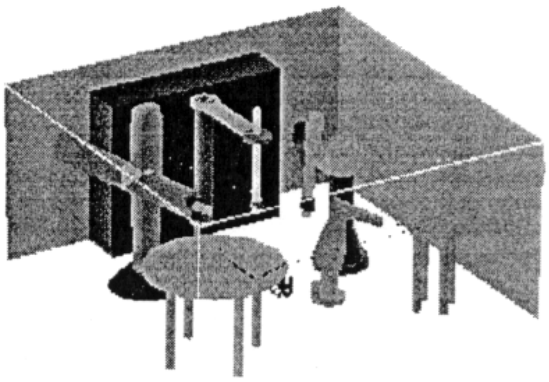


Figura 3.9 Ambiente da simulação 2 [16].

Pretende-se que o robô siga um determinado percurso num ambiente com muitas restrições.

A convergência do algoritmo foi obtida ao fim de cinco gerações e o robô escolhido pelo algoritmo foi o th8. O ponto terminal do robô consegue seguir 95% da trajectória. A posição da base do robô é determinada ao fim de 190 iterações.

3.3.3 Projecto de robôs modulares utilizando AGs

3.3.3.1 Introdução

J. Han *et al.* [5] apresentam um algoritmo para construir um robô modular a partir de um conjunto de elos e de juntas. A construção da estrutura mecânica tem como objectivo incluir o número mínimo de módulos que satisfaça a tarefa pretendida.

3.3.3.2 Resolução do problema

Para determinar o comprimento óptimo de cada elo o algoritmo é dividido em dois passos:

- obter a configuração necessária do robô utilizando as equações da cinemática;
- calcular o comprimento óptimo dos elos.

O robô é constituído por três tipos de módulos:

- base do manipulador (sem nenhum grau de liberdade);
- elos;
- dois tipos de juntas (prismáticas e rotacionais).

Para determinar a configuração adequada do manipulador robótico são necessárias duas variáveis: uma relativa à base do robô e outra relativa ao ponto terminal.

O sistema de coordenadas esta representado na figura 3.10.

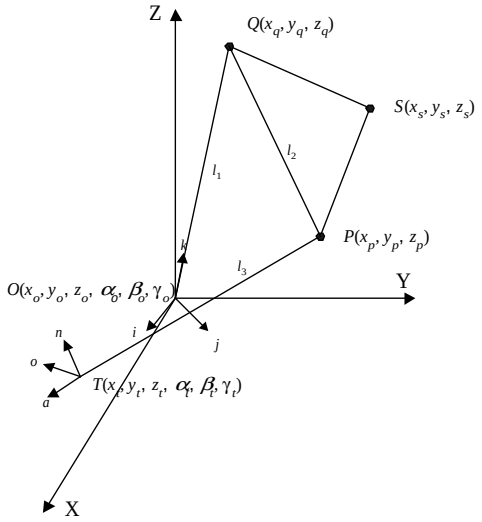


Figura 3.10 Sistema de coordenadas [5].

Inicialmente o algoritmo parte com um robô composto por três elos simples 'BBB' (B é uma junta pivô, na figura 3.10 estas juntas estão situadas nos pontos *P*, *Q* e *S*). De seguida, de acordo com as condições da cinemática, verifica-se a necessidade de adicionar uma junta de rotação ('R') entre as juntas iniciais.

A posição *P* e *Q* são determinadas pelas seguintes equações:

$$Q(x_q, y_q, z_q) = l_1.\vec{k}$$

$$P(x_p, y_p, z_p) = T(x_t, y_t, z_t) - l_3.\vec{a}$$

De acordo com as equações (3.1), (3.2) e (3.3) a adição de uma junta 'R' é determinada pela capacidade do ponto terminal seguir a trajectória pretendida.

$$\vec{j}.\vec{PQ} \stackrel{?}{=} 0 \tag{3.1}$$

$$\vec{a}\cdot(\vec{k}\times\vec{QP}) \stackrel{?}{=} 0 \tag{3.2}$$

$$\vec{o} \cdot \vec{PQ} \stackrel{?}{=} 0 \tag{3.3}$$

Por exemplo, se o produto interno entre $\vec{j}$ e $\vec{PQ}$ não for zero, então é necessário introduzir uma junta de rotação 'R' entre os pontos *O* e *Q*, para corrigir o ângulo de torção. De modo idêntico, se o produto interno na equação (3.3) for diferente de zero é também necessário adicionar uma junta de rotação 'R' entre os pontos *P* e *Q* da trajectória. Por fim, se a equação (3.2) não apresentar um resultado nulo é necessário acrescentar uma junta de rotação entre os pontos *Q* e *S* ou entre os pontos *S* e *P*.

Com este procedimento e para as trajectórias amostradas, a configuração das juntas necessárias do manipulador pode ser determinada a partir dos submanipuladores 'RBBRBR' ou 'RBRBBR'. Das configurações seleccionadas, pode-se obter as equações directa e inversa da cinemática e da dinâmica. Estes dados serão usados para optimizar o comprimento dos elos no passo seguinte dos AGs.

3.3.3.3 Algoritmos genéticos utilizados

Os AG são codificados em *strings* binárias.

A probabilidade p_m é adaptativa (aumenta à medida que o processo decorre):

$$\mathrm{p_m} = p_{\max} \times \frac{t}{T};$$

Os limites dos parâmetros são:

$$x^j_{\min} = \bar{x}^j - (\bar{x}^j - \bar{x}^j_{\min}) \times e^{\eta \frac{t}{T}}$$

$$x^j_{\max} = \bar{x}^j - (\bar{x}^j - \bar{x}^j_{\max}) \times e^{\eta \frac{t}{T}}$$

e a resolução dos parâmetros é dada pela fórmula:

$$R^j = \frac{x^j_{max} - x^j_{min}}{2^{B^j} - 1}$$

onde:

t é o número da geração corrente;

T é o número de gerações;

η é a taxa de redução;

x_j é o parâmetro *j* da *string*;

$\bar{x}^j$ é o parâmetro *j* com melhor aptidão da *string*;

B^j é o número de *bits* do parâmetro *j*;

R^j é a resolução do parâmetro *j*.

3.3.3.4 Função objectivo e função de aptidão

A medida para avaliar as soluções deve ser independente do manipulador. Assim, é definida a capacidade de manipulação relativa (ou manipulabilidade relativa) por:

$$M_r = \frac{M}{\mathrm{f}_M}$$

onde:

f_M = função(comprimento$^{\mathrm{m}}$)

$$M = \sqrt[m]{JJ^T}$$

se *m* = 2 então:

$$\mathrm{f_M} = l^2 \text{ e } l_i = \sqrt{a_i^2 + d_i^2}$$

onde:

J é a matriz jacobiana da cinemática instantânea;

m é a dimensão do espaço da tarefa;

l representa o comprimento total do manipulador;

a_i é o comprimento de um elo;

d_i representa os parâmetros cinemáticos.

A função objectivo ao longo de uma trajectória é a dada por:

$$M_{rms} = \sqrt{\frac{\sum_{k}^{K_{\max}} \left[M_{RK}^{2} T_{amostragem} \right]}{T_{tarefa}}}$$

onde:

$k_{\max}$ é o número máximo de pontos de trabalho;

T_{tarefa} é o tempo de execução da tarefa;

$T_{\text{amostragem}} = T_{\text{tarefa}} \times k_{\max}$.

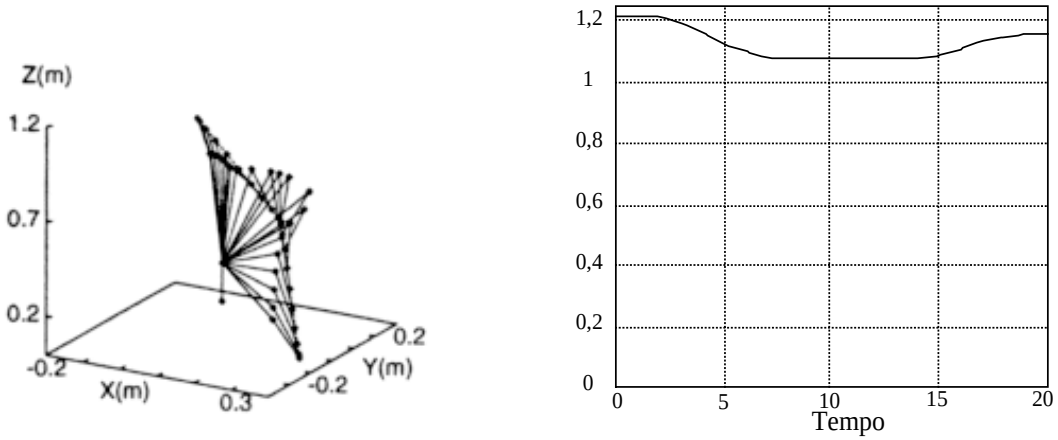
3.3.3.5 simulações

Na tarefa da tabela 3.1 são apresentadas as posições que o manipulador deve seguir. A trajectória desejada é calculada pela interpolação linear entre os pontos da tarefa. O intervalo inicial dos parâmetros é [0,1; 0,6] [m].

Tabela 3.1 Especificação da tarefa 1 [5].

N.º tarefa	p_x	p_y	p_z
1	0,5	-0,5	0,2
2	0,1	0,1	0,4
3	-0,3	0,4	0,6

Os resultados do manipulador óptimo estão representados na figura 3.11.



(a) (b)

Figura 3.11 Resultado da simulação da tabela 3.1 [5].

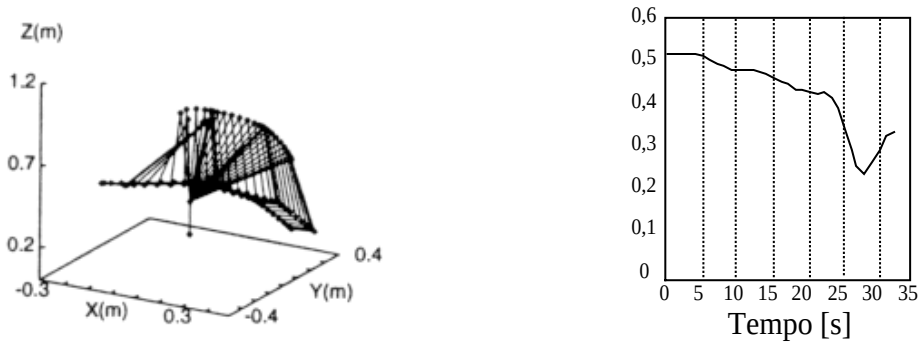
(a) Trajectória do ponto terminal; (b) capacidade de manipulação.

O tipo de manipulador obtido nesta simulação foi 'RBB' com 3 graus de liberdade e com os comprimentos dos elos de l_1 = 0,2 [m], l_2 =0,47 [m] e l_3 = 0,245 [m].

Para a tarefa representada na tabela 3.2 os resultados da simulação estão representados nas figura 3.12 e na figura 3.13. O manipulador obtido é 'RBBRBR' com 6 graus de liberdade e comprimentos dos elos: l_1 = 0,2 [m] e $l_2 = l_3$ = 0,49 [m].

Tabela 3.2 Especificação da tarefa 2 (pontos e orientação do ponto terminal) [5].

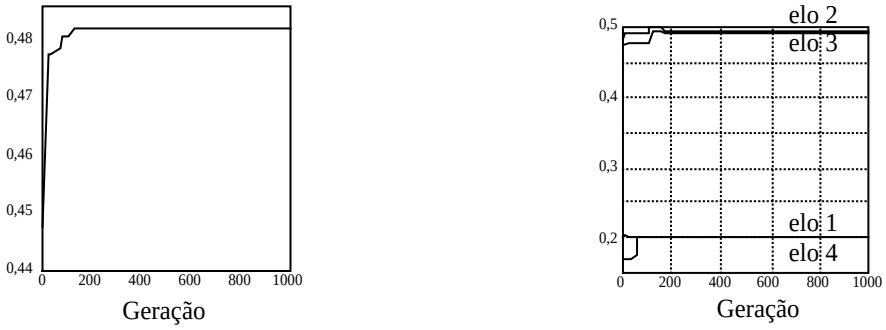
N.º da tarefa	p_x	p_y	p_z	θ_p	ψ_p	ϕ_p
1	0.2	0.4	-0.1	-300	200	300
2	0.1	0.3	0.1	-200	200	300
3	-0.1	0.2	0.2	-200	100	300
4	-0.2	-0.3	0.4	-200	100	200



(a) (b)

Figura 3.12 Resultado da tarefa 2 [5].

(a) Trajectória do ponto terminal; (b) capacidade de manipulação.



(a) (b)

Figura 3.13 Resultados da tarefa 2 [5].

(a) Aptidão máxima; (b) comprimento dos elos.

3.3.4 Projecto de um manipulador robótico através de AEs

3.3.4.1 Introdução

Chocron e Bidaud [15] descrevem um AE multicromossoma, para projectar um sistema robótico de acordo com a tarefa a executar com o menor número possível de módulos. O algoritmo utiliza uma codificação binária para os parâmetros de configuração e uma codificação real para a cinemática. Neste projecto considera-se um robô com uma base móvel e uma mão (ponto terminal), que pode ser construído com os seguintes mo-

delos: juntas (rotacionais (R), prismáticas (P) ou fixas (F)) e elos ([0; 1] m). O manipulador deve percorrer um conjunto de pontos num ambiente tridimensional.

Parâmetros de projecto para um segmento:

- orientação da junta (I_d, R_x, R_y, R_z);
- tipo de junta (R, P ou F);
- comprimento dos elos (0, 1/15,…,1 [m]).

A função de aptidão é dada por:

$$f = e^{-(a \times L + b \times A + c \times I + d \times O + g \times M)}$$

sendo *a*, *b*, *c*, *d* e *g* os pesos de cada critério;
e as funções:

L – distância linear;

A – distância angular;

I – módulos envolvidos;

M – capacidade de manipulação.

3.3.4.2 Algoritmo evolutivo multicromossoma

O algoritmo utiliza os operadores de selecção, cruzamento e mutação. A pesquisa da topologia e da configuração é feita simultaneamente pelo uso de um genoma para os dois subconjuntos de parâmetros (topologia e configuração). Deste modo, é evitada a interrupção da evolução global para determinar os parâmetros e acelerar o processo evolutivo. A inclusão de todo o genótipo num só cromossoma é pouco eficiente para *strings* de grande comprimento (neste problema eram necessários mais de 300 *bits*). Por exemplo, enquanto uma parte da *string* procura uma boa configuração, a outra parte (*i.e.* a topologia) pode estar a perder a sua qualidade. Para prevenir este tipo de problemas a informação é distribuída em várias *strings*. Cada *string* guarda informação de ligação que não é afectada pelo cruzamento global. Assim, cada configuração tem a sua própria *strings*, não sendo

influenciada pelo que acontece as outras *strings*. Cada *string* é submetida ao seu próprio cruzamento e é independente do que acontece a este operador de cruzamento. Além do mais, como a *string* é constituída naturalmente pela ligação dos parâmetros (*e.g*. configuração), pode ser avaliada localmente e pode ser feita qualquer consideração pelo operador genético com respeito à sua manipulação. De facto, todas as *strings* são ligadas globalmente através da topologia da *string*, mas algumas delas são mais dedicadas a partes em particular não relacionadas com a função objectivo.

A codificação é binária para a topologia e real para a posição da base e para as configurações(figura 3.14).

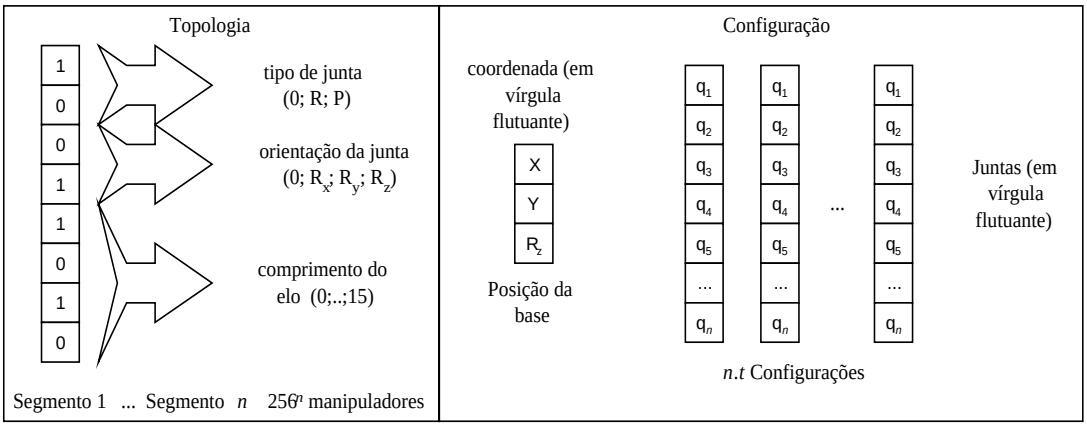


Figura 3.14 Genoma heterogéneo [15].

Estas decisões foram tomadas porque para uma topologia modular é melhor descrita por valores inteiros, além de que, valores reais evitam a perda de precisão nas configurações após as discretizações.

Cada segmento do manipulador é codificado com 8 *bits*: os dois primeiros indicam o tipo de junta, os dois seguintes dão a orientação relativa do eixo da junta (I_d, R_x, R_y, R_z), os quatro restantes indicam o comprimento do elo (figura 3.14).

A configuração da base e das juntas é codificada através de valores reais e colocados em *strings* separadas. Os valores reais podem variar no intervalo [0; 1], e são codificados com seis dígitos reais que permite 10^6 valores diferentes.

O espaço de pesquisa para a topologia inclui 256^n indivíduos e permite $256^n \times 10^{6\times n\times n\times t+18}$ soluções para um manipulador com n graus de liberdade e $n \times t$ metas.

3.3.4.3 Operadores genéticos

Relativamente aos operadores genéticos há a referir:

- O operador de selecção adopta uma amostragem estocástica (*remaider stochastic sampling with replacement*). A pressão da selecção é escolhida de acordo com a progressão da evolução, ou seja:

$$\rho = \rho_0 \times (F_m + \sigma_f)$$

$$f_m = f^{\rho}$$

onde:

ρ_o é a referência da pressão de selecção;

ρ é a pressão de selecção;

F_m é a média da função de aptidão da população;

f_m é a aptidão individual modificada;

σ_f é o desvio padrão da aptidão.

- O operador de cruzamento escolhido para a representação binária é do tipo uniforme devido à ineficiência do cruzamento de ponto simples para cromossomas de grande comprimento. Para a codificação real, o cruzamento uniforme consiste em trocar valores reais entre os dois pais. Para prevenir que a configuração de cruzamento seja

disruptiva, a probabilidade de permutação é modificada de acordo com a aptidão local de cada configuração para desactivar a migração dos parâmetros mais aptos, ou seja:

$$g_{\alpha i} = \frac{f_{\alpha i}}{f_{m\alpha}}$$

$$p_{\alpha\beta i} = \frac{0{,}5}{g_{\alpha i} \times g_{\beta i}}$$

onde:

$g_{\alpha i}$ é a aptidão local relativa, para o robô α e a configuração i;

$f_{m\alpha}$ é a aptidão local média, para o robô α ;

$p_{\alpha\beta i}$ é a probabilidade de permutação para a configuração i.

- O operador de mutação utilizado, na representação binária, é adaptativo e o ajustamento da probabilidade p_m é feito de acordo com cada função de aptidão:

$$p_m = \frac{p_{m\min}}{f}$$

onde:

$p_{m\min}$ é a probabilidade de mutação mínima;

f é a função de aptidão.

- A probabilidade de mutação, para a representação real, adapta-se à evolução pela modificação do desvio padrão, de acordo com a aptidão de cada indivíduo:

$$\sigma = \frac{\sigma_0}{f_i}$$

$$z = N(0, \sigma^2)$$

onde:

σ_0 é o desvio padrão mínimo do operador de mutação;

f_i é a função de aptidão da *string i*;

z é uma variável aleatória normal.

3.3.4.4 Simulação

Os parâmetros utilizados na simulação foram:

$\rho_o = 1$; $p_{cmin} = 0{,}6$; $\sigma_0 = 0{,}001$; $p_{mmin} = 0{,}001$; $\mu = 20$ e $T = 50$.

O problema a resolver é tridimensional e são utilizados três algoritmos para resolver o problema (AGs com dois níveis – TGA, AE multicromossoma – MEA e AE multicromossoma adaptativo – AMCA). A diferença entre estes algoritmos encontra-se na tabela 3.3.

Tabela 3.3 Diferenças entre os algoritmos utilizados [15].

Área	TGA	MEA	AMEA
Codificação	binário	binário/real	binário/real
Evolução	separada	simultânea	simultânea
Operadores	estática	estática	adaptativa

O manipulador utilizado e a respectiva tarefa encontram-se representados na figura 3.15. Existem quatro pontos a ser visitados e seis obstáculos. Todos os critérios são colocados a “1” e a distância de segurança é de 0,1 metros. Os parâmetros utilizados estão descritos acima e as topologias adoptadas incluem até oito graus de liberdade permitindo, assim, escolher manipuladores redundantes.

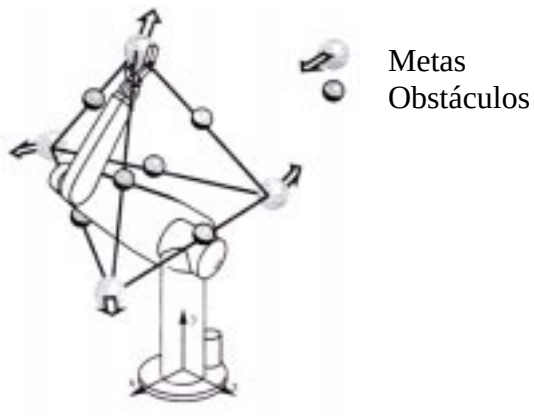


Figura 3.15 Especificação da tarefa [15].

A figura 3.16 (a) mostra o melhor indivíduo com os algoritmos TGA, MEA e AMEA. O algoritmo AMEA não só apresenta o melhor indivíduo como é cerca de 120 vezes mais rápido. Na figura 3.16 (b) encontra-se a evolução das médias de aptidão. Como se pode verificar as diferenças são mais notáveis e a evolução da população é superior com o algoritmo AMEA. Isto mostra que o algoritmo continua a melhorar toda a população enquanto não for encontrado um indivíduo mais apto. De facto, os algoritmos TGA e MEA mantêm uma diferença constante devido à mutação ser constante, perturbando a população. Contrariamente, o algoritmo AMEA diminui o ruído introduzido pelo operador de mutação com o aumento da aptidão.

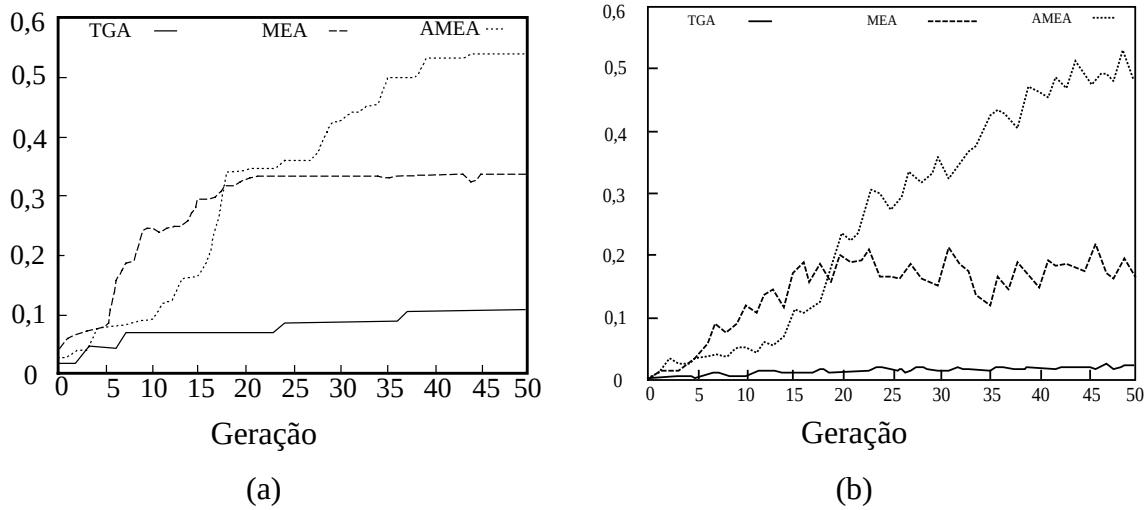


Figura 3.16 Comparação entre os três algoritmos [15].

(a) Comparação entre os melhores valores de aptidão;

(b) comparação entre os valores médios de aptidão.

3.3.5 Desenho de manipuladores através de um AGMPs

3.3.5.1 Introdução

Na publicação [6] apresenta um algoritmo para desenhar um manipulador que se adeque melhor a uma tarefa. Para esse fim utilizam-se AGs com várias populações (AGMPs).

As variáveis do projecto são:

- dimensão (comprimento dos elos);
- configuração (ângulos das juntas) e posição da base do manipulador.

O projecto é constituído por 4 fases:

- projecto;
- protótipo;
- planeamento;
- controlo.

O projecto de um manipulador planar com três graus de liberdade (figura 3.17), tendo como tarefa rodar uma manivela, pelo que a trajectória a descrever é circular (figura 3.18). Assim, interessa apenas a posição do ponto terminal do manipulador redundante.

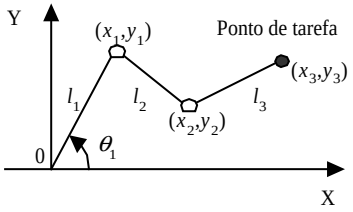


Figura 3.17 Manipulador planar com 3 graus de liberdade [6].

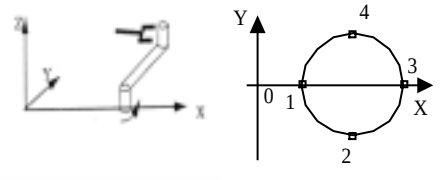


Figura 3.18 Tarefa a executar [6].

Para resolver o problema são usadas as variáveis (x_1, y_1) e (x_2, y_2) para as juntas 2 e 3. Se (x_2, y_2) é conhecido então o comprimento l_3 é automaticamente conhecido pela linha que liga a junta 3 e o ponto terminal.

A medida para escolher o manipulador óptimo (M) deve ser independente da escala. Nesta perspectiva utiliza-se a capacidade de manipulação relativa como medida:

$$M_r = \frac{M}{\mathrm{f}_M}$$

$$\mathrm{f}_M = \text{função}(\text{comprimento}^m)$$

$$M = \sqrt[m]{JJ^T}$$

$$\text{se } m = 2 \text{ então } \mathrm{f_M} = l^2 \text{ e } l_i = \sqrt{a_i^2 + d_i^2}$$

onde:

J é a matriz jacobiana da cinemática;

m é a dimensão do espaço da tarefa;

l é o comprimento total do manipulador;

a_i é o comprimento de um elo;

d_i representa os parâmetros cinemáticos.

As restrições utilizadas no problema são:

- capacidade de alcançar (RC) (*i.e.* atingir o ponto pretendido);
- heurísticas (RH) (evitar certas configurações, *e.g.* elo com comprimento nulo);
- limitações das juntas (RLJ);
- mudanças dos ângulos das juntas (MAJ) (*i.e.* evita mudanças bruscas);
- especificas da tarefa (RET).

3.3.5.2 Operadores genéticos e representação

Os operadores genéticos utilizados são: reprodução, cruzamento e mutação (normais).

A representação das *strings* é da forma (x_1, y_1, x_2, y_2) em que cada variável é codificada com 5 *bits*.

Para o projecto do manipulador planar, o número de pontos varia de acordo com o número de pontos da tarefa (considerando que l_i e θ_i são utilizados como variáveis de projecto):

$$(n-2) \times p + n + 2$$

onde:

n é o número de graus de liberdade;

p é o número total de pontos da tarefa.

O espaço de pesquisa cresce exponencialmente com o número de variáveis. Isto implica que a qualidade da solução diminui rapidamente. Para evitar este problema, são utilizadas p funções de optimização correspondentes a cada ponto da tarefa. Assim, são

projectados *p* manipuladores para *p* pontos de tarefa. A vantagem desta implementação paralela é que o número de variáveis para cada função é fixada em $p = 1$ na equação anterior. Esta vantagem obtém-se à custa da introdução de duas restrições adicionais: a restrição RL – os comprimentos dos elos, dos *L* manipuladores, devem ser idênticos, e a restrição RBP – a base dos *L* manipuladores deve ter a mesma posição. Estas restrições são necessárias porque, como é obvio, se pretende que um manipulador seja o mesmo em todos os pontos.

3.3.5.3 Função de aptidão

A função de aptidão é formada pela função objectivo e pelo valor de ajuste das restrições de ligação (que é obtida pela comparação dos indivíduos das outras populações):

$$f_{ij} = M_{rij} + RET_{ij} + RC_{ij} + RLJ_{ij} + RL_{ij} + RBP_{ij} + MAJ_{ij}$$

onde: *i* é o indivíduo da população *j*;

M_r é a capacidade de manipulação relativa;

RET são as restrições especificas da tarefa;

RC é a capacidade de alcançar;

RLJ são as restrições das limitações das juntas;

RL diz respeito as restrição dos comprimentos dos elos;

RBP é a restrição relativa à base do manipulador;

MAJ são as restrições dos ângulos das juntas.

A função de aptidão pode incluir pesos nas subfunções. Cada subfunção atribui um valor ao manipulador de acordo com o desempenho deste nessa área.

Por exemplo, quando, uma população tem uma função objectivo bastante elevada e quando a média da população tiver uma restrição RL bastante diferente das restantes, então o valor da restrição RL vai neutralizar o valor alto da função objectivo.

3.3.5.4 Projecto progressivo

O projecto progressivo reduz sucessivamente o número de variáveis e o espaço de pesquisa, enquanto aumenta o número de pontos finitos do espaço da tarefa. O projecto progressivo tem os seguintes passos (figura 3.19):

- projecto cinemático – é deduzido o valor óptimo das variáveis de projecto para o comprimento dos elos e são determinados os ângulos e a posição da base;
- protótipo – são optimizadas as variáveis dos ângulos e da posição da base;
- planeamento – é planeada uma trajectória com mais pontos do percurso. Se alguma restrição não é satisfeita o algoritmo regressa ao primeiro passo (projecto cinemático);
- controlo cinemático – consiste em continuar a interpolar novos pontos da tarefa e para justificar o projecto e o planeamento dos 3 primeiros passos. Se for detectada uma violação de uma restrição volta-se ao primeiro passo.

3.3.5.5 Resultados

A figura 3.20 mostra a variação da melhor função de aptidão (passo 1). A figura 3.21 apresenta a variação do comprimento total para os quatro pontos da tarefa. O resultado do protótipo é mostrado na figura 3.22 (passo 2). A figura 3.23 e a figura 3.24 apresentam, respectivamente o resultado do passo 3 e o resultado do algoritmo.

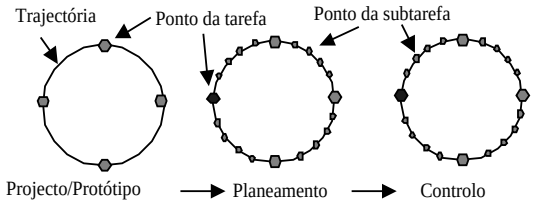


Figura 3.19 Projecto progressivo [6].

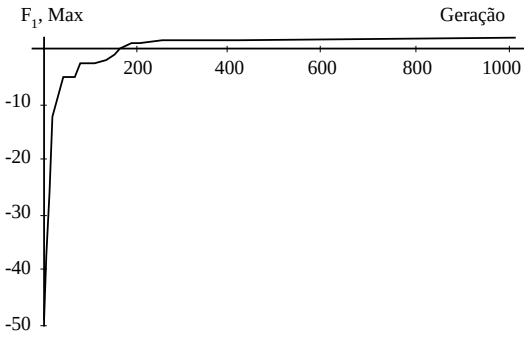


Figura 3.20 $F_{1,max}$ v.s. Número de gerações [6].

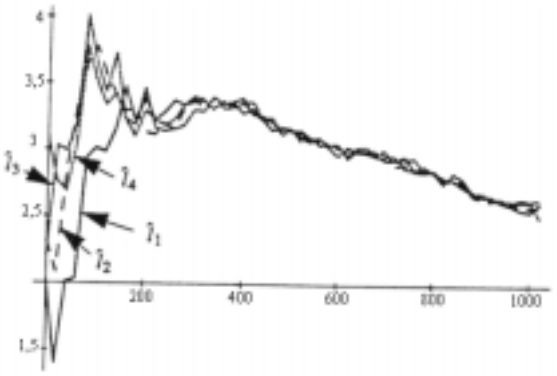


Figura 3.21 Comprimentos dos elos v.s. número de gerações [6].

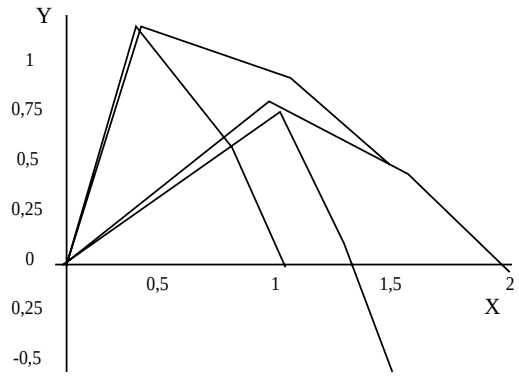


Figura 3.22 dimensão óptima e a base dos pontos da tarefa [6].

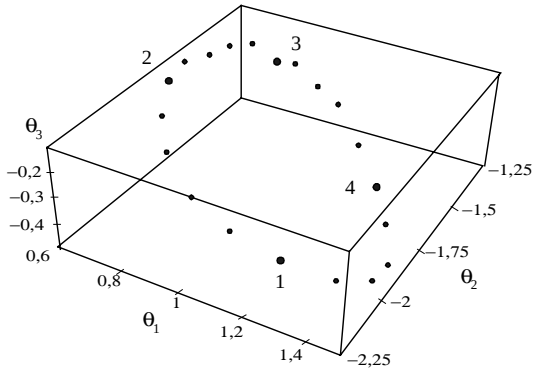


Figura 3.23 Resultados do protótipo e do planeamento no espaço das juntas [6].

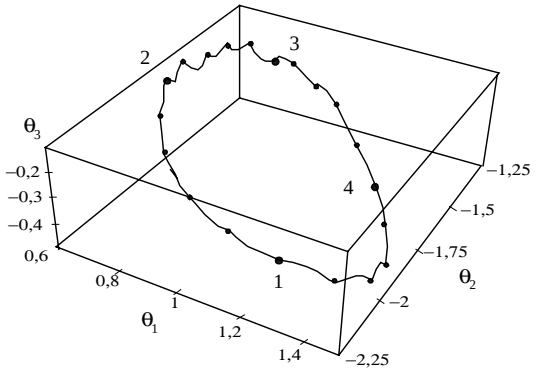


Figura 3.24 Resultado do controlo no espaço das juntas [6].

3.4 Estudo da locomoção de robôs

3.4.1 Introdução

Os AGs são bastantes utilizados no estudo da locomoção através de robôs com pernas [7],[8],[9],[18],[12],[19],[20]. Um sistema robótico com pernas deve ser autónomo, nomeadamente do ponto de vista da energia consumida. Nesta parte são apresentadas algumas aplicações de AGs a sistemas robóticos de locomoção.

3.4.2 Locomoção do “Stiquito”

3.4.2.1 Introdução

Gray Parker *et al.* [9] tentam optimizar o modo de andar de um robô (Stiquito) de seis pernas. São utilizados AGCs para desenvolver um modo suave de caminhar. O robô tem dois actuadores unidireccionais por perna: um para a erguer e o outro para fazer a operação contrária.

3.4.2.2 Representação

O cromossoma tem uma parte que caracteriza o modo ciclo de andamento. O comprimento do cromossoma é fixo:

$(C\ I\ ((A\ R))\ ((A\ R)_1\ (A\ R)_2\ (A\ R)_3\ (A\ R)_4\ (A\ R)_5\ (A\ R)_6\ (A\ R)_7\ (A\ R)_8\ (A\ R)_9\ (A\ R)_{10}\ (A\ R)_{11}\ (A\ R)_{12}))$

onde:

A representa um activador;
C representa um coordenador;
I representa um inibidor;
R representa um repetidor.

A parte inicial da *string* é composta por:

- Coordenadores globais (12 *bits*) que coordenam o movimento das pernas (2 *bits* por perna). O primeiro *bit*, coordenador *back_down*, (de cada perna) indica quando a perna esta no chão ou no caso contrario quando está a deslocar-se para trás. O segundo *bit* (coordenador *forward_up*) assegura que a perna se move em frente, se estiver levantada.

- Inibidores globais (15 *bits*) onde cada *bit* representa um par de pernas. Se o *bit* estiver a '1' não é permitido que essas duas pernas se movam ao mesmo tempo (*i.e*. inibe a activação da perna com o número maior).
- Gene inicial (12 *bits*) necessário para o robô transitar do estado em repouso para o estado de caminhar periódico.
- Repetições inicial (8 *bits*) que indica o número de vezes que o gene inicial se repete.

A parte iterativa do cromossoma é composto por 12 genes (tarefa) e pelo número de repetições (8 *bits*) de cada gene.

3.4.2.3 Operadores genéticos

A probabilidade de selecção de um cromossoma é dada em função do valor de aptidão desse mesmo cromossoma. O melhor cromossoma é sempre seleccionado para a próxima geração.

Os operadores utilizados são os habituais neste tipo de algoritmo (selecção, cruzamento e mutação).

3.4.2.4 Simulações e resultados

Foram realizados 5 testes com o fim de verificar se os AGCs podiam gerar modos de locomoção razoáveis para este modelo. Estes resultados foram verificados experimentalmente no robô.

O número de gerações adoptado foi de 1000 com uma população de 64 cromossomas.

A numeração das pernas é a seguinte:

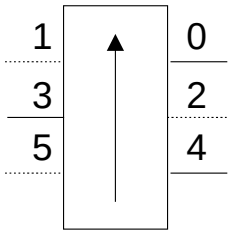


Figura 3.25 Numeração das pernas do “Stiquito” [9].

A solução obtida (cromossoma) é apresentada de seguida.

O número de movimentos da secção inicial do cromossoma é nula (*i.e.* não é necessária esta parte).

A parte da secção iterativa do cromossoma tem cinco genes activos. As activações e os movimentos inibidores e coordenadores são os seguintes:

- com as pernas (0, 3, 4) apoiados no chão, levantar as pernas (1, 2, 5), o número de repetições deste movimento é de cinco;
- com as pernas (0, 3, 4) no chão, baixar as pernas (1, 2, 5), o número de repetições deste movimento é de dois;
- passo equivalente ao anterior;
- com as pernas (1, 2, 5) assentes no chão levantar as pernas (0, 3, 4), com 4 repetições;
- com as pernas (1, 2, 5) no chão, baixar as pernas (0, 3, 4), com 5 repetições.

Foi obtido um modo de locomoção com dois ciclos de nove activações.

3.4.3 Geração de locomoção bípede usando um método hierárquico de geração de trajectórias

3.4.3.1 Introdução

Arakawa e Fukuda [19] propõem um método hierárquico para gerar um movimento natural de locomoção, optimizando a energia dispendida.

O método pode ser dividido em duas camadas:

- A camada PE que gere a configuração interpolada da locomoção bípede do robô;
- A camada dos AGs que selecciona a configuração interpolada efectiva da locomoção bípede do robô, com o objectivo de minimização da energia total dos actuadores.

3.4.3.2 Representação das *strings*

Na camada PE um indivíduo representa uma configuração interpolada (ângulos das pernas, conforme na figura 3.26) da locomoção bípede do robô, sendo cada ângulo codificado através de um número real.

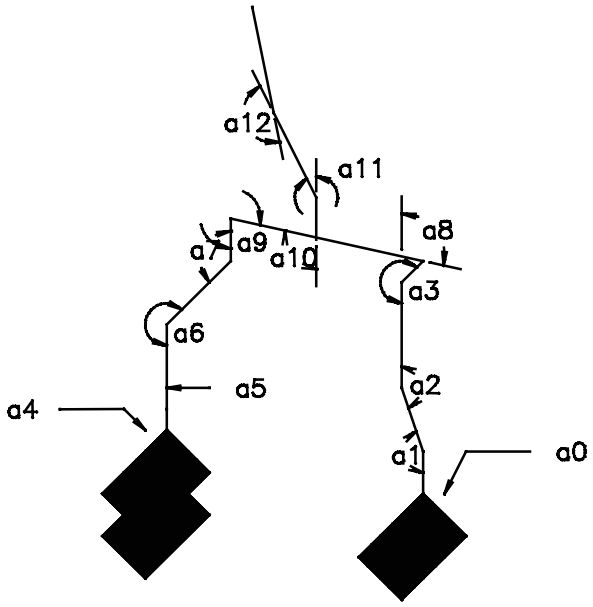


Figura 3.26 Representação dos ângulos das juntas [19].

Na camada AG um indivíduo é constituído por duas partes: ângulos do robô e um *bit* que indica quando a configuração é ou não efectiva.

O Algoritmo do método hierárquico é apresentado em seguida:

```
Início
   Inicialização
   repetir
      PE_Mutação
      PE_Avaliação
      PE_Selecção
      Tranferir_configuração
      AG_Cruzamento
      AG_Mutação
      AG_Avaliação
      AG_Selecção
```

```
   ate codição_de_conclusão_verificada
fim do algoritmo
```

Algoritmo 3.2 Algoritmo hierárquico.

No início do algoritmo os indivíduos da população são gerados aleatoriamente. De seguida são executados os operadores da camada PE. Numa terceira fase, a configuração interpolada da locomoção bípede do robô é transferida, através do procedimento `Transferir_configuração`, para formar os indivíduos da camada AG. Por último, é feita a optimização através dos operadores do AG.

3.4.3.3 Operadores genéticos

Para os operadores genéticos adoptados há a apontar:

- O procedimento `PE_Selecção` que selecciona os melhores μ elementos da população $\mu + \lambda$.

- O procedimento `PE_Mutação` que cria λ ($\lambda = \mu$) descendentes a partir da população inicial (μ), através do operador de mutação:

$$x_{n+i,j} = x_{i,j} + N(0, a \times \frac{F_{max} - \mathrm{f}_i}{F_{max} - F_{min}} + b)$$

onde:

f_i é a função de aptidão do indivíduo *i*;

F_{max} é o valor de aptidão máximo entre todos os indivíduos da população;

F_{min} é o valor de aptidão mínimo entre todos os indivíduos da população;

a e *b* são constantes.

- O procedimento `AG_Cruzamento` que executa o cruzamento (simples) de elementos da população.

- O procedimento `AG_Mutação` que executa a mutação normal sobre indivíduos da população.

- O procedimento `AG_Selecção` que utiliza o método proporcional estocástico, onde o melhor indivíduo é sempre passado para a geração seguinte.

3.4.3.4 Função de aptidão

A função de aptidão do AG tem como objectivo minimizar a energia consumida. Assim, vem:

$$\mathrm{E}_{ag}(x) = k_{ag1} \times \left(\int_0^T \tau^T \dot{\theta} dt \right)^2 + k_{ag2} \times \int_0^T \mathrm{C}_{ag}\, dt$$

Os ângulos θ são retirados da seguinte fórmula:

$$\theta_i = \mathrm{spline}_i(x_j, \mathrm{t}) \quad (j = 1, 2, \ldots, \mathrm{n})$$

onde i é a junta e x_j são os pontos de interpolação da função spline.

Para a função de aptidão do PE vem:

$$\mathrm{E}_{pe}(\theta) = k_{pe1} \times \min_{\theta^* \in \mathrm{B}} \{ \left| \theta - \theta^* \right| \} + k_{pe2} \times \int_0^T C_{pe} dt$$

onde:

k_{ag1}, k_{ag2}, k_{pe1} e k_{pe2} são os pesos de cada componenete;

θ^* é a configuração interpolada na melhor solução de aptidão;

B é o conjunto das configurações interpoladas efectivas;

C_{pe} e C_{ag} são as funções de restrição tais que:

$$\begin{matrix} C_{ga} \\ C_{pe} \end{matrix} = \begin{cases} 0 & \text{se a restriçao é satisfeita} \\ c & \text{caso contrario} \end{cases}$$

3.4.3.5 Simulação

Foi efectuado um teste, num piso plano, com os seguintes parâmetros:

Tabela 3.4 Parâmetros da simulação [19].

Passo	0,30[m]
Tempo que demora um passo	5,0[s]
Tamanho da população	500
Probabilidade de cruzamento do AG	0,60
Probabilidade de mutação do AG	0,10
Número de gerações	100

A figura 3.27 mostra a trajectória calculada para a locomoção bípede de um robô e a figura 3.28 mostra a energia total gasta nos actuadores. Os resultados aplicados ao robô real foram idênticos.

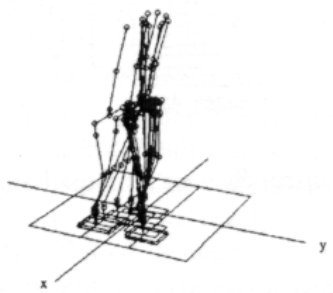


Figura 3.27 Trajectória da locomoção bípede do robô [19].

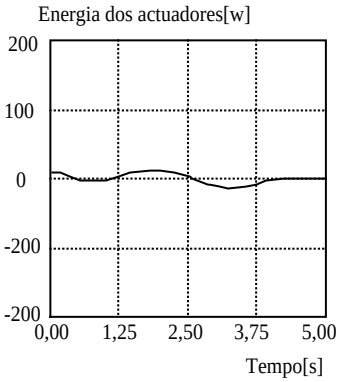


Figura 3.28 Energia consumida pelos actuadores [19].

3.4.4 Metodologia auto-planeada

3.4.4.1 Introdução

Farritor e Dubowsky [18] apresentam um método para conceber planos à medida que o robô se desloca, a partir de um conjunto de acções (inventário) predefinidas. Os planos de acção incluem a navegação, a aquisição de informação através de sensores e incluem instruções das tarefas a executar. Um plano é constituído a partir de acções, realizáveis fisicamente, e de módulos de acções, que são agrupados para produzir um plano satisfatório.

3.4.4.2 Aproximação

Um plano é válido quando permite ao robô completar os objectivos da tarefa sem violar qualquer restrição física do robô ou da tarefa (*e.g.* saturação dos actuadores, estabilidade estática, energia consumida, restrições cinemáticas, obstáculos).

O plano de acções é construído a partir de módulos de acções consecutivas (*e.g.* figura 3.29). Se estes módulos forem construídos correctamente o robô executa a tarefa. Um exemplo simples de um plano de acções é apresentado na figura 3.30.

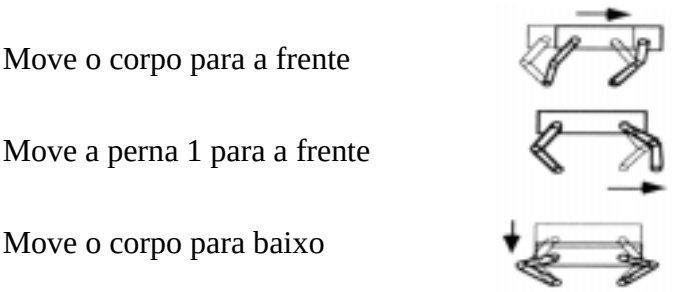


Figura 3.29 Modulo de acção simples [18].

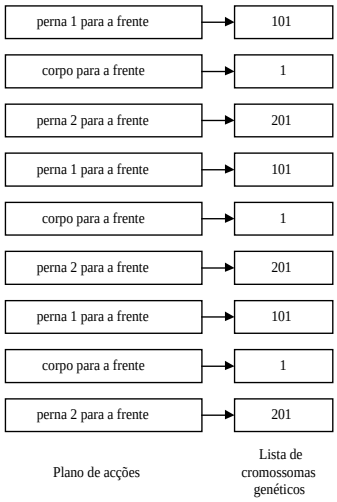


Figura 3.30 Representação da *string* [18].

Para pesquisar eficientemente o espaço, o algoritmo utiliza um método hierárquico que reduz o espaço de acções através de filtros de módulo e de tarefa.

O processo de selecção hierárquico consiste:

- em retirar os módulos de acções que não digam respeito à tarefa; O segundo passo consiste em remover módulos de acções que não pertençam à configuração deste robô;
- no uso de AGs para determinar o melhor plano de acções.

3.4.4.3 Estudo de um caso

Foi realizada uma experiência com um robô constituído por quatro pernas e por dois manipuladores. O corpo do robô tem dimensão $8" \times 4" \times 4"$ e o comprimento de cada membro é de 9".

A simulação executa um plano de acção para determinar se o robô acompanha a tarefa sem qualquer violação de restrições físicas. O plano é executado até ao fim ou até ao ponto em que uma restrição deixa de ser satisfeita.

Uma *string* representa um plano de acções (figura 3.30) e utiliza os operadores de cruzamento e mutação simples.

O processo de selecção explora os fundamentos do sistema, ou seja, os dados adquiridos pelos sensores e o conhecimento da tarefa e do ambiente, a fim de produzir planos o mais eficiente possível.

A função de aptidão é dada por:

$$f = \alpha_1 \times |D| - \alpha_2 \times P + \alpha_3 \times \delta$$

onde:

D é a distância entre o robô e o ponto destino após a execução da simulação;

P é a energia consumida pelo robô;

δ indica se o alvo foi atingido ($\delta = 1$ quando o alvo é atingido, $\delta = 0$ no caso contrário);

α_i (i = 1, 2 e 3) são os pesos de cada componente da função de aptidão.

A dimensão do espaço de pesquisa é dado por $D = N^m$, onde D é o número possível de planos de acções, N é o número de módulos de acção após a redução do inventário e m é o número de módulos de acção usados no plano de acções.

3.4.4.4 Resultados obtidos

Depois do espaço de pesquisa ser substancialmente reduzido, são aplicados os AGs com uma população de 40 *strings* para um número 8000 de gerações. Para levar o robô ao ponto desejado são necessários 189 módulos de acções.

3.4.4.5 Horizonte do robô

O reconhecimento do ambiente num pequeno raio à volta do robô, permite-lhe que este tenha um carácter mais reactivo. Para determinar qual o comprimento óptimo deste raio foi executada uma tarefa de locomoção de “24" em linha recta”. Cada vez que seja encontrada uma solução melhor é guardado o comprimento dessa solução. Os resultados da simulação encontram-se representados na figura 3.31. Onde se pode ver que a maior parte dos melhoramentos envolvem um número de módulos de acções inferiores a cinco.

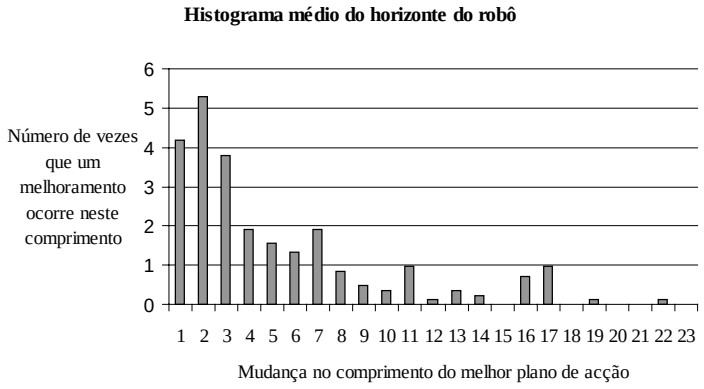


Figura 3.31 Histograma médio do horizonte do robô [18].

Para este método desenvolver um bom plano de exploração, o problema de planeamento deve ser considerado a níveis diferentes. Deve ser usado um inventário de módulos de acção que contenham módulos de alto nível. Para resolver problemas de alto nível, a tarefa é subdividida em submetas e os subproblemas são resolvidos a baixo nível.

3.4.4.6 Aprendizagem do robô

Para sublinhar esta característica do método, o robô executa a tarefa repetitiva: "trepar sequencialmente 4" passos". A pesquisa conduziu à criação de um plano de 514 módulos de acções tendo, para tal, sido executadas 1000 gerações de acções requerendo 80 gerações.

Após o desenvolvimento do plano foi analisada a existência de padrões repetidos. Foram encontrados padrões de módulos de 12 acções repetidos em cada um dos quatro passos, sugerindo assim que tenha ocorrido aprendizagem. Esses módulos de 12 acções foram agrupados para produzir módulos de alto nível que foram então adicionados ao inventário. A tarefa foi repetida utilizando este inventário aumentado. Os resultados obtidos (figura 3.32) permitem concluir que a convergência do algoritmo com o inventário de módulos de alto nível é mais rápida e o valor de aptidão superior. Assim, os módulos de alto nível devem ser incorporados no inventário.

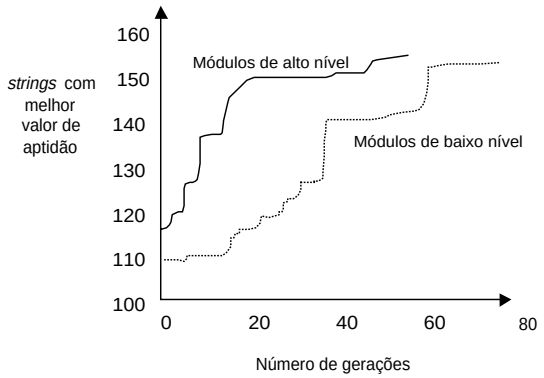


Figura 3.32 Convergência do AG [18].

3.5 Planeamento de trajectórias para manipuladores robóticos

3.5.1 Introdução

Os AGs são também utilizados no planeamento de trajectórias de manipuladores robóticos [1],[2],[3],[13],[14],[17],[22].

As trajectórias são compostas por uma sequência de deslocamentos do braço robótico. Uma trajectória pode ser vista como uma sequência de pontos nos quais o ponto terminal deve passar. Como resultado da movimentação do braço, ao longo dos pontos discretos obtém-se uma trajectória continua. Assim, a optimização da trajectória de um robô significa a identificação da combinação óptima e do número de posições intermédias.

Neste subcapítulo são apresentadas alguns métodos para gerar trajectórias de manipuladores robóticos.

3.5.2 Planeamento de trajectórias

3.5.2.1 Introdução

Neste secção é apresentado um algoritmo capaz de gerar trajectórias para um manipulador robótico redundante [22]. Este robô é constituído por três elos e três eixos planares.

3.5.2.2 Representação da trajectória

Uma trajectória de *n* pontos é representada na seguinte forma (*string*):

$$\{(A_1, A_2, A_3)_1, (A_1, A_2, A_3)_2, \ldots, (A_1, A_2, A_3)_{n-1}, (A_1, A_2, A_3)_n\}$$

onde um gene é constituído por o terno $(A_1, A_2, A_3)_j$ e A_i é o ângulo correspondente ao ângulo da junta $i = 1, 2, 3$.

Para este tipo de representação devem ser salientados os seguinte pontos:

- o comprimento das *strings* não deve ser fixo, para permitir que uma trajectória possa ser descrita por um vector com um número variável de movimentos;
- sendo as *strings* de comprimento variável, a posição dos genes deixa de ter importância;
- o número de trajectórias diferentes é elevado;
- a representação contém subestruturas repetitivas que não devem ser separadas.

Como o espaço de pesquisa é muito grande, é implementado um mecanismo com a finalidade de reduzir o referido espaço (*i.e.* o número de valores que os genes podem tomar). Numa primeira fase o algoritmo começa a pesquisa num intervalo grande, onde a diferença entre dois alelos com valores consecutivos é grande (ou seja, com uma resolução pequena do espaço de pesquisa). Nesta fase é encontrado o valor de base para o passo seguinte.

Numa segunda fase a resolução dos valores dos parâmetros é aumentada, estando a população inteira sujeita a valores de probabilidade de mutação elevados. A finalidade de utilizar uma taxa de mutação elevada, no início de cada fase, é aumentar a diversidade da população na vizinhança do valor de base. Este processo continua até ser atingida a resolução desejada.

As resoluções devem ser suficientemente pequenas de forma a permitir a identificação de todos os óptimos locais.

3.5.2.3 Operadores genéticos

O operador de cruzamento usado é o cruzamento de segregação/análogo. Assim, o local de cruzamento só pode ocorrer entre genes, sendo a posição da segunda *string* a mais análoga do ponto de cruzamento da primeira *string*. O critério de similaridade fenotípica no cruzamento análogo adopta a distância Euclidiana mínima entre os pontos ter-

minais do manipulador. Para implementar este critério é necessário uma função de projecção entre o espaço fenótipo e o espaço genótipo.

O operador de mutação é implementado ao nível dos ângulos, isto é, a mutação pode ocorrer em qualquer ângulo (dentro de um gene). O operador segue uma lei uniformemente distribuída para um gene e, dentro deste, com importância inversa à ordem do elo ($p_{m1} = 4 \times p_m$, $p_{m2} = 2 \times p_m$, $p_{m3} = p_m$; p_{mi} – probabilidade de mutação do elo *i*). A mutação adiciona um valor inteiro arredondado ao ângulo em questão. Tipicamente o valor a adicionar é dado pela lei de distribuição de Poisson com $\lambda = 0{,}3$.

Os operadores de adição e remoção são os responsáveis pela gestão do comprimento das *strings*. O operador de remoção elimina um *gene* escolhido aleatoriamente da *string*. Enquanto que o operador de adição (duplicado, relacionado, aleatório) insere um gene (igual ao gene que se encontra na posição anterior, igual à média dos genes vizinhos, aleatório) na *string*.

3.5.2.4 Função de aptidão

A função objectivo é dada pela expressão:

$$\text{f.o.} = \int_{trajectoria} |desvio| + \left| E_{desejado} \right|_{inicio} + \left| E_i - E_{desejado} \right|_{fim}$$

A primeira parcela da equação corresponde ao desvio acumulado entre a trajectória representada pela *string* e a trajectória pretendida. As segunda e terceira parcelas indicam, respectivamente, a discrepância entre a trajectória desejada e a trajectória realizada no início e no fim.

A função de aptidão é:

$$f = 100 \times e^{-\text{f.o.}}$$

o factor de 100 é para o resultado ser expresso em percentagem.

3.5.2.5 Mecanismo de selecção

No mecanismo de selecção o descendente substitui o pior dos pais na população no caso de ter um valor de aptidão superior.

3.5.2.6 Resultados

Foram efectuados vários testes com um robô de três elos. Pretende-se que o robô siga uma trajectória com 4,2 [m] de comprimento, com o ponto de partida e de chegada, respectivamente, em (0, 0) [m] e (–1, 1) [m]. Os elos são livres de rodar em torno das suas juntas, mas estão restringidas ao plano vertical. O número de elementos da população é de 100 elementos, a probabilidade de cruzamento é de 1, a probabilidade de remoção é de 0,05, a probabilidade de mutação segue uma lei de Poisson com $\lambda = 0{,}3$ e a probabilidade de adição é de 0,06. Inicialmente cada elo pode estar posicionado em dez pontos diferentes. Posteriormente a resolução é aumentada de acordo com a diversidade da população (diferença entre o erro médio e o melhor erro da população). Em média ocorre um aumento da resolução entre 400 e 500 gerações.

Quando a resolução é aumentada (fase de duplicação), a probabilidade de mutação tem o valor 1, mas é incluída a melhor trajectória na nova geração.

Na figura 3.33 esta representada a melhor *string* para 5 experiências com idênticos parâmetros. Os resultados são diferentes devido à sequência de números aleatórios ser distinta. Este comportamento instável da trajectória do AG pode ser explicado quando o erro da melhor trajectória é examinado. Os erros mostram uma pequena diversidade. Pode-se concluir que o algoritmo é bastante robusto e repetitivo ($\sigma = 0{,}20$) pois, apesar das trajectórias no final das experiência sejam diferentes, elas têm uma qualidade semelhante.

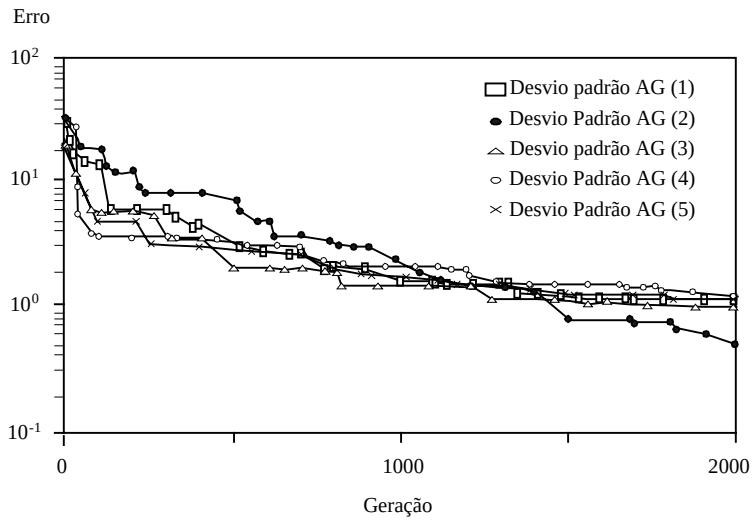


Figura 3.33 Solução da melhor *string* para cinco simulações iguais [22].

A figura 3.34 apresenta o erro médio das cinco experiências. Verifica-se um comportamento semelhante nas diferentes simulações, que pode ser visto como uma indicação de robustez. Podem também ser observadas duas características:

- as curvas não são nem suaves nem monótonas (como acontece na maioria dos AGs);
- as curvas da função de aptidão média exibem um comportamento similar mesmo que sigam trajectórias diferentes.

Os "saltos" drásticos no valor de aptidão médio são o resultado do mecanismo especial de selecção introduzido. Deve ser notado que, ignorando os saltos, a curva tem um comportamento de convergência característico. A explicação da existência dos saltos é fornecida no parágrafo seguinte.

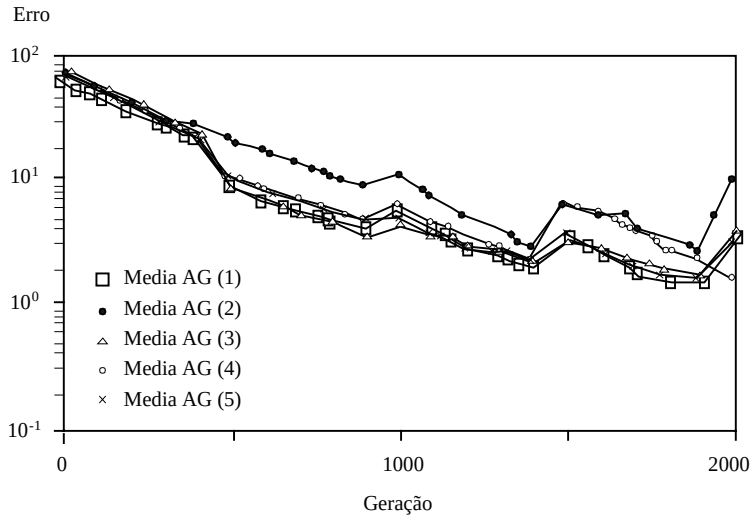


Figura 3.34 Erro médio das cinco simulações [22].

A figura 3.35 apresenta a aptidão média da população e a aptidão do melhor elemento da população de uma das experiências. As variações mais acentuadas (indicadas por setas na figura) ocorrem quando há um aumento na resolução (das juntas) do problema. As variações do método nesses locais são o resultado de dois processos contraditórios: um reduz a aptidão média e o outro aumenta a aptidão média. A diminuição, do erro médio, resulta da menor diversidade das trajectórias face ao aumento do número das melhores trajectórias. O aumento, do erro médio, resulta da probabilidade de mutação ser elevada aquando o aumento da resolução do problema. Este efeito é mais evidente na parte inicial da pesquisa. Assim, o erro médio tende a diminuir na primeira alteração na resolução do problema, mas aumenta nas alterações seguintes da resolução.

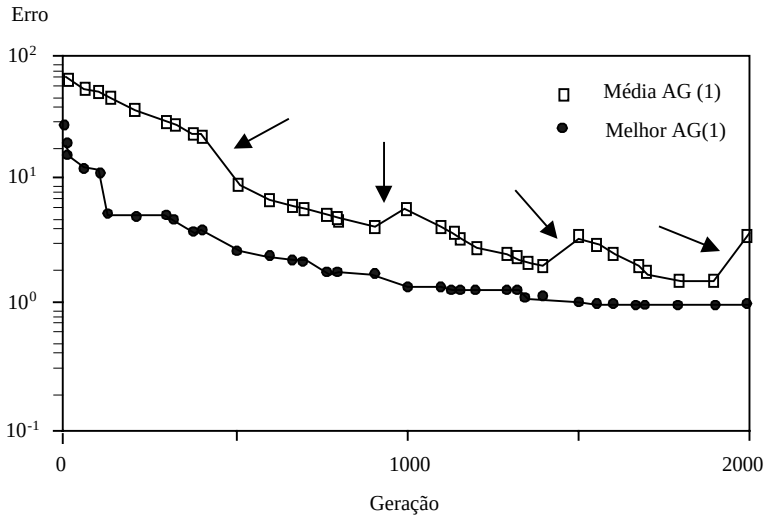


Figura 3.35 Erro médio e melhor erro de uma das cinco experiências [22].

3.5.3 Planeamento de trajectória utilizando um AG com vírus

3.5.3.1 Introdução

Kubota *et al.* [14] desenvolveram um vírus evolutivo/AG (VEAG) (*virus--evolutionary genetic algorithms*) para o planeamento hierárquico de trajectórias de um manipulador redundante.

O método de planeamento de trajectórias hierárquico é composto por duas camadas:

- *Geração de posições* – gera posições livres de colisões para o manipulador redundante no espaço de trabalho, como pesquisa local. Quando a trajectória satisfaz a aspiração de nível, é enviada à camada de geração de trajectórias.
- *Geração de trajectórias* – gera uma trajectória livre de colisões combinando algumas posições intermédias (fornecidas pela camada de geração de posições), como pesquisa global. Isto é, a camada de geração de trajectórias gera outras posições intermédias na solução do melhor candidato.

Para gerar uma trajectória livre de colisões do manipulador redundante, o algoritmo VEAG usa o planeamento de trajectórias hierárquico, baseado na cinemática directa.

O método hierárquico de planeamento de trajectórias executa simultaneamente a geração de posições e a geração de trajectórias.

A população é composta por uma população de indivíduos (hospedeiros) e uma população de vírus. A coevolução das duas populações permite obter uma solução óptima de forma rápida. O comprimento da *string* hospedeiro é fixa enquanto que para a *string* vírus é variável. A mutação é adaptativa de modo a melhorar o desempenho da solução.

O Algoritmo VEAG (algoritmo 3.3) é baseado no modelo AGs regime permanente onde é executado apenas um cruzamento em cada geração.

```
Inicialização
   repetir
     Selecção
     Cruzamento
     Mutação
     Virus_infecção
     Reposição
   até condição_de_conclusão_verdadeira
fim do algoritmo
```

Algoritmo 3.3 Algoritmo VEAG.

```
Procedimento Vírus_infecção
   repetir
      Seleccinar_hospedeiro
      Transcrição_inversa
      Avaliação
   até life_{i,t+1} < 0
```

```
   Transdução (selecciona_hospedeiro) vírus_i
retornar
```

Algoritmo 3.4 Procedimento Vírus_infecção.

O procedimento `Inicialização` faz a inicialização aleatória da população de hospedeiros e cria a população de vírus através de sub-*strings* aleatórias da população de hospedeiros. A função `Vírus_infecção` (algoritmo 3.4) gera novos hospedeiros a partir dos vírus. Após a infecção, os novos indivíduos substituem os seus progenitores se forem mais aptos.

3.5.3.2 Representação

- Camada de geração de posições

Uma *string* de hospedeiros inclui todas as juntas do manipulador e é codificada com o alfabeto {0, 1}. A *string* do vírus inclui os símbolos {0, 1, #} e tem o mesmo comprimento que a *string* hospedeiro.

- Camada de geração de trajectórias

Uma posição é expressa por um conjunto de ângulos das juntas do manipulador. Para evitar colisões, o algoritmo VEAG gera uma posição intermédia apropriada que optimiza a função objectivo, baseada na distância entre o manipulador e os obstáculos. Esta distância é medida utilizando o conceito do pseudo-potencial num espaço com $N \times N \times N$ células.

Nesta camada existe uma série de soluções candidatas (incluindo ângulos das juntas) de posições intermédias. A figura 3.36 mostra a *string data-set* para as posições intermédias e um conjunto dos ângulos das juntas de algumas posições intermédias. Inicialmente o algoritmo VEAG coloca a zero toda a *string data-set*. Cada *string* desenvolve-se gradu-

almente através de um conjunto de ângulos (das juntas) de posições intermédias recebidas da camada de geração de posições.

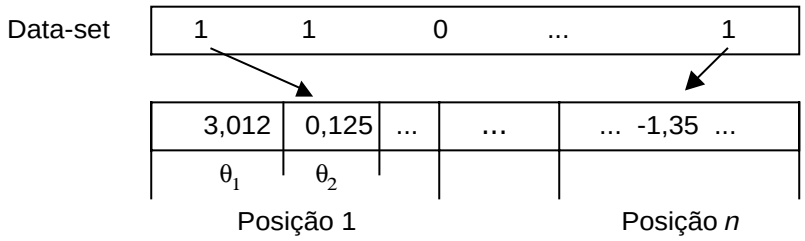


Figura 3.36 Codificação na camada de geração de trajectória [14].

Os ângulos nesta camada são codificados através de valores reais.

3.5.3.3 Função de aptidão

Sejam p_a e p_b duas posições com o ponto intermédio p_c. As variáveis a serem optimizadas são as variáveis angulares do manipulador. O objectivo consiste em gerar a trajectória mínima, o mais afastada possível dos objectos. Assim, a função a minimizar vem:

$$f = w_1 \times f_p + w_2 \times f_d + w_3 \times f_r + w_4 \times (max_{pot})^2 + w_5 \times sum_{pot}$$

onde:

w_i (i = 1,…,5) são os pesos dos coeficientes;

f_p é a soma quadrática da distância dos pontos terminais do manipulador entre p_a e p_c e entre p_b e p_c;

f_d é a soma quadrática das diferenças angulares entre p_a e p_c e entre p_c e p_b;

f_r é a soma da função de avaliação, utilizando uma função de distribuição normal, para garantir que cada junta esteja no seu intervalo disponível;

max_{pot} é o valor do pseudo-potencial máximo (p) (*i.e.* quando há colisão com obstáculos);

sum_{pot} representa a soma dos valores dos pseudo-potenciais para todos os pontos amostrados.

3.5.3.4 Operadores relativos a infecção de vírus (camada de geração de posições)

Os tipos de infecção de vírus possíveis, nas *strings* hospedeiro, são os seguintes:

- operador de transcrição inversa – um vírus coloca uma cópia do seu conteúdo dentro de um hospedeiro (figura 3.37);
- operador de transdução (*transduction*): cópia e reposição – no operador tipo cópia o valor de um vírus é substituído através da copia de uma sub-*string* da população hospedeiro, no operador tipo reposição são colocados alguns *bits* aleatoriamente do vírus a # (figura 3.38).

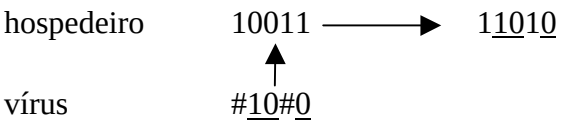


Figura 3.37 Operador de transcrição inversa [14].

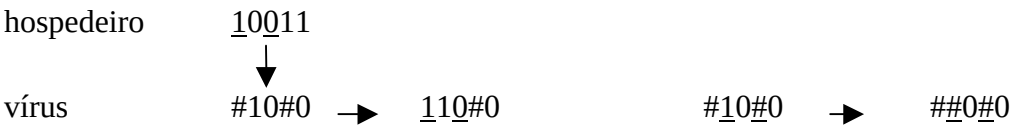


Figura 3.38 Operador de transdução: cópia e reposição [14].

3.5.3.5 Funções relativas à população de vírus

As funções que a população de vírus pode exercer são:

$fitvirus_i$ – parâmetro relativo à infecção do vírus *i* e a taxa de infecção da transcrição inversa;

$fithost_j$ – valor de aptidão do hospedeiro *j* antes de ocorrer a infecção;

$fithost'_j$ – valor de aptidão do hospedeiro *j* depois de ocorrer a infecção;

$fitvirus_{i,j} = fithost'_j - fithost_j$;

$\text{fitvirus}_i = \sum_{j \in S} \text{fitvirus}_{ij}$, onde *i* é o vírus e S é o conjunto de hospedeiros que estão infectados pelo vírus *i*;

O tempo de vida de um vírus é dado pela seguinte fórmula:

$$\text{life}_{i,t+1} = r \times \text{life}_{i,t} + \alpha \times \text{fitvirus}_i$$

onde:

t é o número da geração;

r é a taxa de redução de vida;

α é o coeficiente de redução de vida.

3.5.3.6 Operador de mutação (camada de geração de trajectórias)

O operador de mutação modifica um gene aleatoriamente na *string data-set*. Para melhorar o desempenho é aplicado o operador de mutação auto-adaptativo utilizando uma variável aleatória normal com média zero:

$$x_{j,i} = x_{j,i} + N(0, a_i \times \frac{\text{f}_j}{max\ \text{f}_j} + b)$$

onde:

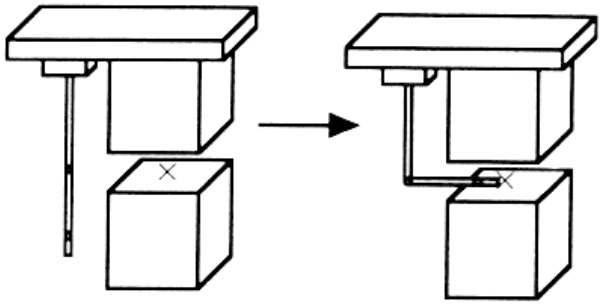
$x_{j,i}$ representa o ângulo *i* do hospedeiro *j*;

f_j é a aptidão do hospedeiro j.

Um vírus tem um sub-conjunto de posições intermédias e transmite-as aos hospedeiros para, em pouco tempo, gerar uma trajectória livre de colisões.

3.5.3.7 Simulação

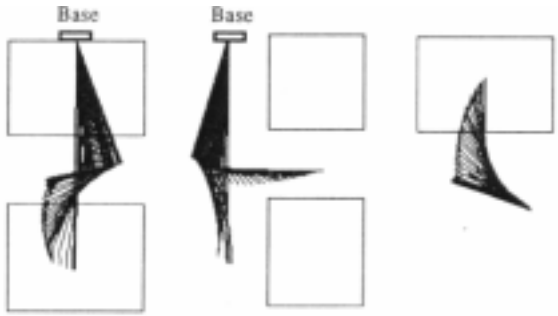
Na simulação é utilizado um manipulador com 7 graus de liberdade situado num ambiente (dividido em $30 \times 30 \times 30$ células) com dois obstáculos (figura 3.39). O pseudo--potencial máximo é de 5. O número de indivíduos na população de hospedeiros é 50 (100) para a camada de geração de trajectórias (geração de posição). Por outro lado a população de vírus é 10% para a população de hospedeiros.



posição inicial posição final

Figura 3.39 Simulação com um manipulador com 7 graus de liberdade [14].

A figura 3.40 mostra os resultados da solução óptima em diversas perspectivas.



vista de frente vista de lado vista de cima

Figura 3.40 Trajectória resultante da simulação [14].

3.5.4 Optimização de trajectórias de dois robôs móveis

3.5.4.1 Introdução

Rana e Zazala [2] propõem uma técnica evolutiva para planear o tempo mínimo em malha aberta do histograma do binário de controlo. O planeamento é feito no espaço das juntas do manipulador e a trajectória é representada por uma *string* de pontos intermédios ligados por *splines* polinomiais cúbicas. A modificação da trajectória é feita através de AEs para pesquisar um percurso, livre de colisões, óptimo em termos de tempo. A técnica apresentada calcula o tempo óptimo para percorrer a trajectória. Este tempo é utilizado na função de aptidão a ter em conta na pesquisa global para o caminho óptimo.

3.5.4.2 Formulação do problema

A formulação para um manipulador robótico com dois graus de liberdade é considerado em primeiro lugar.

O espaço dos pontos terminais é mostrado na figura 3.41 (a). A configuração do espaço para A_1 (robô 1) é $c_1 = \theta_1 \times \theta_2 \in \Re^2$ para A_2 (robô 2) é $c_2 = \alpha_1 \times \alpha_2 \in \Re^2$. Os espaços c_1 e c_2 são considerados desacopulados. O espaço A_2 será considerado um obstáculo para A_1 quando $cA_2 \subset c_1$ e A_1 é considerado um obstáculo para A_2 quando $cA_1 \subset c_2$. Com o fim de determinar colisões entre os dois manipuladores, os elos são aproximados por círculos (figura 3.42) onde o raio e o espaçamento dos círculos depende da precisão requerida.

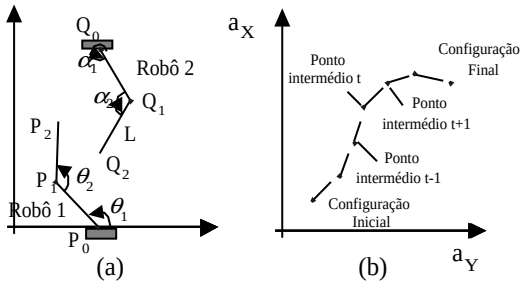


Figura 3.41 (a) Espaço operacional dos dois robôs; (b) pontos da trajectória [2].

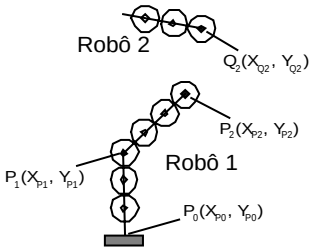


Figura 3.42 Aproximação dos robôs por círculos [2].

Para determinar a curva de tempo óptima no plano (s, $\dot{s}$) é calculada a integração da aceleração máxima $\ddot{s}_a$ dada pela equação (3.4) desde o ponto (0, 0) até ao ponto onde as curvas de aceleração e desaceleração se encontram, e o integral da desaceleração máxima dada pela equação (3.5) desde o ponto onde as duas curvas se encontram até o ponto (s_{fi-nal}, 0).

$$\ddot{s}_a(s,\dot{s}) = \max_i \left\{ \min_{T_i} \left(\frac{T_i - g_i - c_i \times \dot{s}^2}{m_i} \right) \right\}; i = 1,..,n \tag{3.4}$$

$$\ddot{s}_d(s,\dot{s}) = \min_i \left\{ \max_{T_i} \left(\frac{T_i - g_i - c_i \times \dot{s}^2}{m_i} \right) \right\}; i = 1,..,n \tag{3.5}$$

3.5.4.3 Representação

A codificação das *strings* é a seguinte:

$$p_1, p_2, \ldots, p_{n-1}, q_1, q_2, \ldots, q_{n-1}$$

onde $p_i \in c_1$ é o ponto intermédio *i* do percurso 1 e $q_i \in c_2$ é o ponto intermédio *i* do percurso 2. Os pontos inicial e final não são incluídos uma vez que se mantêm inalterados.

3.5.4.4 Função de aptidão

A função de aptidão é composta por uma primeira componente relativa ao tempo que demora o percurso das trajectórias (PC_1) e uma segunda componente relativa à ocorrência de colisões entre A_1 e A_2 (PC_2), ou seja:

$$PC_2 = \sum_{i=1}^{n} pc_i$$

tal que (i = 1, 2,…,n)

$$pc_i = \begin{cases} 1 & \text{se } A_1 \text{ e } A_2 \text{ colidem} \\ 0 & \text{outros casos} \end{cases}$$

$$f = C_{max} - k_1 \times PC_1 + k_2 \times PC_2$$

onde C_{max}, k_1 e k_2 são constantes.

3.5.4.5 Operadores genéticos utilizados

Os operadores genéticos utilizados são:

- Reprodução – baseado na roleta redonda.
- Cruzamento – ponto duplo (um ponto de cruzamento na trajectória de cada robô). O cruzamento é efectuado apenas se a distância entre as coordenadas do local de cruzamento forem menores que um determinado valor.
- Redistribuição/relaxamento da trajectória – após a ocorrência do operador de cruzamento, vão existir transições abruptas nos locais de cruzamento. Este operador tem como finalidade “suavizar” essas transições. Assim, a trajectória entre os pontos intermédios são redistribuídos com distâncias iguais ao longo do percurso.

- Mutação – são geradas trajectórias novas e repostas aleatoriamente na nova população com uma certa probabilidade.

3.5.4.6 Resultados das simulações

São efectuadas simulações com robôs RR e RRR.

- Simulação com robôs de dois graus de liberdade

Pretende-se que o robô 1 se desloque da posição (θ_1, θ_2) = (45º, 225º) para a posição (135º, 123º) e que o robô 2 se desloque do ponto (α_1, α_2) = (45º, 225º) para o ponto (315º, 315º). Os parâmetros físicos dos robôs encontram-se na tabela 3.5. A figura 3.43 mostra os resultados para as trajectórias dos robôs. Os binários que devem ser aplicados aos eixos dos robôs encontram-se na figura 3.44. A duração do movimento é de 0,7785 s.

Tabela 3.5 Características dos robôs RR [2].

Comprimento dos elos	0,4 m
Massa dos elos	0,5 kg
Momento de inércia dos elos relativamente ao seu centro de gravidade	0,1 kg-m^2
Distância dos eixo *i* ao centro de gravidade dos elo *i*	0,2 m
Distância entre a primeira junta do robô 1 e a primeira junta do robô 2	1,8 m
Binários máximos dos eixos	±10 N-m

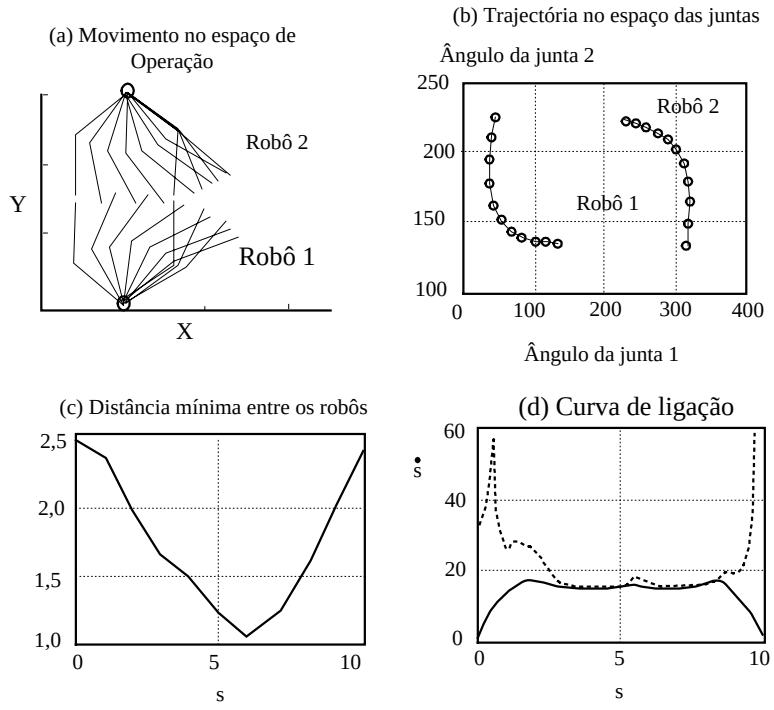


Figura 3.43 Resultados dos pontos terminais dos robôs planares [2].

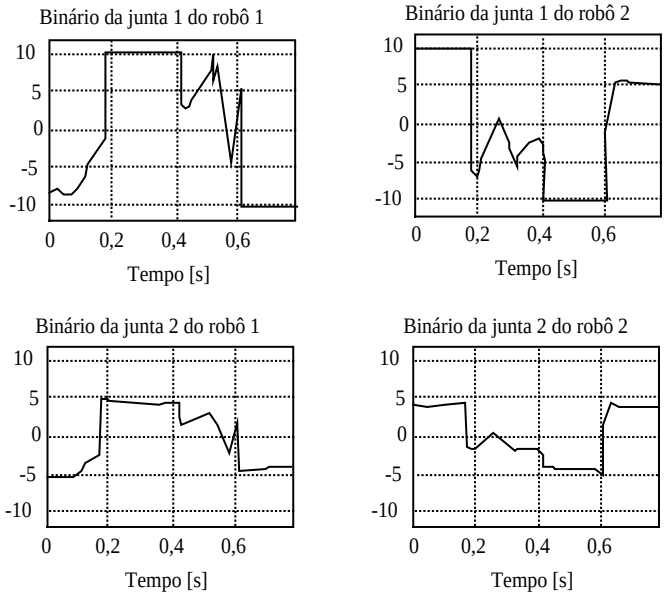


Figura 3.44 Binários aplicados aos robôs com dois graus de liberdade [2].

- Simulação com robôs de três graus de liberdade

Nesta simulação são utilizados dois robôs Puma 560 (figura 3.45) mas são apenas considerados os três primeiros graus de liberdade, na perspectiva de se evitar colisões, pois os restantes são utilizados para orientar o ponto terminal do robô.

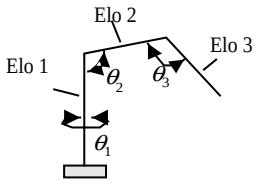


Figura 3.45 Representação do robô Puma 560 [2].

Os robôs devem-se deslocar das posições inicias $(\theta_1, \theta_2, \theta_3) = (150º, -15º, 30º)$ e $(\alpha_1, \alpha_2, \alpha_3) = (-150º, -15º, 30º)$ para as posições finais (150º, -15º, 30º) e (-15º, -15º, 30º). Os parâmetros dos robôs estão indicados na tabela 3.6. As distâncias entre as bases dos robôs é de 1,5 m.

Tabela 3.6 Parâmetros dos robôs RRR [2].

Elo	Comprimento [m]	Centro de gravidade [m]	Massa [kg]	I_{xx} [kg-m^2]	I_{yy} [kg-m^2]	I_{zz} [kg-m^2]	Binários das Juntas [Nm]
1	0,1	0,05	12	0,4750	0,0150	0,4750	±200
2	0,5	0,25	24	0,0075	2,0037	2,0037	±400
3	0,5	0,25	6	0,0018	0,5009	0,5009	±300

Os resultados são apresentados na figura 3.46 à figura 3.49. O tempo total é de 0,6053 s.

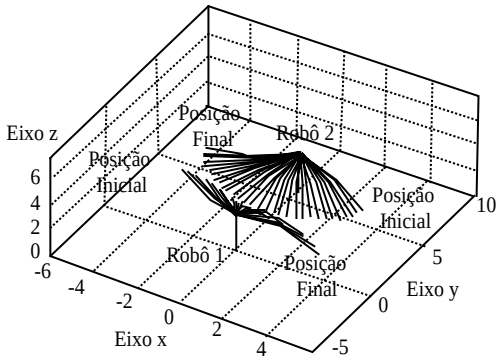


Figura 3.46 Trajectórias finais dos robôs com 3 graus de liberdade [2].

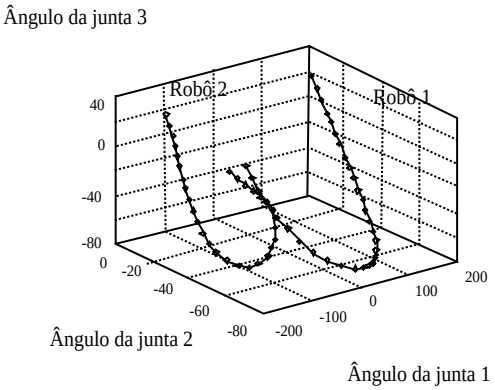


Figura 3.47 Percursos no espaço das juntas [2].

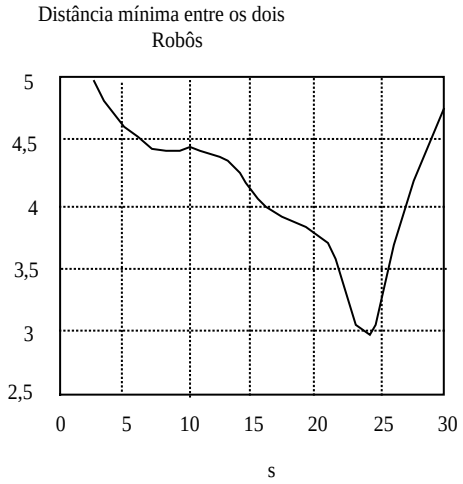
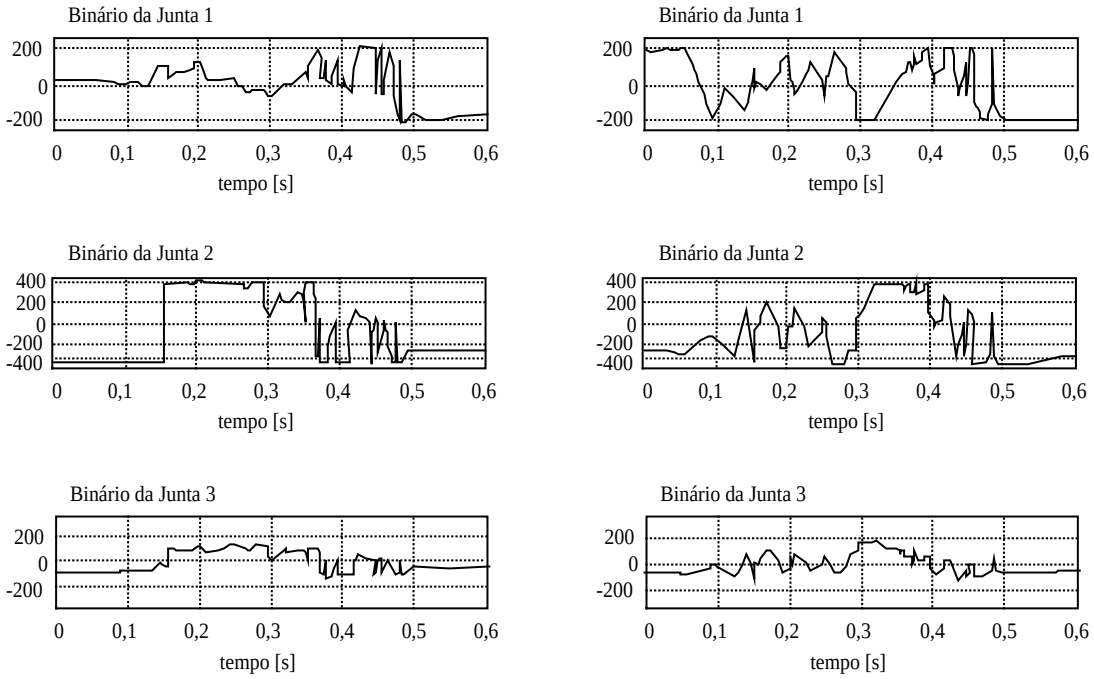


Figura 3.48 Distância mínima entre os robôs [2].



(a) (b)

Figura 3.49 Binário aplicado as juntas dos robôs com três graus de liberdade [2].

(a) Binário de controlo para o robô 1; (b) binário de controlo para o robô 2.

3.6 Calibração de robôs

3.6.1 Introdução

Neste secção é apresentado um algoritmo para estimar os parâmetros do robô, através de AGs, tendo em vista a sua calibração.

3.6.2 Planeamento óptimo de calibração de um robô

3.6.2.1 Introdução

Zhuang *et al.* [10] desenvolveram uma técnica com o fim de resolver problemas de optimização na calibração de robôs.

A escolha da configuração óptima pode ser vista como um problema de determinação da configuração do robô tal que o efeito das incertezas na estimação dos parâmetros cinemáticos seja minimizado.

3.6.2.2 Observabilidade do erro dos parâmetros

Para desenvolver o algoritmo é necessário estimar um vector, ρ, de parâmetros cinemáticos independentes, para o cálculo dos mínimos dos quadrados relativos ao posicionamento e orientação do robô. Uma aproximação comum consiste em definir um conjunto de incrementos nas variáveis, $d\rho$, aditivos ao vector ρ^0 de parâmetros nominais do robô no módulo cinemático. A linearização do robô através das equações directas da cinemática (relativas a ρ^0) na junta particular i da configuração medida q_i, fornece a matriz de identificação Jacobiana:

$$J_i = J(q_i, \rho^0)$$

Esta medida relaciona J_i, os erros de posição do ponto terminal na configuração i para $d\rho$, e o vector y_i de erros dos parâmetros cinemáticos independentes da seguinte forma:

$$y_i = J_i \times d\rho$$

Seja $Y = [y_1^T, y_2^T, \ldots, y_s^T]^T$, onde s é o número de medidas, e defina-se uma matriz Jacobiana, J, de identificação 'agregada', que é obtida através do 'empilhamento' de vários J_i. A equação de medida total de $d\rho$ para a estimação do mínimo dos quadrados, é:

$$Y = J \times d\rho$$

O erro cinemático do vector $d\rho$ é observável, se e só se a matriz $J^T \times J$ tem característica máxima.

O número de condição de J é usado como índice de observabilidade neste trabalho

$$\text{cond}(J) = \frac{\sigma_{max}}{\sigma_{min}}$$

onde σ_{max} e σ_{min} são os valores singulares máximos e mínimos da matriz J.

3.6.2.3 Formulação do problema

O problema de planeamento experimental óptimo de calibração de robôs pode ser formulado como:

Determinar *m* configurações medidas, no espaço dos eixos do robô, admissíveis de tal forma que o efeito das incertezas no erro de estimação dos parâmetros seja mínimo. De notar que *m* deve ser suficientemente grande. Quando é possível medir todos os valores da posição do ponto terminal do robô a condição necessária é $6 \times m > c$, onde *c* é definido como o número de parâmetros cinemáticos independentes do manipulador.

O problema descrito acima é de resolução difícil pois a relação matemática envolvendo os erros dos parâmetros, o ruído medido e a configuração, é complexa de modelizar. Todavia, se for adoptada uma técnica baseada em modelos de erros, para a identificação dos parâmetros cinemáticos, então o problema da selecção da configuração óptima pode ser resolvido através do número de condição da matriz de identificação Jacobiana. Isto verifica-se sempre que o limite superior da norma dos parâmetros de erro seja proporcional ao número de condição da matriz de identificação Jacobiana. Assim, o problema pode ser formulado da seguinte forma: Determinar *m* configurações medidas do robô (no espaço das juntas) alcançáveis pelo robô tal que cond(J) é mínima.

3.6.2.4 Função de aptidão

A medida para seleccionar as *strings* mais aptas pode ser tanto o número de condição como o índice de observabilidade da matriz de identificação Jacobiana. Estes índices matemáticos reflectem um certo grau do erro de estimação se as variáveis das juntas seleccionadas forem utilizadas para estimarem os parâmetros de um estágio posterior. A

relação entre os valores do número de condição ou do índice de observabilidade e a qualidade da solução não é conhecida.

Os índices de desempenho variam com as configurações medidas, que podem ser representadas pelas variáveis nos eixos do robô. No trabalho de Zhuang *et al.* é utilizada o número de condição da matriz de identificação Jacobiana, definida na secção anterior, como função de aptidão.

3.6.2.5 Representação

Supondo que o robô tem três juntas (RRP) e que cada ângulo (θ_1, θ_2, θ_3) é codificado com 4 *bits*, conclui-se que são necessários 12 *bits* para codificar cada sistema de juntas (*i.e.* uma configuração medida). Se for necessário medir 4 configurações para calcular a matriz de identificação Jacobiana, então a *string* que representa o vector de variáveis é constituída por 48 *bits* ($s_L = m \times n \times b$; s_L – número de *bits*, b – número de *bits* de cada eixo, n é o número de graus de liberdade do robô).

3.6.2.6 Operadores genéticos

Os operadores genéticos adoptados são:

Operador de reprodução – Depois das *strings* serem avaliadas é feito um escalonamento para prevenir que super-*strings* prosperem em excesso. De seguida é aplicado o método de selecção de amostragem estocástica (*stochastic remainder*) para determinar o número máximo de cópias que vão acasalar. O número de cópias da melhor *string* é limitado e o escalonamento aplicado é linear. Por outro lado, é também testada a estratégia elitista. Deve ainda referir-se que o operador de cruzamento é de ponto simples e que o operador de mutação usado é o clássico.

3.6.2.7 Parâmetros do algoritmo genético

Para seleccionar os parâmetros do AG é usado um AG meta-nível. Uma população de AGs com parâmetros diferentes está em competição com outra. Baseada no seu desempenho, o AG meta-nível ajusta os parâmetros do AG. O processo continua até que ao critério de convergência ser satisfeito.

Nesta simulação é feita a pesquisa sucessiva (método parecido com AGs meta nível) para determinar os parâmetros do AG. A simulação começa com os parâmetros da tabela 3.7. Os parâmetros são modificados um por cada geração, até que todos os parâmetros sejam mudados. Este processo continua até não se conseguir um melhoramento significativo.

Tabela 3.7 Parâmetros do AG inicial e final [10].

	AG inicial	AG final
Tamanho da população	50	70
Probabilidade de cruzamento	0,5	0,6
Probabilidade de mutação	0,01	0,01
Número máximo de cópias da melhor *string*	2	2
Número de *bits* de cada eixo	6	8
Método elitista	Sim	Sim

3.6.2.8 Experiências e resultados

É utilizado um robô Intelledex RRP para testar o algoritmo. A melhor solução obtida é depois utilizada para calibrar o robô seguida da simulação real.

Foi considerado que o dispositivo de medição garante uma precisão moderada (0,001 cm) e que os parâmetros do robô não são conhecidos com precisão. Os erros dos sensores das juntas são desprezáveis. A simulação foi iniciada com os parâmetros nominais do

robô e, iterativamente, convergiu para os parâmetros reais, usando um número de 70 medições da posição do ponto terminal do robô não ideal. Após a convergência do algoritmo os parâmetros cinemáticos identificados do robô são colocados na rotina de verificação. Nesta rotina, um número diferente de posições é gerado pelos parâmetros identificados do robô e são medidas pelo "dispositivo de medição" simulado. A discrepância entre as posições do robô calculada e medida, é referido como o erro de posição, para cada posição do robô gerada no estado de verificação. O sistema de parâmetros do AG que conduz ao melhor resultado é considerado o vencedor. A verificação dos resultados é apresentada na figura 3.50.

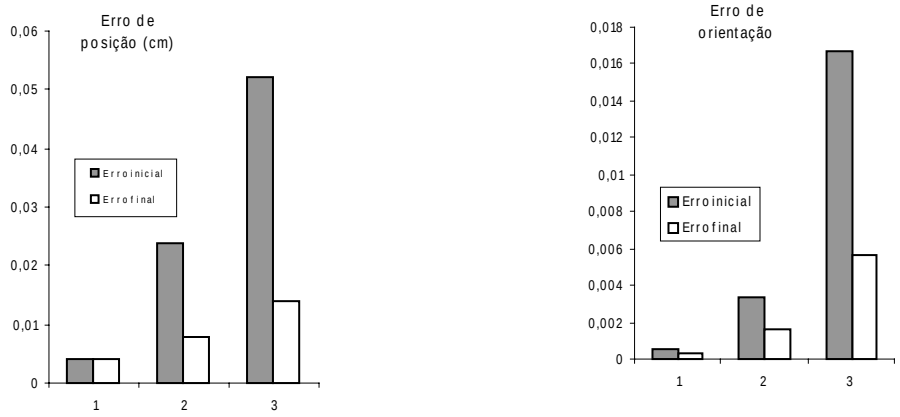


Figura 3.50 Resultado das experiências [10].

(a) Erro de posição; (b) erro de orientação.

1. Erro utilizando a melhor *string*
2. Erro utilizando uma *string* média
3. Erro utilizando a pior *string*

3.7 Referências

[1] A.B. Doyle, D.I Jones, "2nd Portuguese Conference on Automatic Control", pp. 312-218, Porto, Portugal, September, 11-13 1996.

[2] A. S. Rana, A. Mente Zalzala, "An Evolutionary Planner for Near Time-Optimal Collision-Free Motion of Multi-Arm Robotic Manipulators", UKACC International Conference on Control, vol. 1, pp. 29-35, 1996.

[3] Carlos Jorge Almeida Costa, "Algoritmos Genéticos e Robótica", Porto, Julho 1997.

[4] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski, "Adaptative evolutionary Planner/Navigator for Mobile Robots", IEEE Transactions on Evolutionary Computation Vol. 1, Num. 1 pp. 18 – 28, April 1997.

[5] Jeongheon Han, W. K. Chung, Y. Youm, S. H. Kim, "Task Based Design of Modular Robot Manipulator using Efficient Genetic Algorithm", IEEE International Conference on Robotics and Automation, pp. 507-512, Alburquerque, New Mexico, 20-25 April 1997.

[6] Jin-Oh Kim, Pradeep K. Khosla, "A Multi-Population Genetic Algorithm and Its Application to Design Of Manipulators", IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, pp. 279-286, Raleight, North Caroline, USA, 7-10 July 1992.

[7] Gonzalo Cabodevila, Gabriel Abba, "Quasi Optimal Gait for a Biped Robot using Genetic Algoritm", IEEE International Conference on Systems, Man, and Cybernetics – Computational Cybernetics and Simulation, pp. 3960-3965, Orlando, Florida, 12-15 October 1997.

[8] Gray B. Parker, "Generating Arachnid Robot Gaits With Cyclic Genetic Algorithms", http://www.cs.indiana.edu/hyplan/gaparker.html, 29-5-98.

[9] Gray B. Parker, David W. Braun, Ingo Cyliax, "Learning Gaits for the Stiquito", 8th International Conference on Advanced Robotics, pp. 285-290, Monterey, California, USA, 7-9, July 1997.

[10] Hanqi Zhuang, Jie Wu, Weizhen Huang, "Optimal Planning Calibration Experiments by Genetic Algorithms", pp. 981-986, IEEE International Conference on Robotics and Automation, Minneaplois, Minnesota, 22-28, April 1996.

[11] LI Gan, "Designing control Strategies For a simulated Robot Using Non-Randomly Initialized Genetic Algorithms", http://cs226.cs.unr.edu/~sushil/su…isca/reno_96/ganli/html/paper.html, April 9 1996.

[12] M. Anthony Lewis, Andrew H. Fagg, "Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot", IEEE International Conference on Robotics and Automation, pp. 2618-2623, 12-14 May 1992.

[13] Mingwu Chen, Ali M. S. Zalzala, "A Genetic Approach to Motion Planning of Redundant Mobile Manipulator Systems Considering Safety and Configuration", Journal of Robotic Systems vol. 14, Num. 7, pp. 529-544, 1997.

[14] Naoyuki Kubota, Takemasa Arakawa, Toshio Fukuda, "Trajectory Generation for Redundant Manipulator Using Virus Evolutionary Genetic Algorithm", IEEE International Conference on Robotics and Automation, pp. 205-210, Albuquerque, New Mexico, 20-25 April 1997.

[15] O. Chocron, P. Bidaud, "Evolutionary Algorithms in Kinematic Design of Robotic Systems", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1111-1117, Grenoble, France, 7-11 September 1997.

[16] P. Chedmail, E. Ramstein, "Robot Mechanism Synthesis and Genetic Algorithms", IEEE International Conference on Robotics and Automation, pp. 3466-3471, Minneapolis, Minnesota, 22-28 April 1996.

[17] Q. Wang, A. M. S. Zalzala, “Genetic control of Near Time-optimal Motion for an Industrial Robot Arm, IEEE International Conference on Robotics and Automation, pp. 2592-2597, Minneapolis, Minnesota, 22-28 April 1996.

[18] Shane Farritor, Steven Dubowsky, “A Self-Planning Methodology for Planetary Robotic Explorers”, 8th International Conference on Advanced Robotics, pp. 449-504, Hyatt Regency Monterey, California, USA, 7-9 July 1997.

[19] Takemasa Arakawa, Toshio Fukuda, “Natural Motion Generation of Biped Locomotion Robot using Hierarchical Trajectory Generation Method Consisting of GA, EP Layers”, IEEE International Conference on Robotics and Automation, pp. 211-216, Albuquerque, New Mexico, 20-25 April 1997.

[20] Toshio Fukuda, Youichirou Komata, Takemasa Arakawa, “Stabilization Control of Biped Locomotion Robot based Learning with Gas having Self-Adaptive Mutation and Recurrent Neural Networks”, IEEE International Conference on Robotics and Automation, pp. 217-220, Albuquerque, New Mexico, 20-25 April 1997.

[21] Woong–Gie Han, Seung-Min Baek, Tae-Yong Kuc, “Genetic Algorithm Based Path Planning and Dynamic Obstacle Avoidance of Mobile Robots”, Conference Proceedings IEEE International Conference on Systems, Man, and Cybernetics, pp. 2747 – 2751, Hyatt, Orlando, Florida, USA, 12-15 October 1997.

[22] Yuval Davidor, “Genetic Algorithms and Robotics, a heuristic strategy for optimization”, Word Scientific Publishing Co. Pte Ltd, Series in Robotics and Automated Systems – Vol. 1, ISDN 9810202172, 1991.

[23] Zbigniew Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs”, Third Edition, Springer, pp 253 – 261, ISBN 3-540-60676-9, 1996.

4 Algoritmos genéticos no planeamento de trajectórias para manipuladores robóticos

4.1 Introdução

Nesta secção é desenvolvido um algoritmo de planeamento de trajectórias para manipuladores robóticos. Assim, na secção 4.2 é exposto o problema e um modelo para a sua resolução. Nas secções 4.3 e 4.4 é estudado a influência dos operadores de mutação e de cruzamento e, de seguida, na secção 4.5 é analisado o peso dos coeficientes da função de aptidão. Uma vez formulados os aspectos fundamentais, na secção 4.6 são expostos os resultados para um ambiente com um obstáculo e na secção 4.7 é estudado o efeito do comprimento do cromossoma na solução do problema. Na secção 4.8 é apresentado o resultado de um problema com dois obstáculos. Na secção 4.9 é indicado um modo de reduzir a ondulação da cinemática diferencial. Por último, na secção 4.10 é analisado o tempo que o robô requer para planear as trajectórias.

4.2 Formulação do problema

Pretende-se mover um manipulador robótico do ponto (1, 1) até ao ponto (0, 2). O robô é constituído por um conjunto de elos com um comprimento total de 2 [m]. Assim, o comprimento de cada elo do robô de 2, 3 e 4 elos é, respectivamente, de 1, 2/3 e 0,5 [m]. Considera-se que cada elo do robô pode rodar livremente em torno do seu eixo pelo que o problema é resolvido através da cinemática directa.

A configuração angular inicial do robô de 2 elos é de (0º; 90º) para os elos 1 e 2. A posição inicial do robô de 3 elos foi escolhida de modo a obter a configuração próxima da configuração do robô com dois elos. Assim, a posição inicial do robô é (–10,89º; 34,11º; 34,11º), respectivamente para os elos 1, 2 e 3. De modo análogo, a posição inicial do robô de 4 elos é de (0º; 0º; 90º; 0º), respectivamente para os elos 1, 2, 3 e 4.

Com este fim são usados AGs onde a representação da solução é constituída por um cromossoma com 15 genes. Cada gene é composto pelos deslocamentos angulares das juntas do robô. Assim, para um robô com três elos (figura 4.1) o gene é composto por três valores ($\Delta q_1, \Delta q_2, \Delta q_3$) (figura 4.2), sendo o primeiro valor respeitante ao deslocamento angular da base do robô, o segundo correspondente ao ângulo do meio, e por fim, $\Delta \alpha_3$ o deslocamento entre os dois elos terminais.

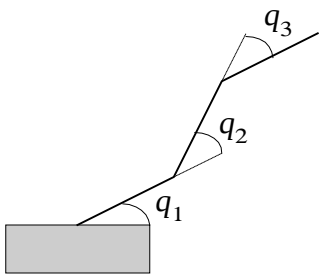
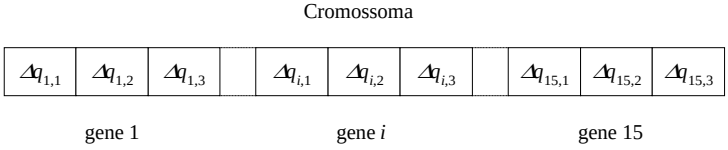


Figura 4.1 Robô com três elos.



O parâmetro $\Delta q_{i,j}$ é o deslocamento i ($1 \leq i \leq 15$) do ângulo j ($1 \leq j \leq 3$) do robô.

Figura 4.2 Representação do cromossoma para o robô de três elos.

A configuração da posição angular k do robô determina-se a partir do somatório dos desvios angulares do cromossoma até essa posição e com a posição inicial do ângulo k (q_{0k}). Assim, o valor do ângulo k do robô na posição i é dado pela seguinte expressão:

$$q_{ik} = \sum_{j=1}^{i} \Delta q_{jk} + q_{0k}$$

E, a configuração da posição 15 do robô (posição final) o valor dos ângulos é o seguinte:

$$(\sum_{j=1}^{15} \Delta q_{j1} + q_{01}, \sum_{j=1}^{15} \Delta q_{j2} + q_{02}, \sum_{j=1}^{15} \Delta q_{j3} + q_{03})$$

Os genes são codificados através de três valores reais e cada valor real pode ter um valor compreendido entre –1,8 e 1,8. Este valor corresponde ao desvio angular máximo (em radianos) que uma junta pode sofrer entre duas posições consecutivas. Para a execução das simulações foi usada uma população de 100 cromossomas e um operador de cruzamento de segregação (*i.e.* o ponto de cruzamento não ocorre dentro de genes). O modelo usado foi um AG com um método de selecção baseado no posto, geracional e elitista. Deve também referir-se que foi utilizado o operador de mutação uniforme. O número de gerações das simulações é de 5000.

Pretende-se obter uma trajectória que atinja o ponto desejado com um custo mínimo, ou seja, que as actuações nas juntas do robô seja mínima, sem ocorrerem colisões com os obstáculos. Para transformar o problema de minimização em maximização, e normalizar a função objectivo no intervalo [0; 1], utiliza-se a função exponencial. Assim, a função de aptidão vem:

$$f = e^{-\alpha_1 \times |PP - PF| - \alpha_2 \times \sum_{i=1}^{n} \Delta q_i - \alpha_3 \times nPNA} \qquad (4.1)$$

onde:

PP é o ponto que se pretende atingir pelo robô;

PF é o ponto atingido pelo robô após a simulação;

Δq_i representa a actuação da junta i do robô;

nPNA é o número de pontos não admissíveis;

α_i são os pesos de cada factor;

n é o número de genes.

O peso α_1 deve ser superior ao peso α_2, caso contrario uma solução que se desloque no sentido do ponto desejado tem um desempenho inferior à solução onde o robô permanece no ponto de partida ($\Delta q_{ik} = 0$ para todo o i e k).

O valor de nPNA fornece uma medida do conflito existente entre o robô e os obstáculos. Este valor é calculado do seguinte modo: cada elo do robô é dividido em *p* partes iguais (*p* = (4, 3, 2) respectivamente para um robô com 2, 3 e 4 elos). De seguida, são atribuídos pesos a cada um dos pontos, sendo os pesos distribuídos da seguinte forma: 1 para o ponto correspondente ao ponto terminal, 2 para o ponto adjacente ao ponto terminal, e assim sucessivamente até atingir o ponto da base. Por fim, o valor nPNA é calculado a partir da soma desses valores. Quando, não são considerados obstáculos no problema o valor de α_3 é zero como seria de esperar.

O algoritmo genético utilizado é o seguinte

```
Algoritmo Gerador de trajectórias
Inicializa_População_de_Trajectórias
Avalia
Geração = 0
Repetir
 selecciona
 Insere_melhor_elemento_na_nova_população
 Cruzamento_de_Ponto_Simples
 Mutação_Uniforme
 Avalia
 Geração = Geração + 1
Até Geração = 5000
Fim do algoritmo
```

Algoritmo 4.1 Gerador de trajectórias.

No algoritmo há que salientar que a inicialização de uma trajectória é feita do seguinte modo:

```
Procedimento Inicializa_Trajectória
j = 15 [15 genes]
i = 1
Repetir
 k = Random(j) [gera um número aleatório entre 1 e j]
 gene = random [gera um gene aleatoriamente]
 Repetir de m = i ate m = i + k
    cromossoma[m] = gene
 j = 15 – k
 i = i + k
Ate j = 0
Retornar
```

Algoritmo 4.2 Procedimento Inicializa_Trajectória.

Com o algoritmo 4.2 evita-se que o percurso da trajectória se desenvolva em torno do ponto inicial.

4.3 Variação da probabilidade de mutação

4.3.1 Introdução

Nesta secção são apresentadas as simulações dos robôs com 2, 3 e 4 elos, num ambiente sem obstáculos. Nestas simulações, a probabilidade de cruzamento é mantida constante com o valor de $p_c = 0{,}8$ e a probabilidade de mutação varia de $p_m = 0{,}05$ a $p_m = 0{,}5$. A função de aptidão utilizada é:

$$f = e^{-10^{-4} \times |PP - PF| - 10^{-6} \times \sum_{i=1}^{n} \Delta q_i} \qquad (4.2)$$

onde:

PP é o ponto que se pretende atingir pelo robô;

PF é o ponto atingido pelo robô após a simulação;

Δqrepresenta a actuação de uma junta do robô;

n = 15 genes.

4.3.2 Resultado das experiências

Na figura 4.3 e na figura 4.4 são referidos os resultados de duas experiências para o robô de 2 elos, com probabilidade de mutação respectivamente de $p_m = 0{,}005$ e de $p_m = 0{,}03$. Na figura 4.5 e na figura 4.6 encontram-se os resultados de experiências para o robô com 3 e 4 elos com uma probabilidade de mutação de $p_m = 0{,}03$.

Na tabela 4.1 encontram-se os valores de aptidão das experiências realizadas assim como o número da geração onde as soluções foram encontradas.

Tabela 4.1 Valores de aptidão das soluções ($p_c = 0{,}8$).

Número de elos do robô	Aptidão da solução	Geração onde foi encontrada a melhor solução
2 ($p_m = 0{,}005$)	0,9999969	4266
2 ($p_m = 0{,}03$)	0,9999958	4707
3 ($p_m = 0{,}03$)	0,999995	2289
4 ($p_m = 0{,}03$)	0,9999929	3872

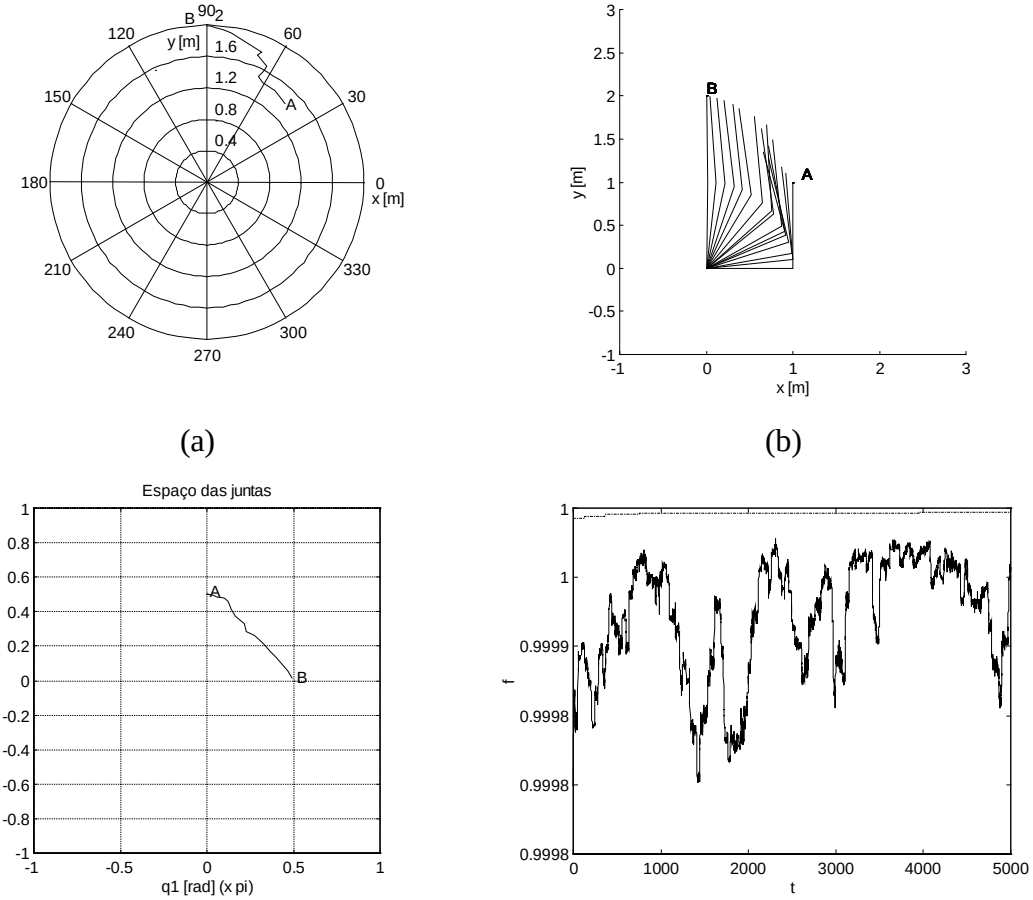


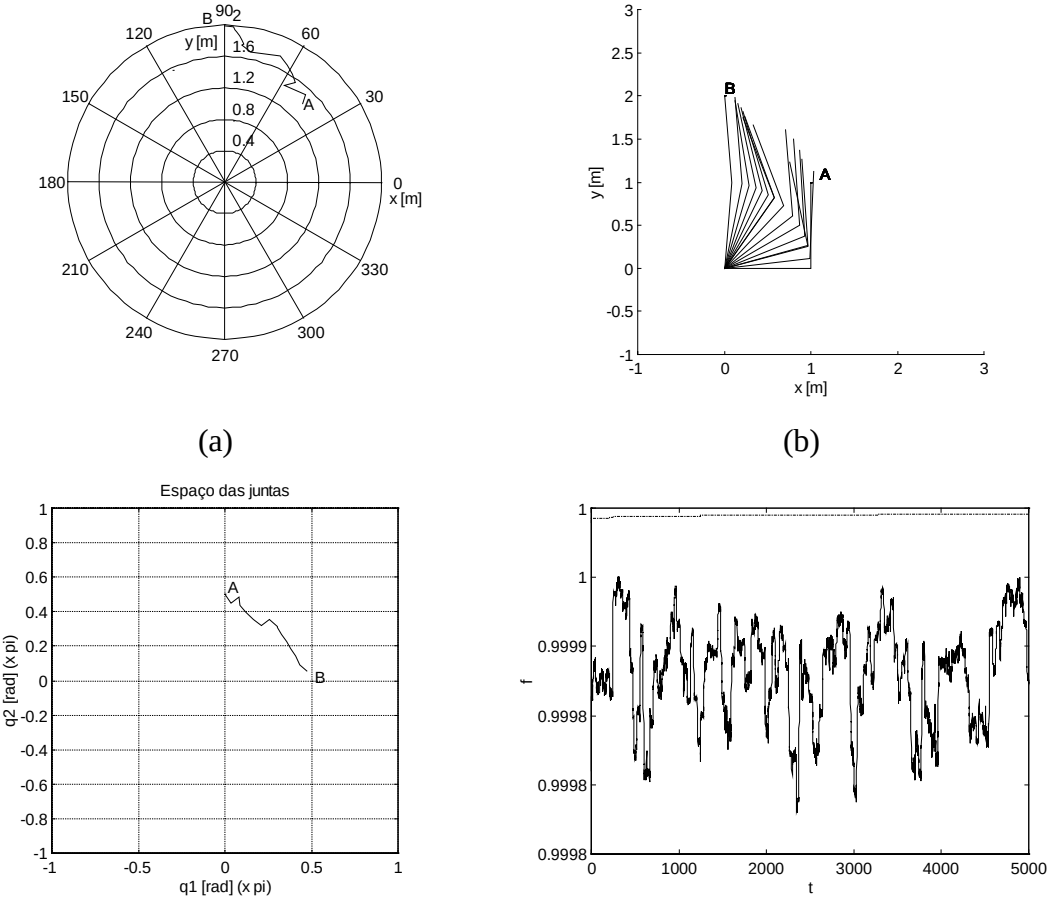
Figura 4.3 Resultados do robô de 2 elos (p_c = 0,8; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a) (b)

(c) (d)

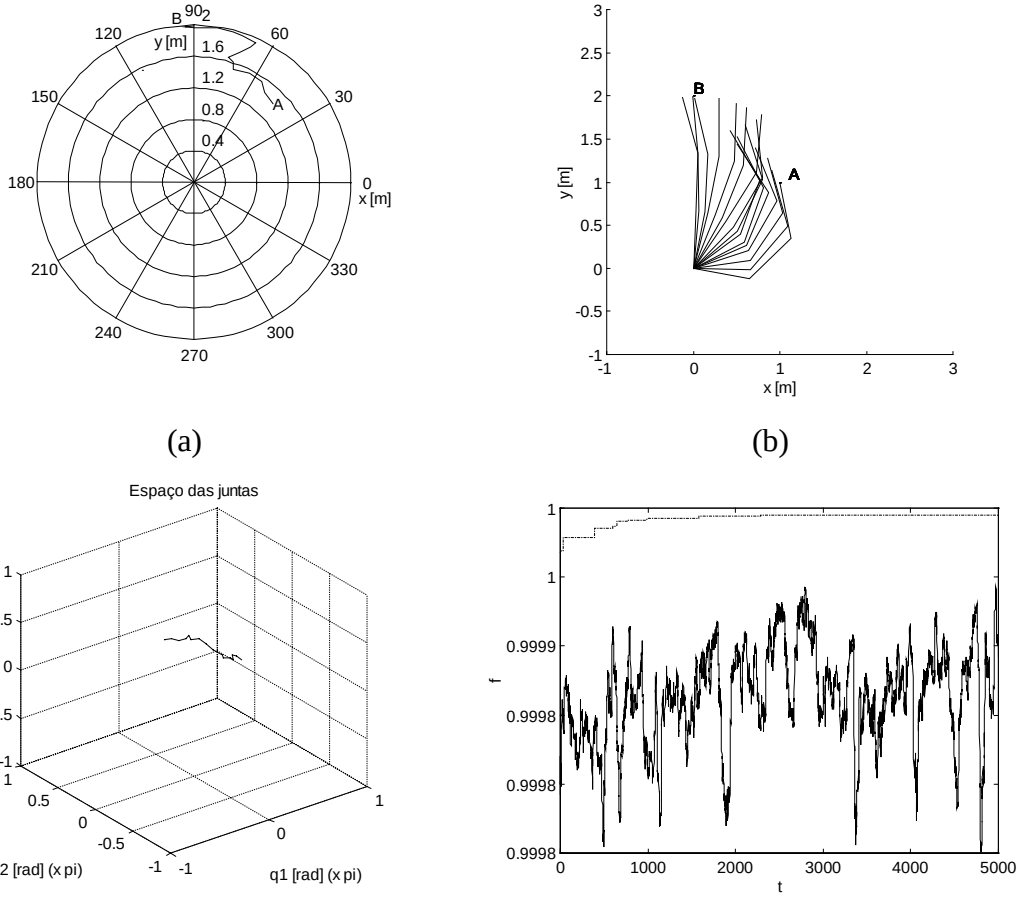
Figura 4.4 Resultados do robô de 2 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a) (b)

(c) (d)

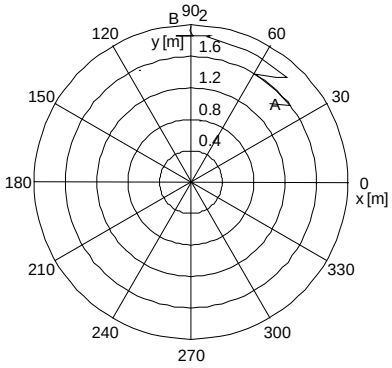
Figura 4.5 Resultados do robô de 3 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

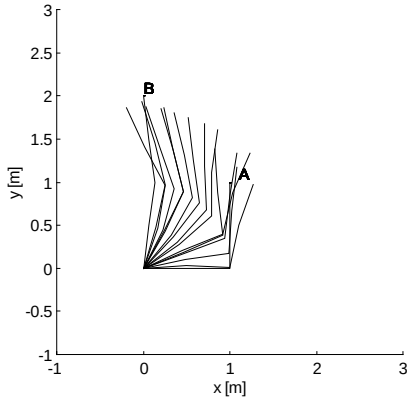
(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

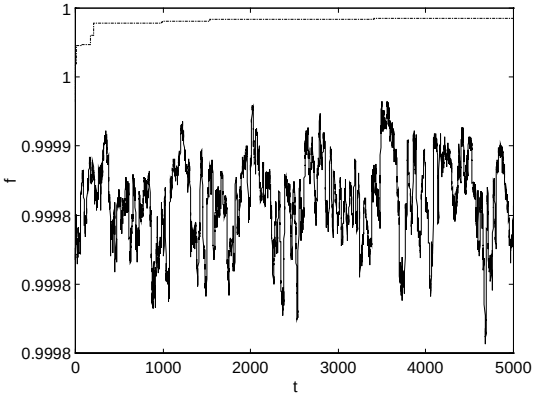
(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a)



(b)



(c)

Figura 4.6 Resultados do robô de 4 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

4.3.3 Influência do operador de mutação

Na figura 4.7 pode verificar-se que a qualidade da solução encontrada diminui com o aumento da probabilidade de mutação. O problema só tem um óptimo local e o aumento da probabilidade de mutação faz diminuir a aptidão da solução encontrada. Com o aumento da

probabilidade de mutação a média do valor de aptidão da população diminui (ver figura 4.7 (d)).

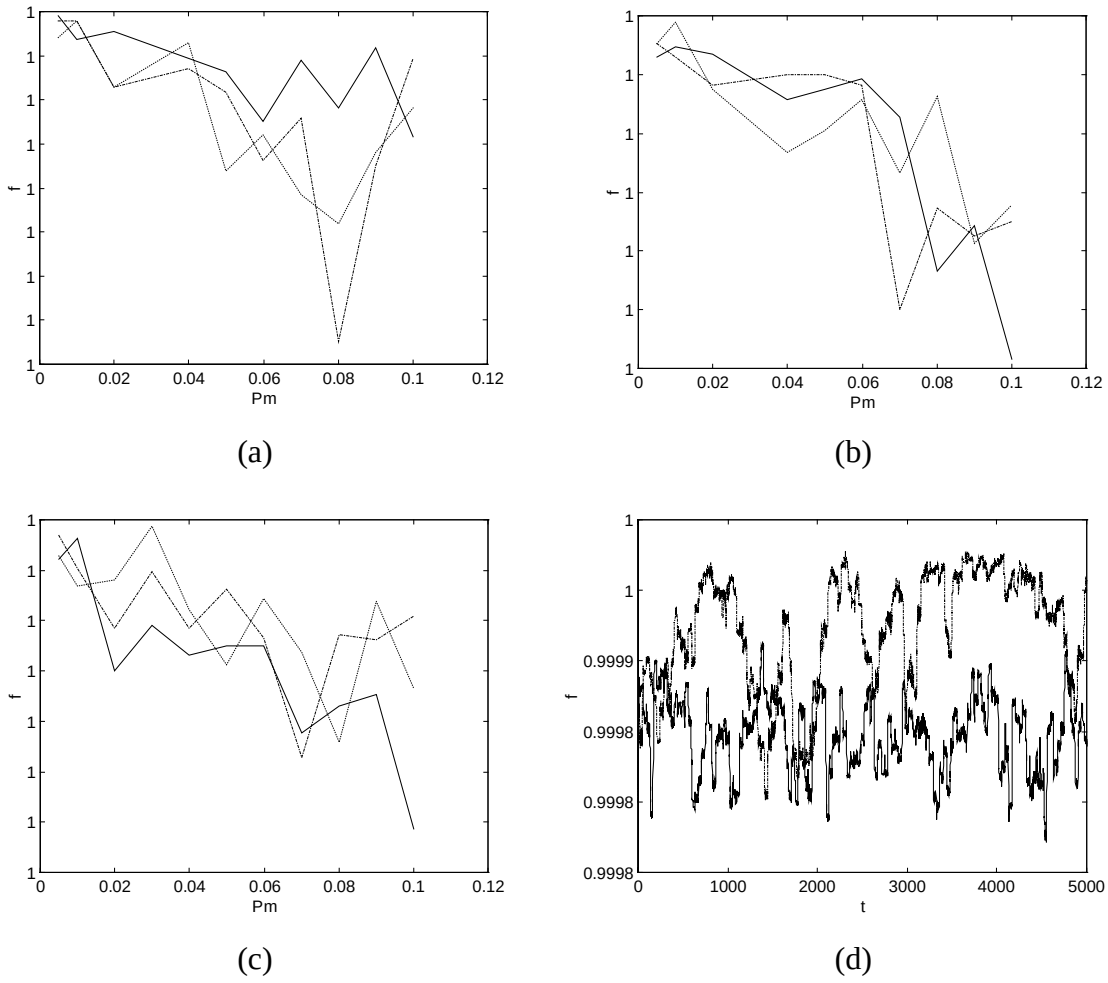


Figura 4.7 Resultado de várias experiências.

(a) Experiências para o robô de dois elos

(b) Experiências para o robô de três elos

(c) Experiências para o robô de quatro elos

(d) Valor médio da população para robô de 2 elos (p_m = 0,005 (– – –); p_m = 0,1 (——))

4.4 Variação da probabilidade de cruzamento

4.4.1 Introdução

Foi simulado o mesmo problema da secção 4.2, fixou-se o valor da probabilidade de mutação em $p_m = 0,005$ e variou-se a probabilidade de cruzamento de $p_c = 0$ até $p_c = 1$ com incrementos de 0,1.

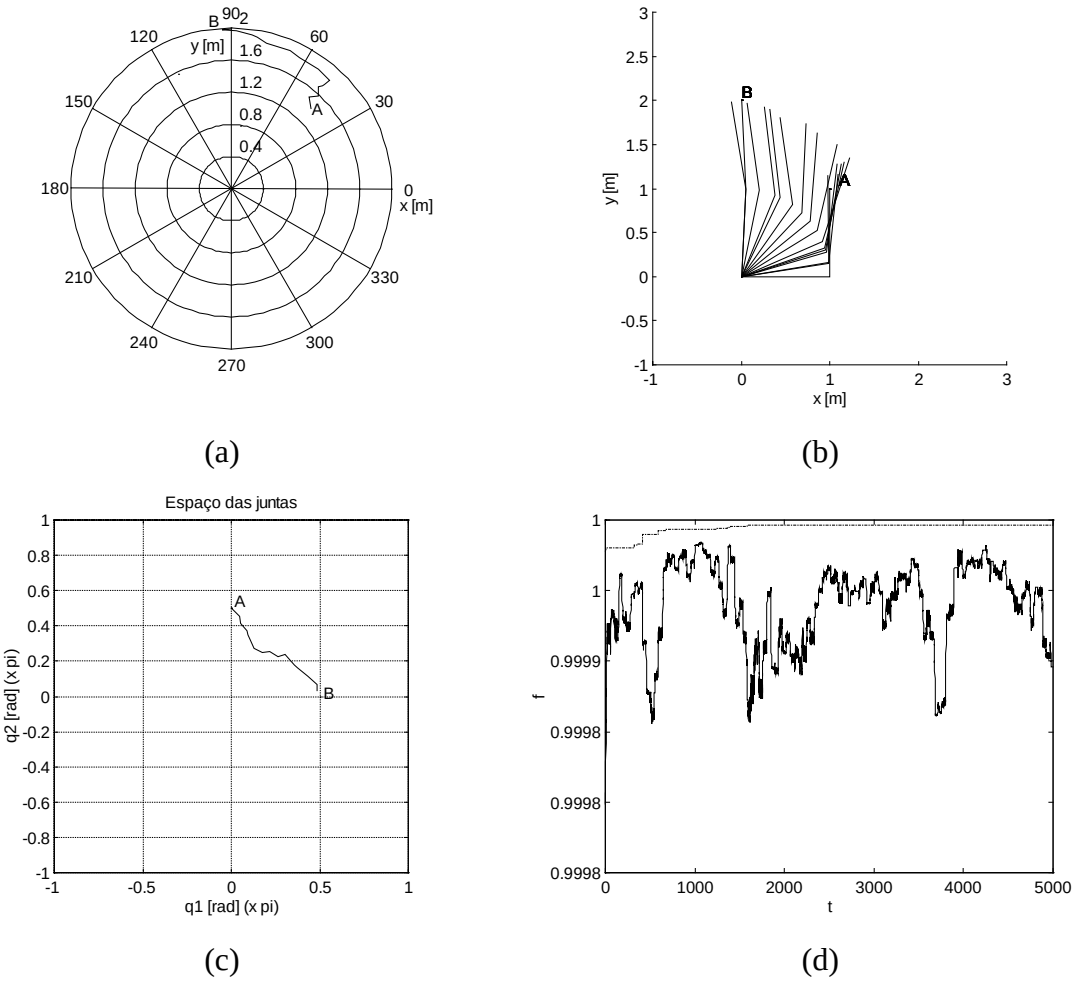
4.4.2 Resultados das experiências

Na figura 4.8 e na figura 4.9 são apresentados os resultados de duas experiências para o robô de 2 elos com probabilidade de cruzamento, respectivamente, de $p_c = 0,1$ e $p_c = 0,8$. Na figura 4.10 e na figura 4.11 encontram-se os resultados de experiências para robôs com 3 e 4 elos, com uma probabilidade de cruzamento de $p_c = 0,8$.

Na tabela 4.2 encontram-se os valores de aptidão das experiências realizadas assim como o número da geração onde as soluções foram encontradas.

Tabela 4.2 Valores de aptidão das soluções ($p_m = 0,005$).

Número de elos do robô	Aptidão da solução	Geração onde foi encontrada a melhor solução
2 ($p_c = 0,1$)	0,9999965	3892
2 ($p_c = 0,8$)	0,9999966	722
3 ($p_c = 0,8$)	0,9999951	4127
4 ($p_c = 0,8$)	0,9999944	4618



(a) (b)

(c) (d)

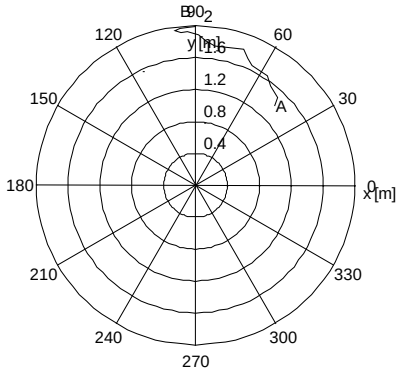
Figura 4.8 Resultados do robô de 2 elos (p_c = 0,1; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

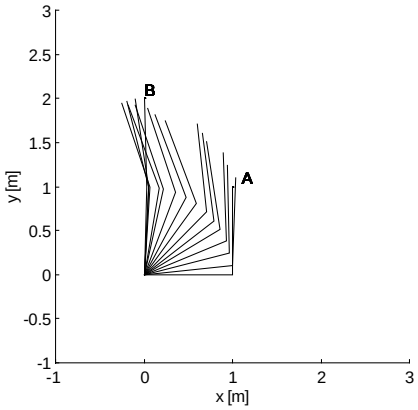
(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

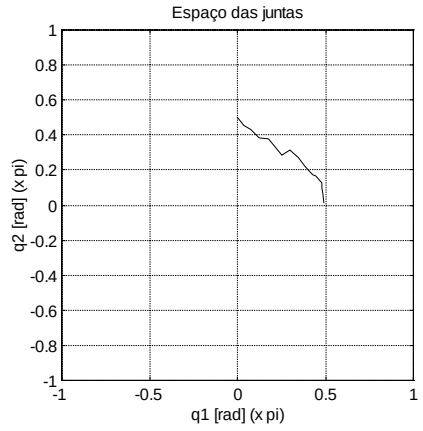
(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



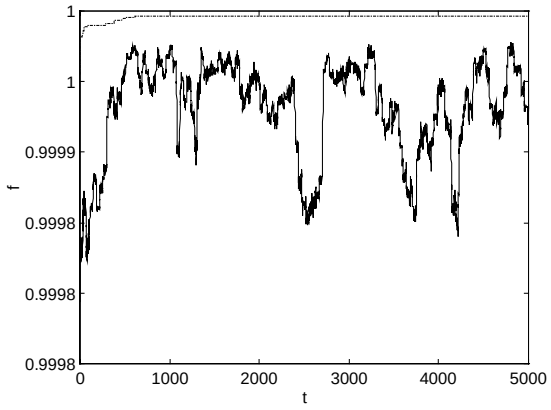
(a)



(c)



(c)



(d)

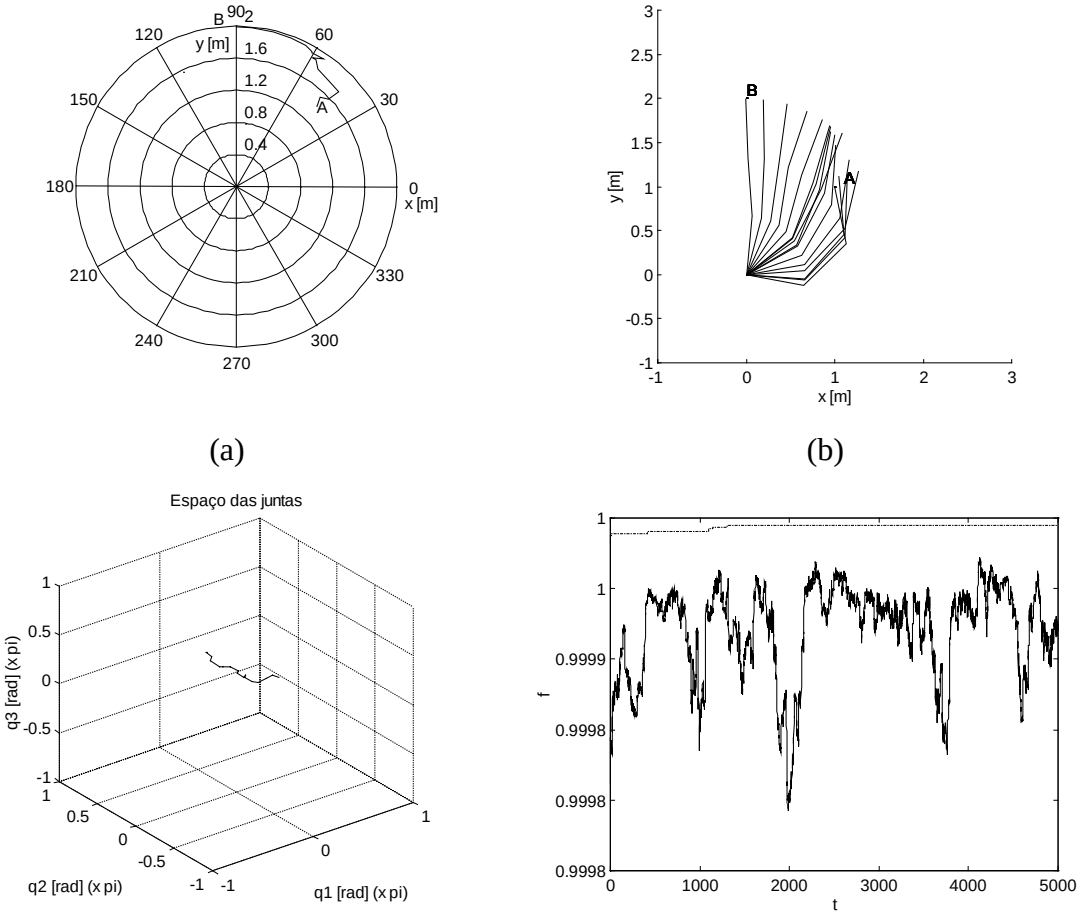
Figura 4.9 Resultados do robô de 2 elos (p_c = 0,8; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a) (b)

(c) (d)

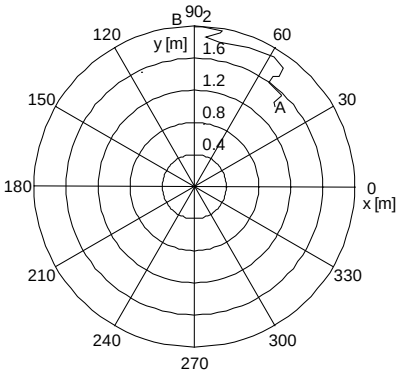
Figura 4.10 Resultados do robô de 3 elos (p_c = 0,8; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

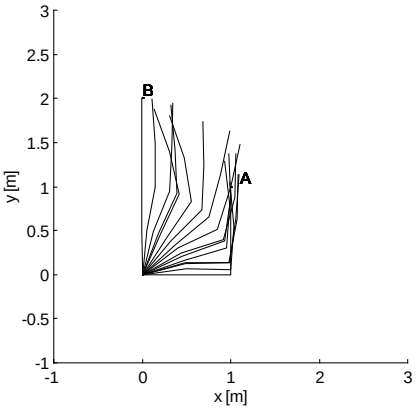
(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

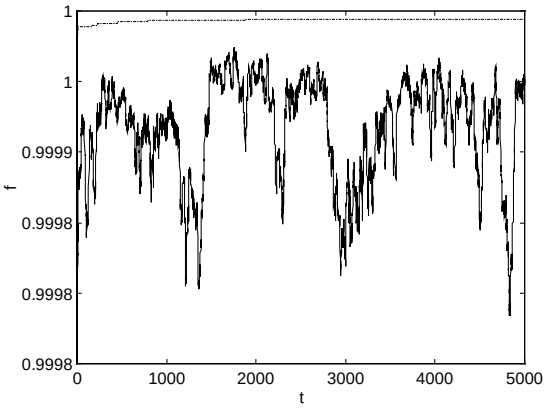
(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a)



(b)



(c)

Figura 4.11 Resultados do robô de 4 elos (p_c = 0,8; p_m = 0,005).

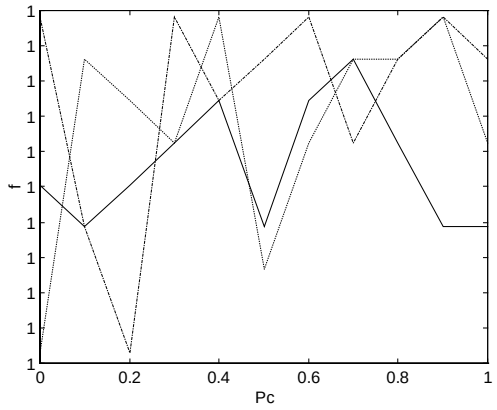
(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

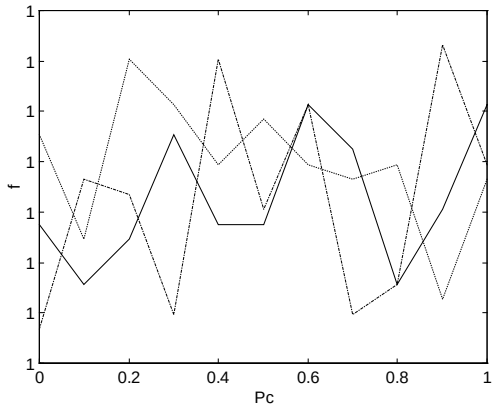
(c) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

4.4.3 Influência do operador de cruzamento

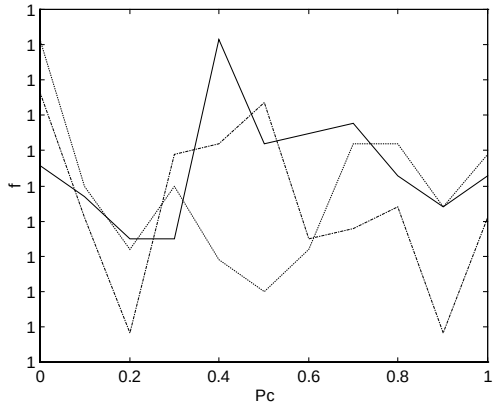
Na figura 4.12 pode verificar-se que a probabilidade de cruzamento tem pouca influência no algoritmo utilizado. Assim, a solução depende essencialmente do operador de selecção e de mutação.



(a)



(b)



(c)

Figura 4.12 Resultado de várias experiências com diferentes probabilidades de cruzamento.

(a) Experiências para o robô de dois elos;

(b) Experiências para o robô de três elos

(c) Experiências para o robô de quatro elos

4.5 Variação dos coeficientes da função de aptidão

4.5.1 Introdução

Repetiu-se a simulação do problema descrito na secção 4.2 variando os coeficientes da função de aptidão

$$f = e^{-\alpha_1 \times |PP-PF| - \alpha_2 \times \sum_{i=1}^{n} \Delta q_i} \tag{4.3}$$

para $(\alpha_1, \alpha_2, \alpha_3) = (10; 0{,}1; 0)$ de modo a obter uma pressão de selecção mais elevada.

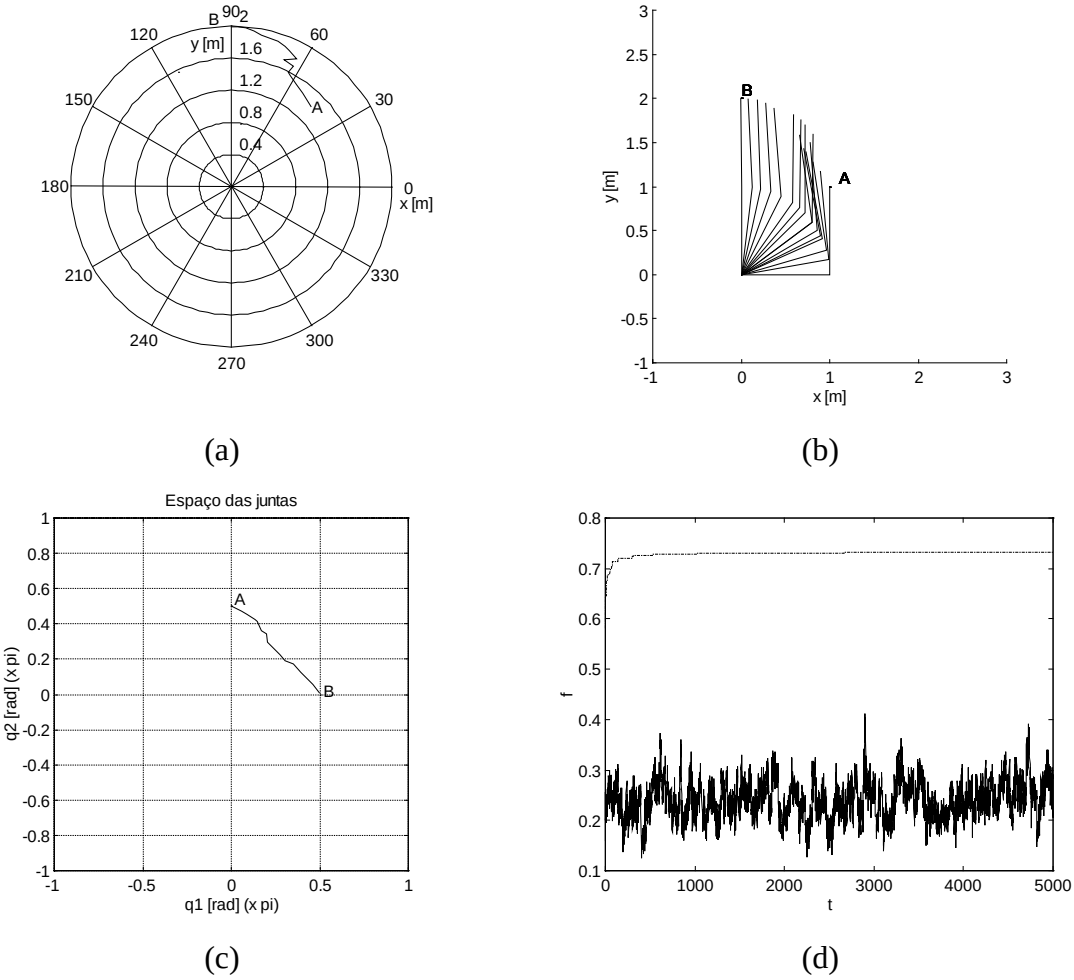
4.5.2 Resultado das experiências

Na figura 4.13 à figura 4.15 encontram-se os resultados para o robô com 2, 3 e 4 elos para uma probabilidade de cruzamento e mutação respectivamente de $p_c = 0{,}8$ e de $p_m = 0{,}03$.

Na tabela 4.3 encontram-se os valores de aptidão das experiências realizadas assim como o número da geração onde as soluções foram encontradas.

Tabela 4.3 Valores de aptidão das soluções ($p_c = 0{,}8$; $p_m = 0{,}03$).

Número de elos do robô	Aptidão da solução	Geração onde foi encontrada a melhor solução
2	0,7318515	3577
3	0,685596	4151
4	0,6251428	2450



(a) (b)

(c) (d)

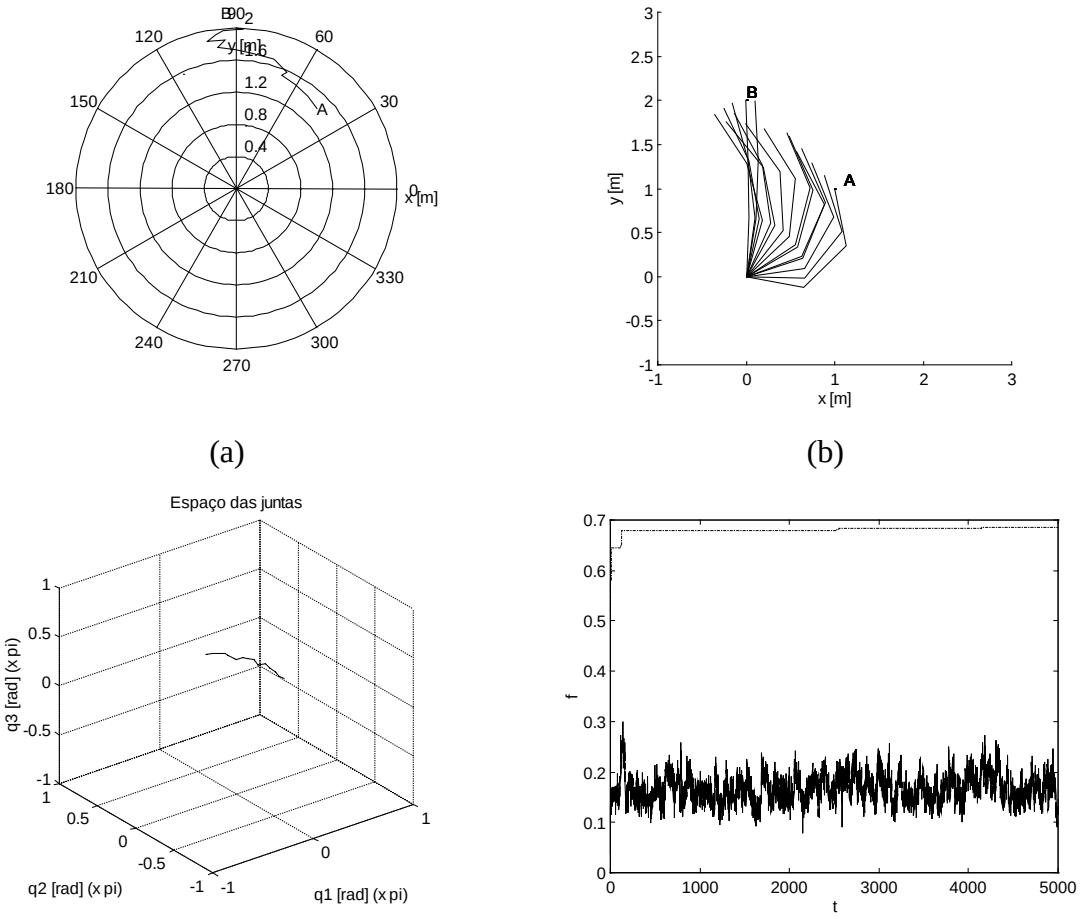
Figura 4.13 Resultados do robô de 2 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a) (b)

(c) (d)

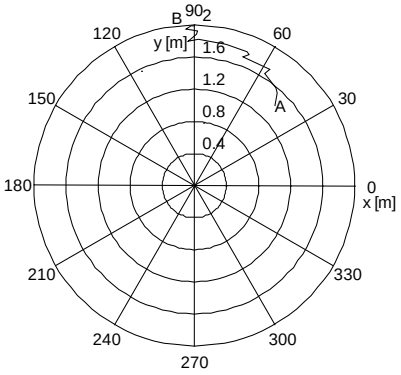
Figura 4.14 Resultados do robô de 3 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

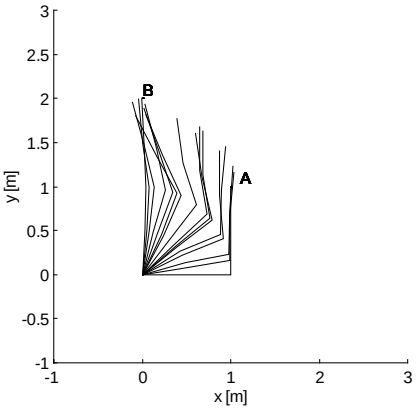
(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

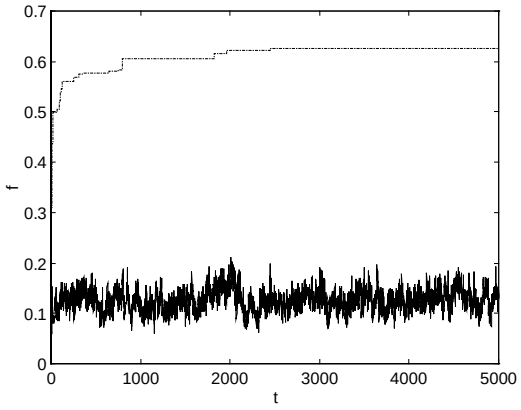
(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a)



(b)



(c)

Figura 4.15 Resultados do robô de 4 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

4.5.3 Conclusões

À medida que se aumenta a importância dos coeficientes que "pesam" no erro final e do custo da trajectória na função de aptidão, o erro final e o custo da trajectória diminuem.

Consequentemente, a trajectória apresenta um comportamento mais estável. No entanto, o aumento da pressão de selecção pode provocar à convergência prematura do algoritmo.

4.6 Teste com um obstáculo simples

4.6.1 Introdução

Realizou-se novamente o planeamento de trajectórias para o manipulador robótico desde o ponto (1, 1) para o ponto (0, 2). O ambiente tem agora um obstáculo formado por um rectângulo com as coordenadas dos cantos superior direito e inferior esquerdo respectivamente (0,5562; 1,7119) e (1,2728; 1,2728). Por outro lado, a função de aptidão vem:

$$f = e^{-10^{-4}\times|PP-PF|-10^{-6}\times\sum_{i=1}^{15}\Delta q_i-10^{-4}\times nPNA} \qquad (4.4)$$

onde:

PP é o ponto que se pretende atingir pelo robô;

PF é o ponto atingido pelo robô após a simulação;

Δq_i representa a actuação de uma junta i do robô;

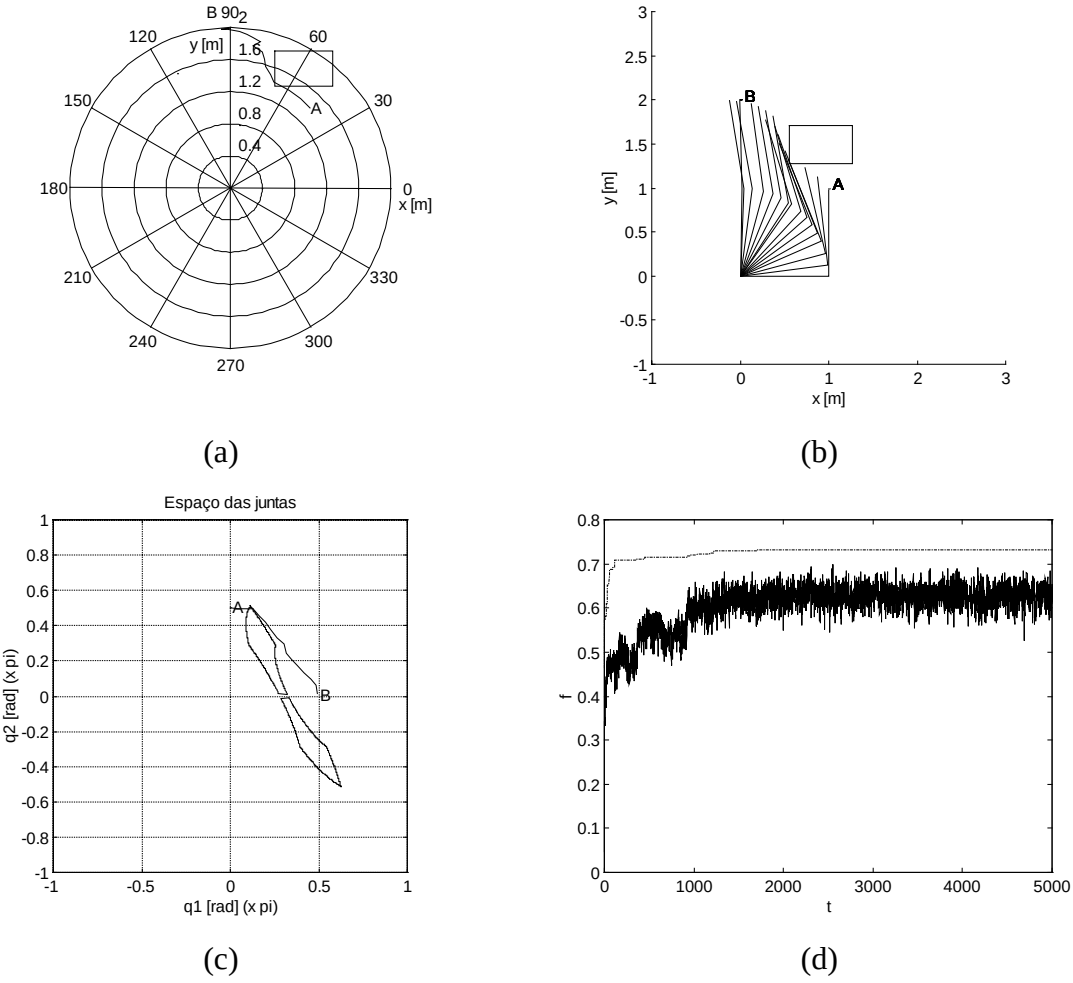
nPNA é o número de pontos não admissíveis.

A probabilidade de cruzamento e de mutação é respectivamente de $p_c = 0{,}6$ e de $p_m = 0{,}005$.

4.6.2 Resultado das experiências

Na figura 4.16 à figura 4.18 mostra-se os resultados das experiências para os robôs de 2, 3 e 4 elos. De notar que as trajectórias obtidas se sobrepõem ligeiramente aos obstáculos. Isto é devido a considerar-se o robô constituído por um conjunto de pontos discretos e o desvio máximo dos ângulos das juntas entre dois pontos consecutivos ser de 1,8 [rad]. Por esse motivo o obstáculo modelizado no problema entra com uma margem de segurança em

relação ao obstáculo real. A margem de segurança depende do desvio máximo e do número de ponto em que o manipulador é dividido.



(a) (b)

(c) (d)

Figura 4.16 Resultados do robô de 2 elos (p_c = 0,6; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

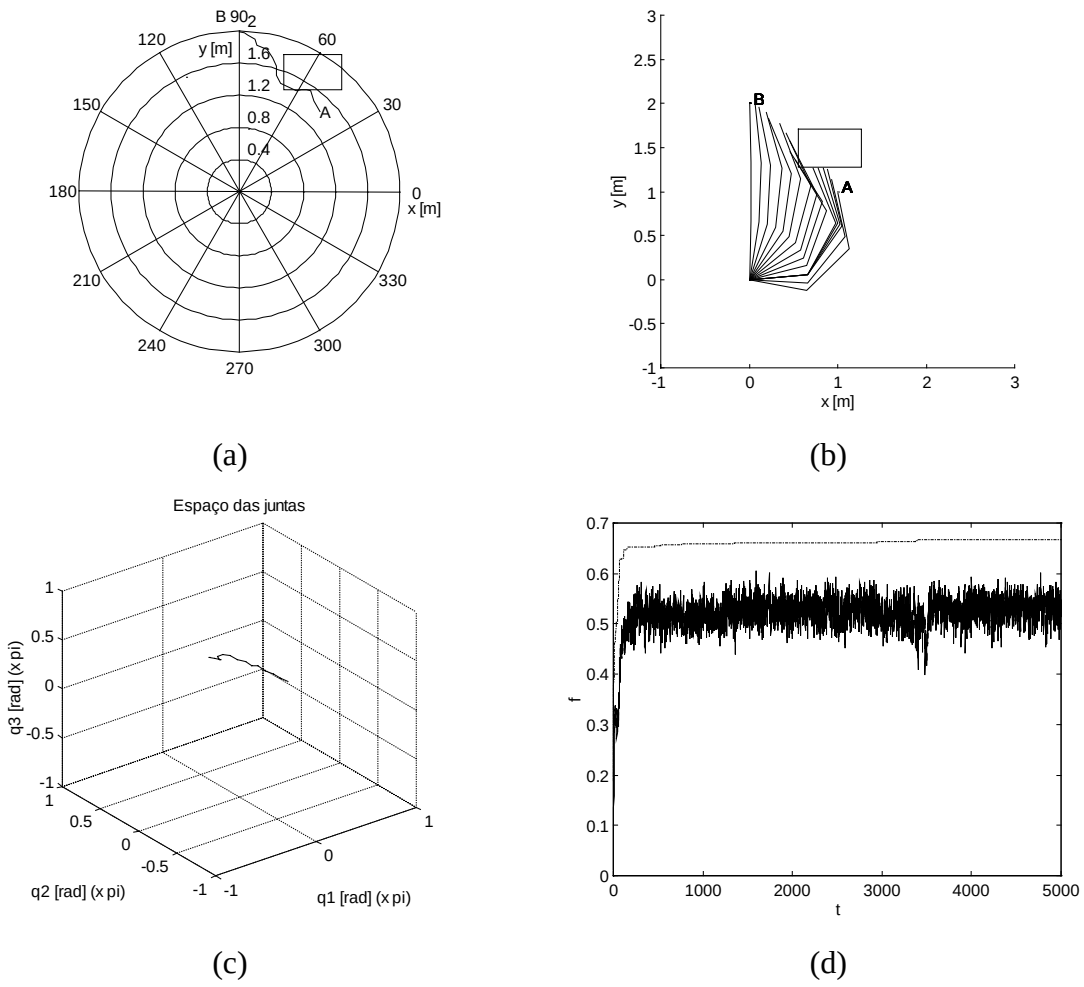


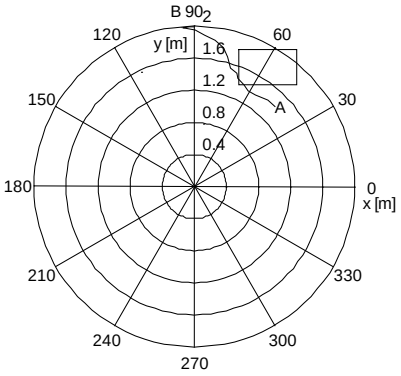
Figura 4.17 Resultados do robô de 3 elos (p_c = 0,6; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

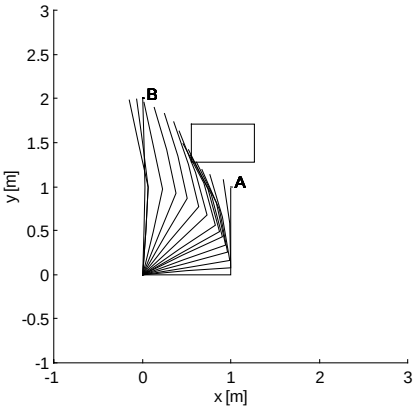
(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

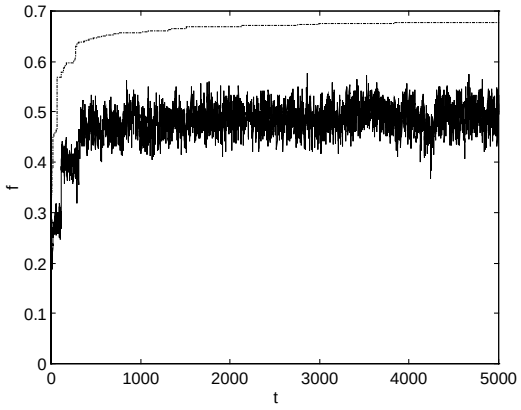
(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a)



(b)



(c)

Figura 4.18 Resultados do robô de 4 elos (p_c = 0,6; p_m = 0,005).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

Na tabela 4.4 encontram-se os valores de aptidão das experiências realizadas assim como o número da geração onde as soluções foram encontradas.

Tabela 4.4 Valores de aptidão das soluções (p_c = 0,6; p_m = 0,005).

Número de elos do robô	Aptidão da solução	Geração onde foi encontrada a melhor solução
2	0,73165965	3783
3	0,6675458	4647
4	0,677814	4659

4.7 Variação do comprimento do cromossoma

4.7.1 Introdução

Nesta secção são feitos dois tipos de experiências com vista a analisar o efeito do comprimento do cromossoma. Na secção 4.7.2 é realizado um conjunto de testes num espaço sem obstáculos enquanto que na secção 4.7.3 é considerada a presença de obstáculos.

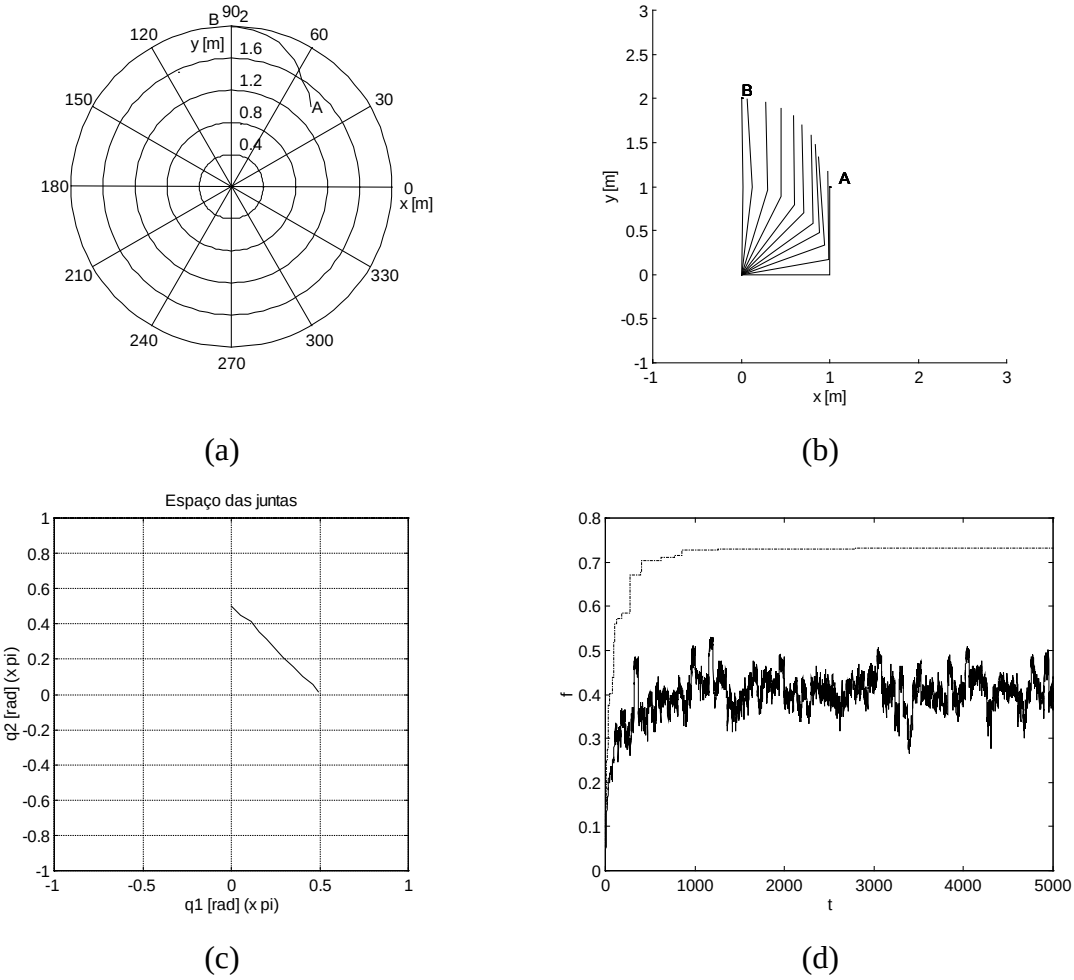
4.7.2 Ambiente sem obstáculos

4.7.2.1 Introdução

As experiências realizadas utilizam cromossomas com comprimento de 10 genes, uma probabilidade de cruzamento e mutação respectivamente de p_c = 0,8 e de p_m = 0,03 e uma função de aptidão conforme indicada na equação (4.3).

4.7.2.2 Resultado das experiências

Na figura 4.19 à figura 4.21 encontram-se os resultados das experiências para os robôs de 2, 3 e 4 elos.



(a) (b)

(c) (d)

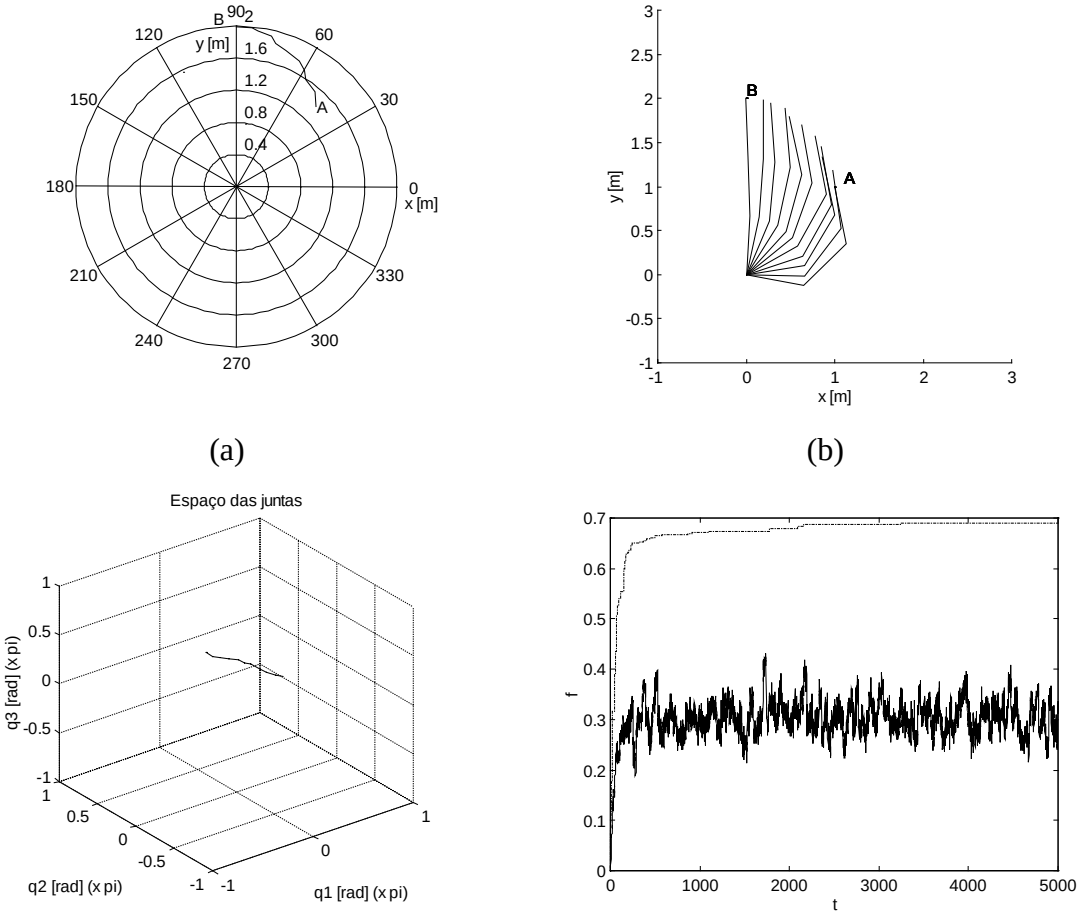
Figura 4.19 Resultados do robô de 2 elos (p_c =0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a) (b)

(c) (d)

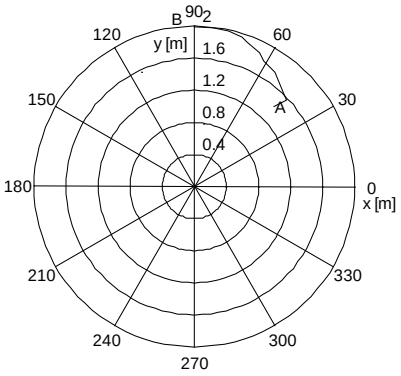
Figura 4.20 Resultados do robô de 3 elos (p_c =0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

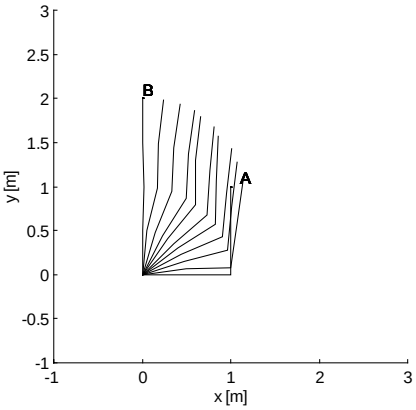
(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Trajectória do robô no espaço das juntas

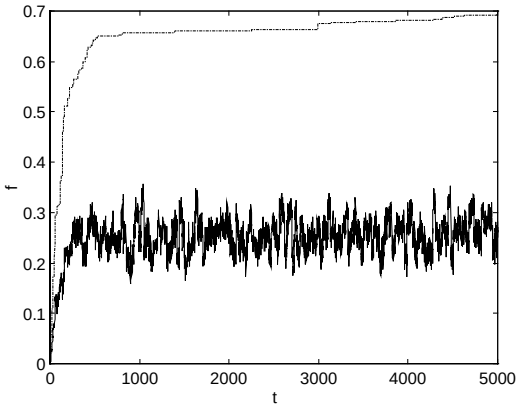
(d) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população



(a)



(b)



(c)

Figura 4.21 Resultados do robô de 4 elos (p_c =0,8; p_m = 0,03).

(a) Trajectória do órgão terminal entre o ponto inicial A e o ponto final B no espaço operacional

(b) As sucessivas posições do robô entre o ponto inicial A e ponto final B no espaço operacional

(c) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

Na tabela 4.5 encontram-se os valores de aptidão das experiências realizadas assim como o número da geração onde as soluções foram encontradas.

Tabela 4.5 Valores de aptidão das soluções (p_c = 0,6; p_m = 0,03).

Número de elos do robô	Aptidão da solução	Geração onde foi encontrada a melhor solução
2	0,7316837	4165
3	0,689519	4104
4	0,693464	4987

4.7.3 Ambiente com obstáculos

4.7.3.1 Introdução

Nesta secção estuda-se a variação do comprimento do cromossoma em ambientes com obstáculos. Para esse fim é simulado o deslocamento do manipulador robótico do ponto (1, 1) para o ponto (0, 2). O ambiente é composto por um obstáculo constituído por um rectângulo com coordenadas dos cantos superior esquerdo e inferior direito respectivamente (0,3; 2) e (0,75; 0,8) e a função de aptidão usada é a indicada na equação (4.4).

A probabilidade de cruzamento e de mutação é respectivamente de $p_c = 0,6$ e de $p_m = 0,005$ e o número de gerações é de $\mu = 500$.

4.7.3.2 Resultado das experiências

Foram realizadas várias simulações para o robô de 2 elos adoptando comprimentos de cromossomas de $l = 15$, $l = 20$, $l = 25$, $l = 30$, $l = 35$ e $l = 40$ genes. Os resultados obtidos para as experiências com cromossomas de tamanho $l = 15$, $l = 20$ e $l = 25$ foram muito semelhantes (figura 4.22 (a)). Identicamente para as experiências com cromossomas de $l = 30$ e $l = 35$ genes os resultados são semelhantes (figura 4.22 (b)). Na figura 4.22 (c) e na figura 4.22 (d) é ilustrado o resultado da experiência com o cromossoma de comprimento de $l = 40$ genes. Apesar da trajectória da simulação com o cromossoma de $l = 30$ genes não se afastar tanto do ponto final como a simulação com o cromossoma de $l = 40$ genes, aquela tem um custo superior visto que o segundo ângulo do robô se desloca cerca de $1,5 \times \pi$ ra-

dianos enquanto que o ângulo deste apenas tem um deslocamento aproximado de $0{,}5 \times \pi$ radianos.

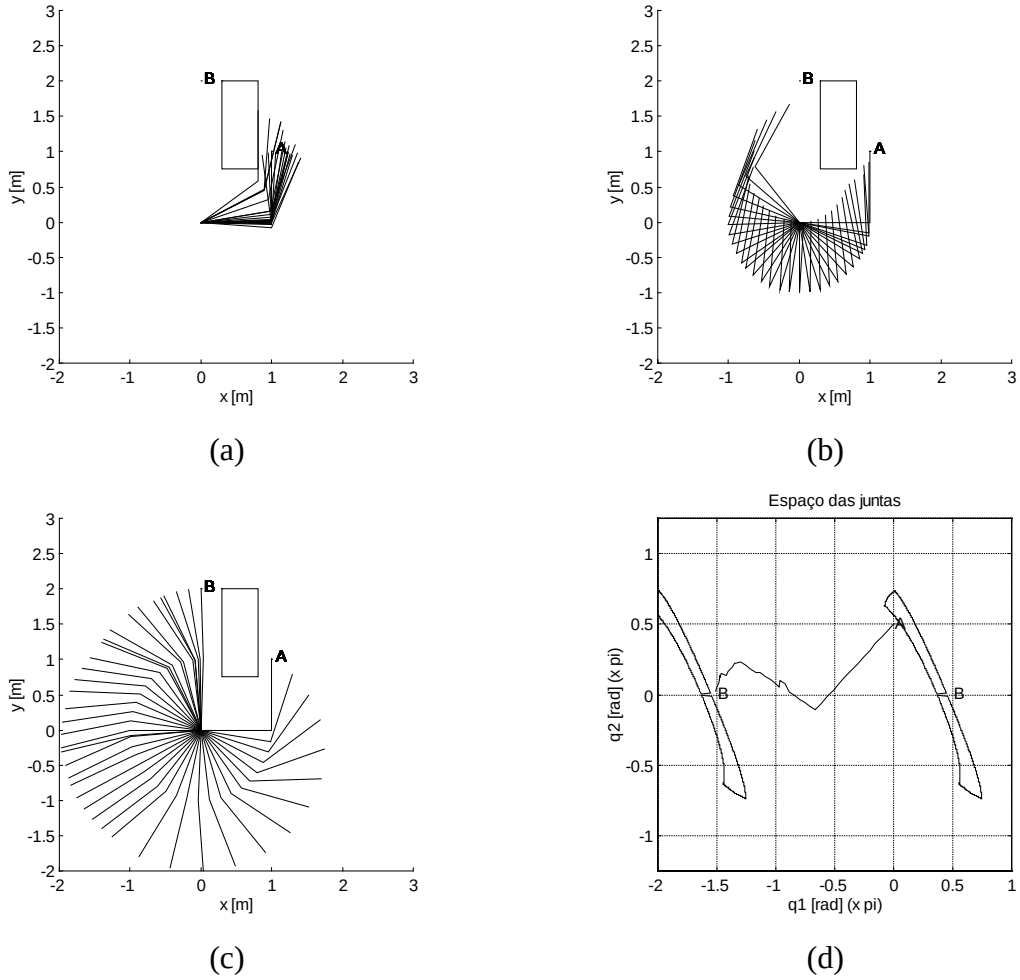


Figura 4.22 Resultado para o robô de 2 elos (p_c = 0,6; p_m = 0,005).

(a) l = 20, no espaço operacional

(b) l = 30, no espaço operacional

(c) l = 40, no espaço operacional

(d) l = 40, no espaço das juntas

4.7.3.3 Simulação para os robô de três e quatro elos

Os resultados obtidos para as simulações com um robô de três e quatro elos são semelhantes aos resultados para o robô de dois elos (figura 4.23).

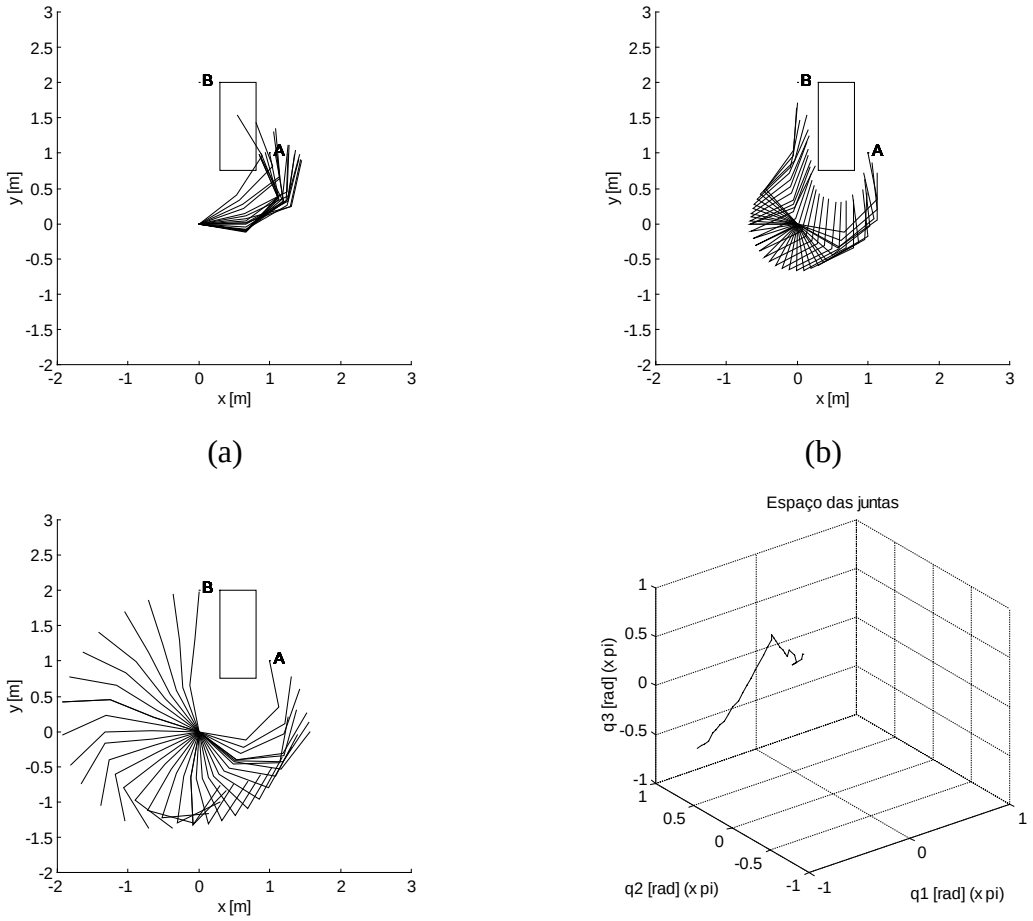


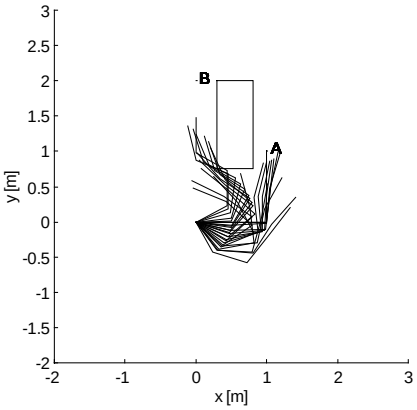
Figura 4.23 Resultados para o robô de 3 e 4 elos (p_c = 0,6; p_m = 0,005).

(a) robô com 3 elos, *l* = 20, no espaço operacional

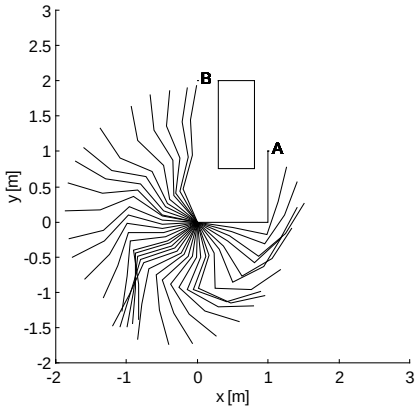
(b) robô com 3 elos, *l* = 30, no espaço operacional

(c) robô com 3 elos, *l* = 35, no espaço operacional

(d) robô com 3 elos, *l* = 35, no espaço das juntas



e)



f)

Figura 4.23 Resultados para o robô de 3 e 4 elos (p_c = 0,6; p_m = 0,005) (continuação).
(e) robô com 4 elos, l = 25, no espaço operacional
(f) robô com 4 elos, l = 35, no espaço operacional

4.7.4 Conclusões

Para ambientes sem obstáculos, quanto menor é menor o número de genes melhor é o comportamento da trajectória (comparar a figura 4.13, figura 4.14 e figura 4.15 respectivamente com a figura 4.19, figura 4.20 e figura 4.21). Por outro lado, o cromossoma deve ter sempre o número de genes necessário para a trajectória atingir o ponto final. No entanto, independentemente do comprimento do cromossoma, o ponto final é sempre alcançável.

Para ambientes com obstáculos, e à medida que o ambiente de trabalho do robô se torna mais "difícil", é necessário utilizar cromossomas com um número superior de genes. Pode ser feita uma analogia com a genética natural onde em ambiente complexos normalmente só sobrevive o organismo com o maior número de cromossomas.

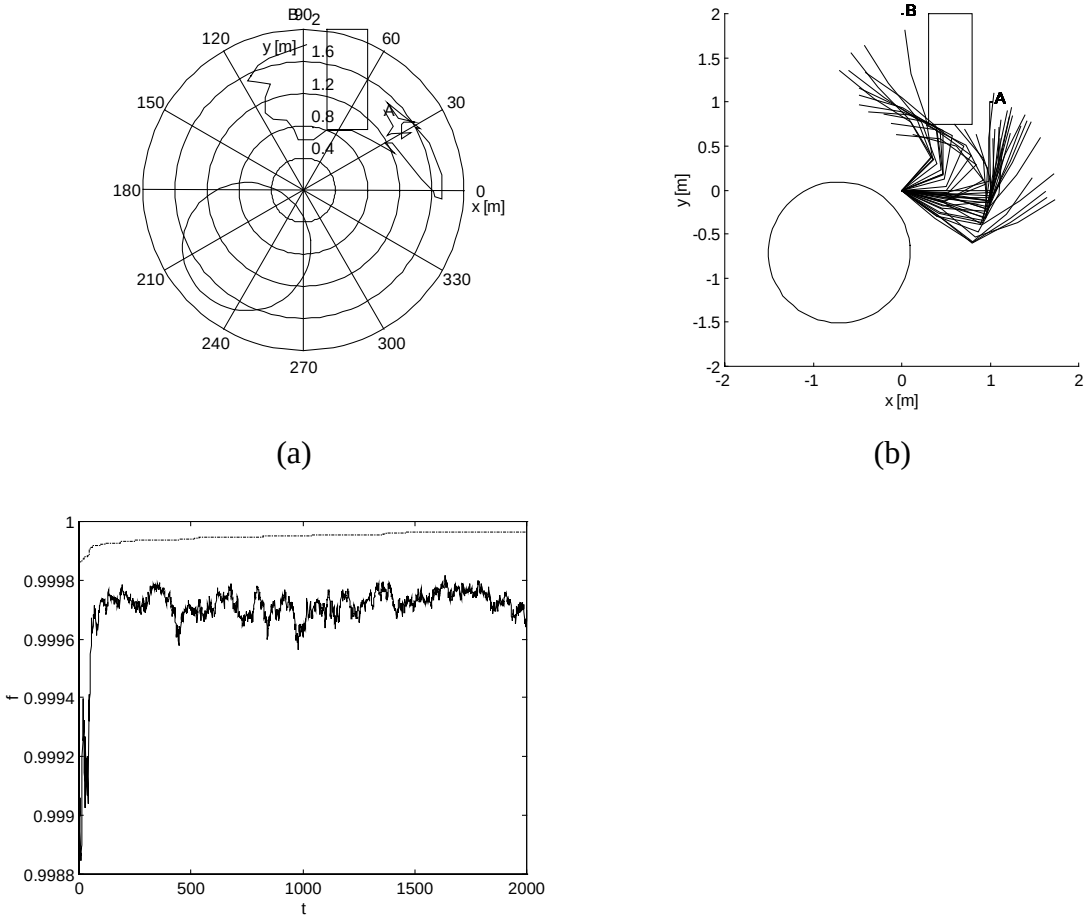
4.8 Simulação com dois obstáculos

4.8.1 Introdução

Nesta secção é apresentada a trajectória para o robô de 4 elos num ambiente com dois obstáculos. O ambiente é constituído por um rectângulo de cantos superior direito e inferior

esquerdo respectivamente (0,3; 2) e (0,75; 0,8) e por um circulo de centro (–0,707; –0,707) e raio 0,8 [m]. A probabilidade de cruzamento e de mutação é respectivamente de p_c = 0,6 e de p_m = 0,005 e o número de gerações é de μ = 500.

4.8.2 Resultado da experiência com dois obstáculos



(a) (b)

(c)

Figura 4.24 Resultados para robô com 4 elos em ambiente com 2 obstáculos (p_c = 0,6; p_m = 0,005).

(a) Trajectória do órgão terminal

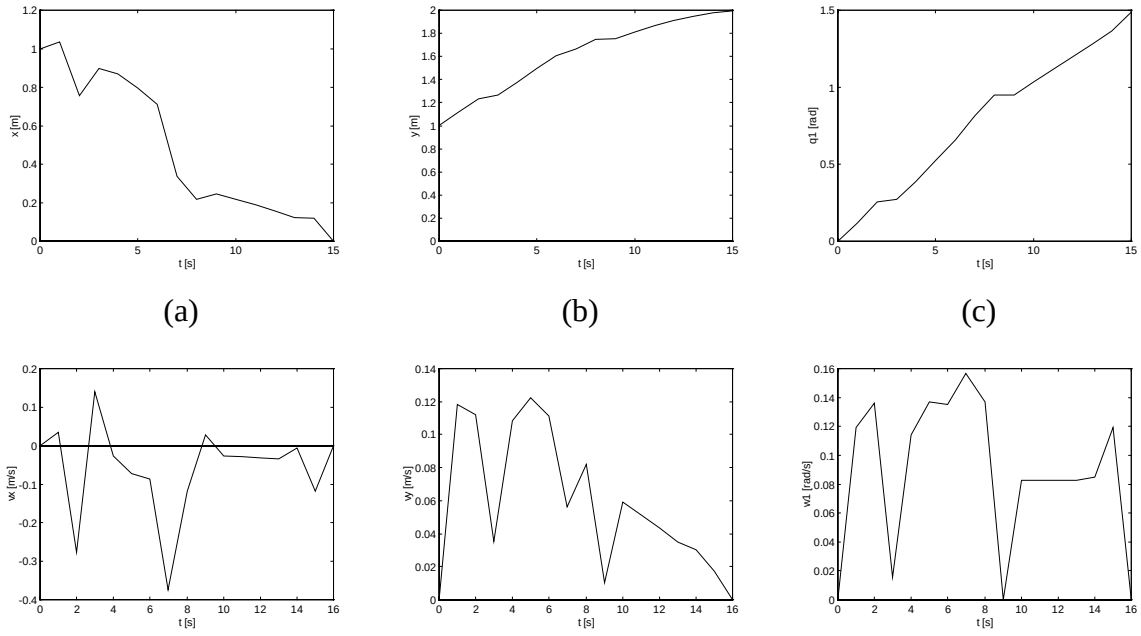
(b) As sucessivas posições do robô entre o ponto inicial A e o ponto final B, no espaço operacional

(c) Evolução do melhor cromossoma (– – –) e da média dos cromossomas (——) da população

Como se pode ver pela figura 4.24, a trajectória não alcança o ponto B. Para tal o algoritmo deve utilizar cromossomas de comprimento superior.

4.9 Ondulação residual

As simulações realizadas nas secções anteriores têm uma ondulação residual (*ripple*) considerável. Por exemplo, na figura 4.25 à figura 4.27 pode ver-se a evolução temporal das posições e velocidades respeitantes às experiências da secção 4.3 com a probabilidade de mutação de $p_m = 0{,}03$.



(a) (b) (c)

(d) (e) (f)

Figura 4.25 Posições e velocidades para o robô de 2 elos (problema da secção 4.3, $p_c = 0{,}8$; $p_m = 0{,}03$).

(a) Deslocamento do órgão terminal – eixo X

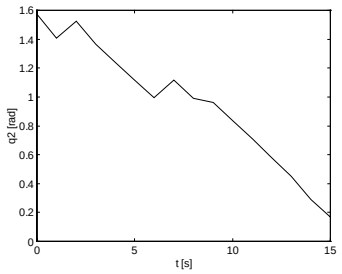
(b) Deslocamento do órgão terminal – eixo Y

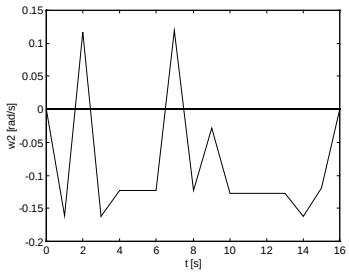
(c) Deslocamento da junta 1

(d) Velocidade do órgão terminal – eixo X

(e) Velocidade do órgão terminal – eixo Y

(f) Velocidade da junta 1

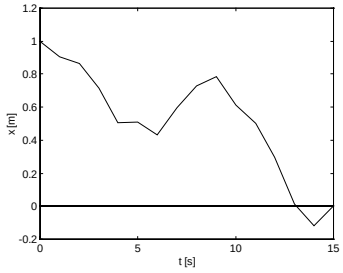


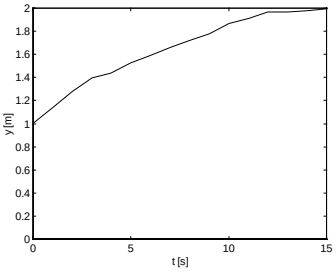


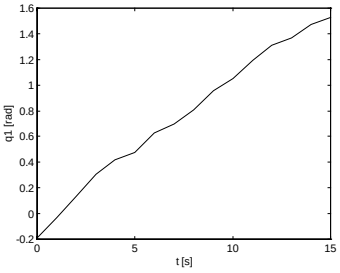
(g) (h)

Figura 4.25 Posições e velocidades para o robô de 2 elos (continuação).

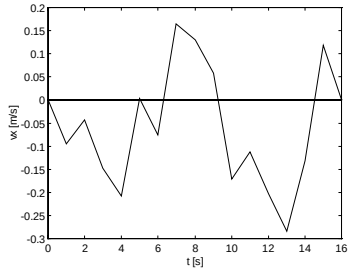
(g) Deslocamento da junta 2; (h) velocidade da junta 2.

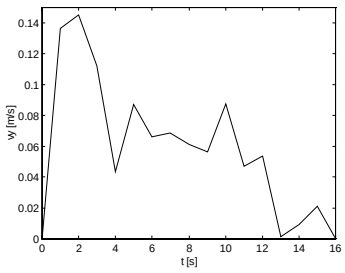


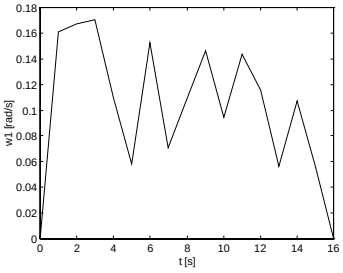




(a) (b) (c)







(d) (e) (f)

Figura 4.26 Posições e velocidades para o robô de 3 elos (problema da secção 4.3, p_c = 0,8; p_m = 0,03).

(a) Deslocamento do órgão terminal – eixo X

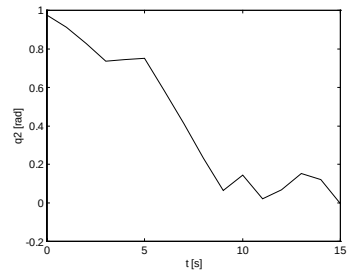
(b) Deslocamento do órgão terminal – eixo Y

(c) Deslocamento da junta 1

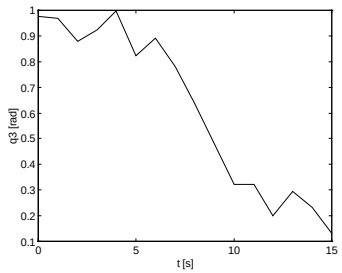
(d) Velocidade do órgão terminal – eixo X

(e) Velocidade do órgão terminal – eixo Y

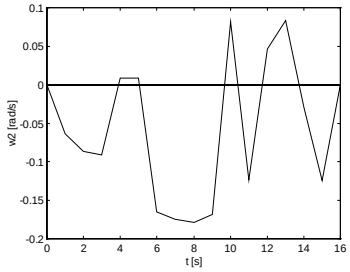
(f) Velocidade da junta 1



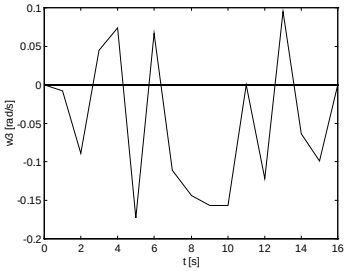
(g)



(h)



(i)



(j)

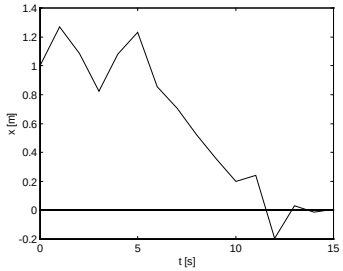
Figura 4.26 Posições e velocidades para o robô de 3 elos (continuação).

(g) Deslocamento da junta 2

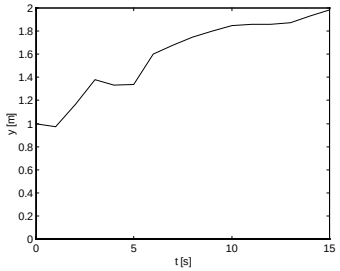
(h) Deslocamento da junta 3

(i) Velocidade da junta 2

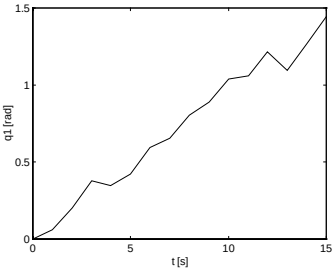
(j) Velocidade da junta 3



(a)



(b)



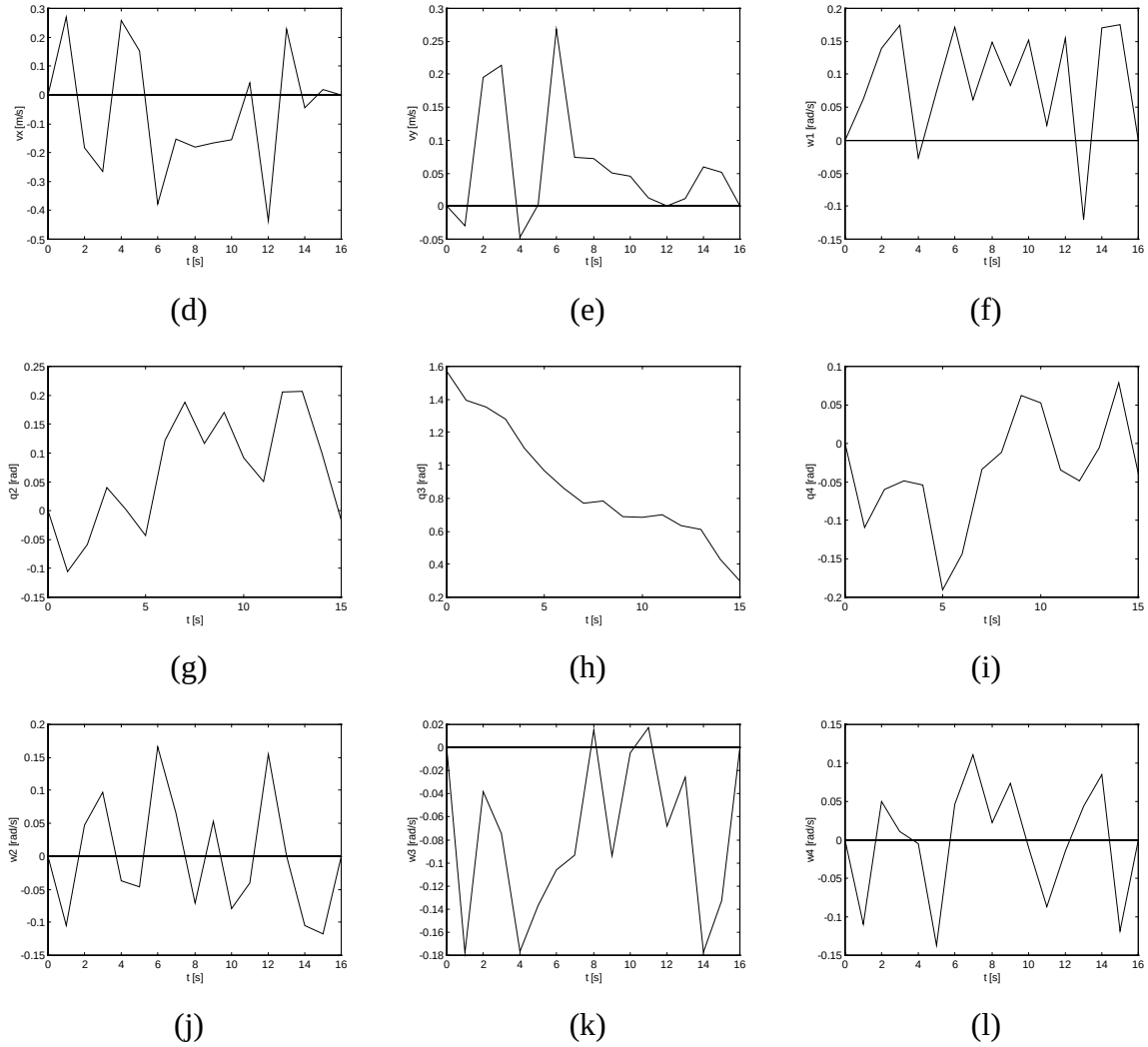
(c)

Figura 4.27 Posições e velocidades para o robô de 4 elos (problema da secção 4.3, p_c = 0,8; p_m = 0,03).

(a) Deslocamento do órgão terminal – eixo X

(b) Deslocamento do órgão terminal – eixo Y

(c) Deslocamento da junta 1



(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figura 4.27 Posições e velocidades para o robô de 4 elos (continuação).

(d) Velocidade do órgão terminal – eixo X

(e) Velocidade do órgão terminal – eixo Y

(f) Velocidade da junta 1

(g) Deslocamento da junta 2

(h) Deslocamento da junta 3

(i) Deslocamento da junta 4

(j) Velocidade da junta 2

(k) Velocidade da junta 3

(l) Velocidade da junta 4

De modo a diminuir a ondulação residual da velocidade introduziu-se um termo adicional na função de aptidão. Este termo consiste nas variações de velocidade que ocorrem nas juntas do robô. Por esta ordem de ideias, a função de aptidão é dada pela seguinte expressão:

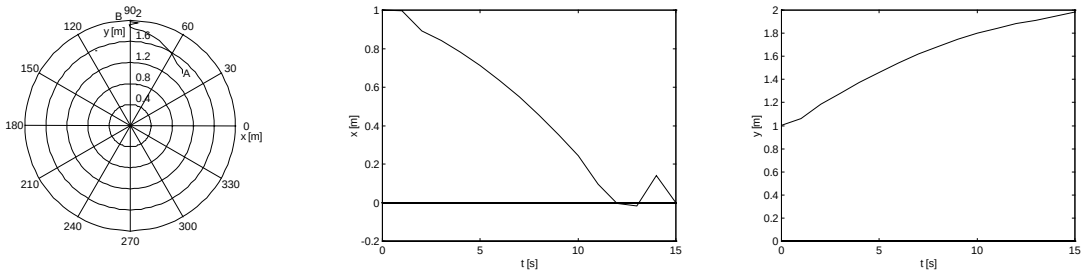
$$f = e^{-10^{-4}\times|PP-PF|-10^{-6}\times\sum_{i=1}^{15}\Delta q_i-10^{-5}\times\sum_{i=1}^{15}\Delta\dot{q}_i} \tag{4.5}$$

onde

$$\Delta\dot{q} = \frac{q_t - q_{t-1}}{T}$$

Nas experiências considerou-se com T = 1 [s].

Simulou-se novamente o problema da secção 4.3 com a função de aptidão (4.5). Os resultados encontram-se na figura 4.28, figura 4.29 e figura 4.30.



(a) (b) (c)

Figura 4.28 Posições e velocidades para o robô de 2 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal no espaço operacional

(b) Deslocamento do órgão terminal – eixo X

(c) Deslocamento do órgão terminal – eixo Y

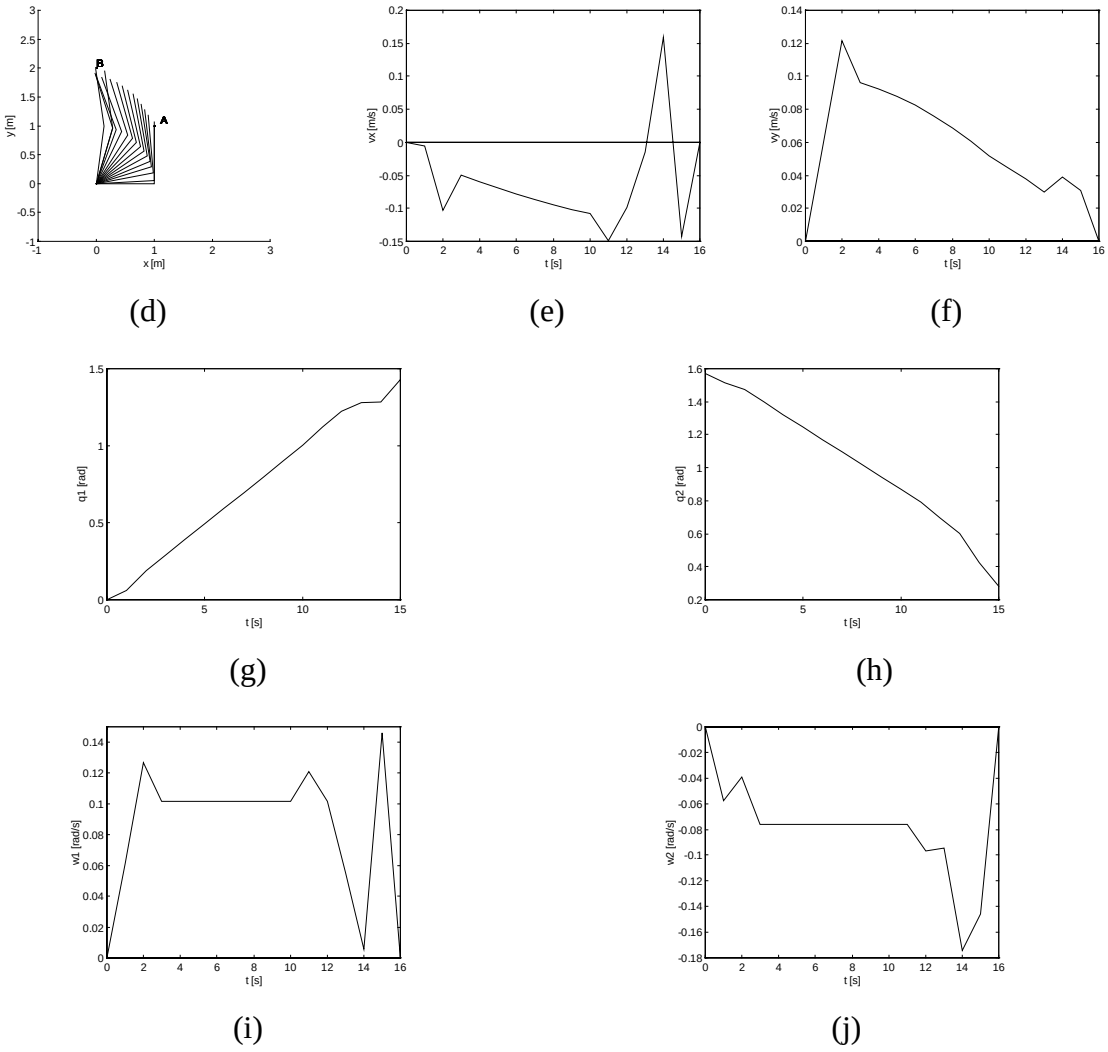


Figura 4.28 Posições e velocidades para o robô de 2 elos (continuação).

(d) As sucessivas posições do robô entre o ponto inicial A e o ponto final B

(e) Velocidade do órgão terminal – eixo X

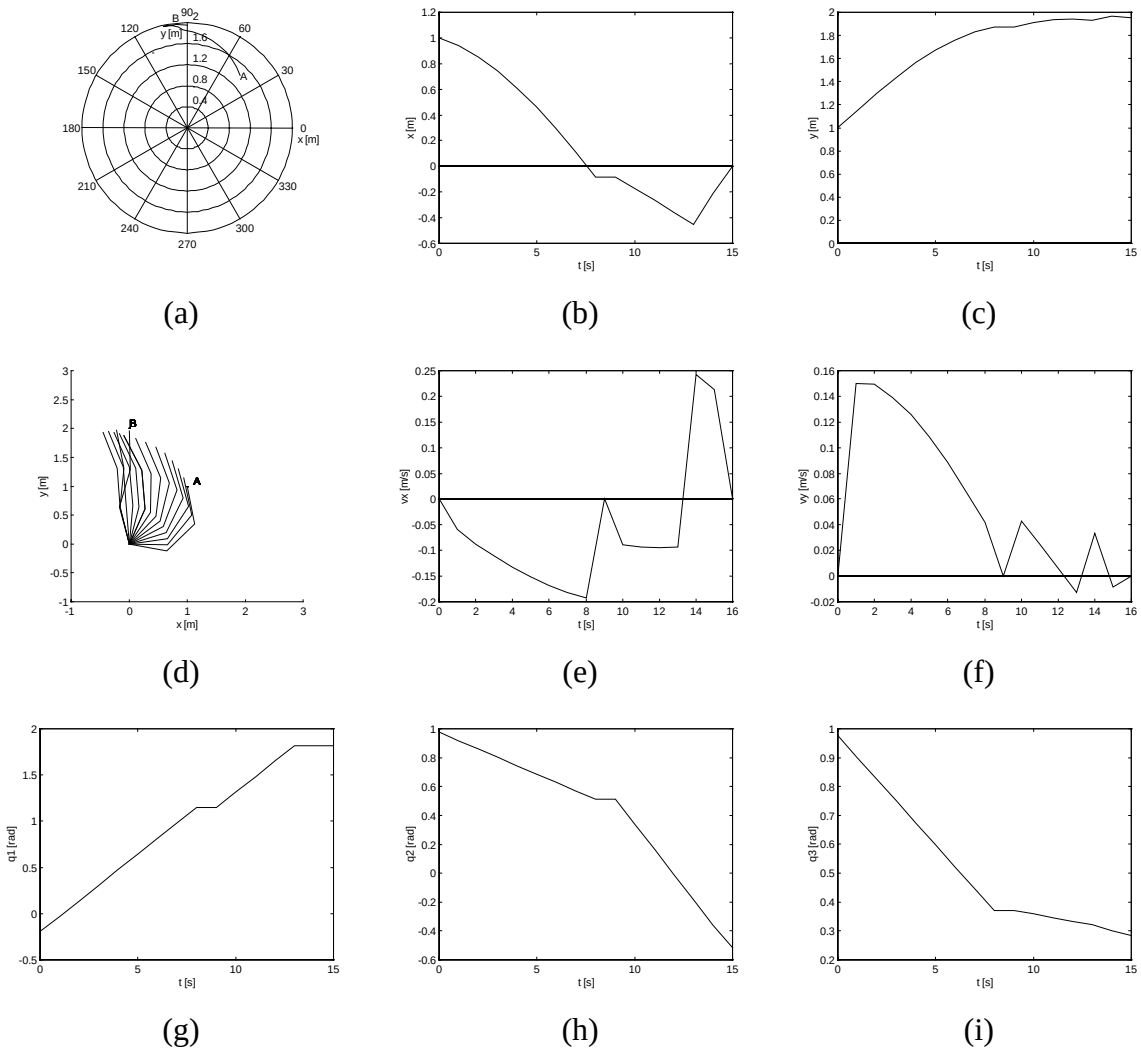
(f) Velocidade do órgão terminal – eixo Y

(g) Deslocamento da junta 1

(h) Deslocamento da junta 2

(i) Velocidade da junta 1

(j) Velocidade da junta 2



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 4.29 Posições e velocidades para o robô de 3 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal no espaço operacional

(b) Deslocamento do órgão terminal – eixo X

(c) Deslocamento do órgão terminal – eixo Y

(d) As sucessivas posições do robô entre o ponto inicial A e o ponto final B

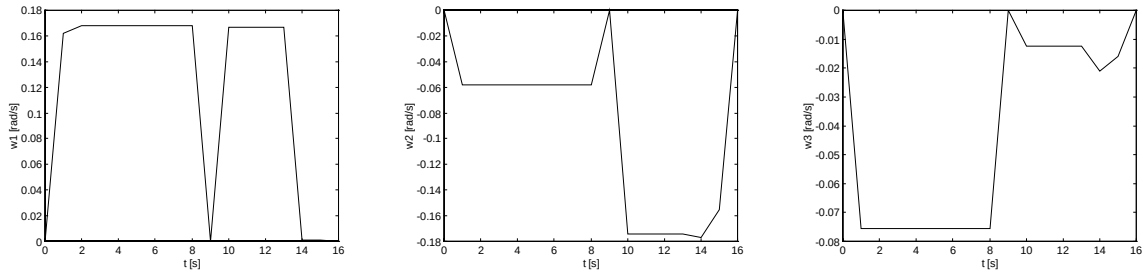
(e) Velocidade do órgão terminal – eixo X

(f) Velocidade do órgão terminal – eixo Y

(g) Deslocamento da junta 1

(h) Deslocamento da junta 2

(i) Deslocamento da junta 3



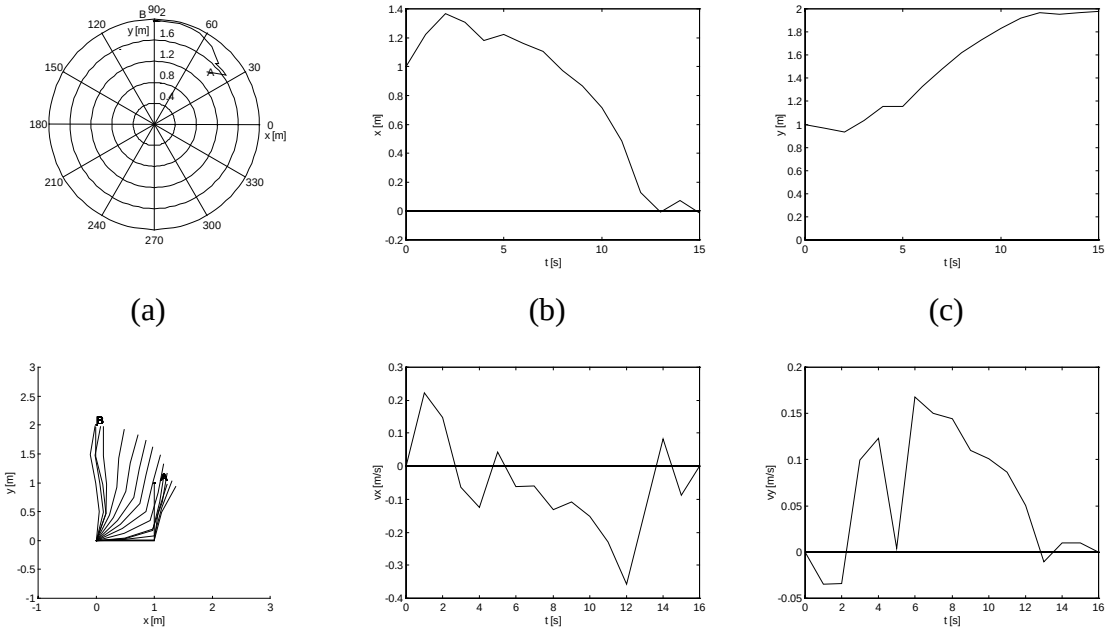
(j) (k) (l)

Figura 4.29 Posições e velocidades para o robô de 3 elos (p_c = 0,8; p_m = 0,03).

(j) Velocidade da junta 1

(k) Velocidade da junta 2

(l) Velocidade da junta 3



(a) (b) (c)

(d) (e) (f)

Figura 4.30 Posições e velocidades para o robô de 4 elos (p_c = 0,8; p_m = 0,03).

(a) Trajectória do órgão terminal no espaço operacional

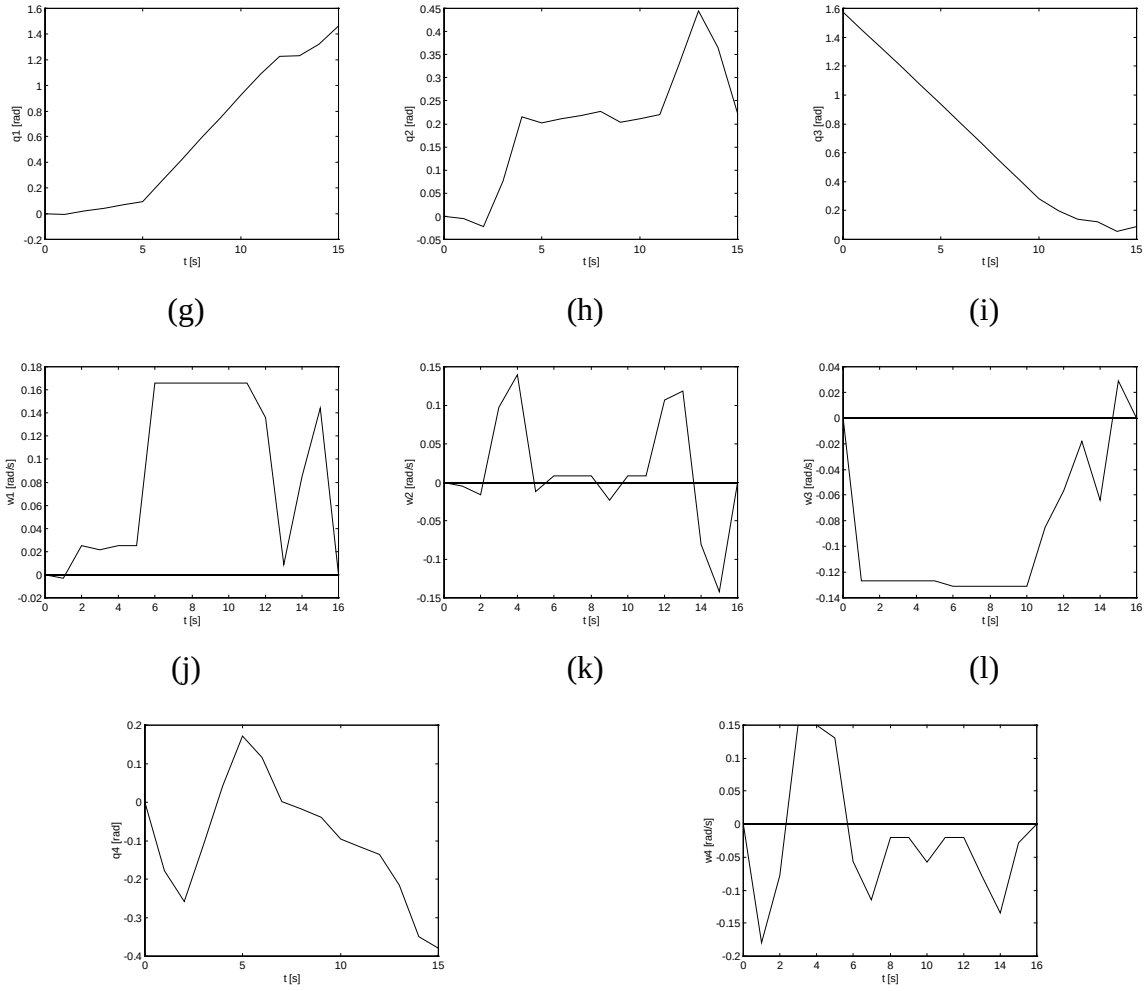
(b) Deslocamento do órgão terminal – eixo X

(c) Deslocamento do órgão terminal – eixo Y

(d) As sucessivas posições do robô entre o ponto inicial A e o ponto final B

(e) Velocidade do órgão terminal – eixo X

(f) Velocidade do órgão terminal – eixo Y



(g) (h) (i)

(j) (k) (l)

(m) (n)

Figura 4.30 Posições e velocidades para o robô de 4 elos (continuação).

(g) Deslocamento da junta 1

(h) Deslocamento da junta 2

(i) Deslocamento da junta 3

(j) Velocidade da junta 1

(k) Velocidade da junta 2

(l) Velocidade da junta 3

(m) Deslocamento da Junta 4

(n) Velocidade da junta 4

Comparando as figuras dos dois conjuntos de experiências conclui-se que as últimas tem uma ondulação residual significativamente menor. Com o fim de se obter um comportamento das velocidades mais estável pode modificar-se os pesos na função de aptidão (*i.e.* os parâmetros α s). De modo análogo pode fazer-se um estudo para as acelerações do robô.

4.10 Análise do tempo requerido pelo robô para planear as trajectórias

4.10.1 Introdução

Nesta secção é feito o estudo do tempo necessário para o robô realizar o planeamento das trajectórias.

4.10.2 Tempos de simulação

Foram realizados quatro conjuntos de teste para estimar o tempo que o algoritmo demora a determinar uma trajectória apropriada. Assim estudaram-se robôs de 2, 3 e 4 elos e espaços operacionais com 0, 1 e 2 obstáculos. Os resultados encontram-se na tabela 4.6 para 500 gerações. Os parâmetros do algoritmo são: $p_c = 0,6$, $p_m = 0,05$, o comprimento do cromossoma é de $l = 15$ e o número de cromossomas da população é de $\mu = 100$

Tabela 4.6 Tempo médio (segundos) de 4 simulação com 500 gerações.

Tempo médio das simulações [s] [1]	Ambiente sem obstáculos	Ambiente com 1 obstáculo	Ambiente com 2 obstáculos
Robô com 2 elos	9	244	277,25
Robô com 3 elos	12,25	246,25	285
Robô com 4 elos	14,75	251,5	290

[1] As simulações foram executadas num Pentium a 166 Hz sendo utilizado o sistema operativo MSDOS.

Como se pode ver pela tabela o algoritmo é bastante rápido para ambientes sem obstáculos. Em ambientes com obstáculo o algoritmo é significativamente mais lento sendo o incremento de tempo insignificante quando se aumenta de 1 obstáculo para dois obstáculos. O aumento de tempo gasto pelo algoritmo quando passa a considerar um obstáculo é devido, sobretudo, à discretização dos elos do robô em pontos com vista analisar o conflito desses pontos da estrutura com o obstáculo.

5 Conclusões e perspectivas para desenvolvimentos futuros

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 Conclusões

Nesta dissertação foram apresentados os aspectos principais sobre os algoritmos evolutivos, especialmente sobre algoritmos genéticos, estratégias de evolução, programação evolutiva e programação genética.

Foi também realizado um resumo de aplicações de sistemas robóticos que utilizam algoritmos evolutivos, com particular incidência nos AGs. Estas aplicações incidem na geração de trajectórias para robôs móveis, na selecção do robô que melhor se adapte a uma determinada tarefa, na locomoção de robôs, no planeamento de trajectórias para manipuladores robóticos e na estimação de parâmetros de robôs.

De seguida foram apresentados os resultados do planeamento de trajectórias para manipuladores robóticos de 2, 3 e 4 elos, sendo, para esse fim, desenvolvido um programa baseado em AGs. Nessas experiências foram variados diversos parâmetros dos AGs, nomeadamente as probabilidades de cruzamento e de mutação, os pesos da função de aptidão e o comprimento do cromossoma. A partir dos resultados foi verificado que todos os parâmetros têm um papel importante na qualidade exibida pela solução final. Por últi-

mo, na função de aptidão, foi ainda introduzido um coeficiente de modo a reduzir a ondulação residual da cinemática diferencial.

Os resultados obtidos demonstram, claramente, que o planeamento de trajectórias através de AGs é viável. Além disso, a complexidade do algoritmo é mantida, independentemente da existência, ou não, de obstáculos, e do número destes. 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 profunda.

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:

- Planeamento de trajectórias tendo em conta a dinâmica robô.

- Planeamento de trajectórias no espaço das juntas (em alternativa ao espaço operacional) utilizando a cinemática inversa e tendo em atenção as singularidades.

- Estudo de novas funções de aptidão nomeadamente a manipulabilidade do manipulador robótico.

- Comparação das estruturas mecânicas PP, RR e RP para um determinado problema. Para cada estrutura determinação dos parâmetros óptimos segundo um determinado objectivo.

- Aplicação dos AGs em sistema com vários braços cooperantes, robôs redundantes e sistemas de locomoção.

- Modificação da função de aptidão de forma a diminuir o tempo de cálculo com vista à sua aplicação em sistemas de tempo real.

- Melhoria ou modificação do operador de cruzamento de modo a aumentar o seu desempenho no algoritmo proposto.

- Aplicação de outros tipos de AEs no planeamento de trajectórias para manipuladores robóticos de forma a encontrar o que melhor se adapte a este tipo de problema.

Em conclusão, pode afirmar-se, sem quaisquer dúvidas, que existe um largo espectro de investigação que, no futuro poderá trazer novos formalismos, permitir aplicações mais sofisticadas com desempenhos ainda mais relevantes.

Índice remissivo

algoritmos paralelos
  difusos, 11, 13
  globais, 11
  migração, 11, 12
casamento por padrão, 40
comprimento de um *arranjo*, 47
concatenação, 54
convergência prematura, 14
corte, 54
crowding, 57
cruzamento
  aritmético não uniforme, 60
  aritmético uniforme, 60
  discreto, 67
  intermédio, 67
cut, 54
estratégia elitista, 41
fenótipo, 10
função de aptidão
  escalonamento linear, 23
  escalonamento por potência, 23
  truncação sigma, 24
genótipo, 10
inbreeding with intermittent crossbreeding, 40
linebreeding, 40
mating templates, 40
método das penalidades, 22
métodos de selecção
  Amostragem estocástica, 30
  amostragem estocástica universal, 30
  Bolzmann, 29
  crowding, 30
  dinâmicos, 26
  elitista, 26, 29
  elitista do valor esperado, 30
  estáticos, 26
  extintivo direita, 26
  extintivo esquerda, 26
  extintivos, 26
  factor de agrupamento, 30
  genarational, 26
  geracional, 26
  nível, 27
  posto, 27
  preservativos, 26
  pura, 26
  remainder stochastic sampling with replacement, 30
  stochastic universal sampling, 30
  substituição imediata, 26
  torneio, 28
  valor esperado, 29
migração entre vizinhos, 12
migração livre, 12
migração sem restrições, 12
modelo casamento espacial, 58
modelo das ilhas, 58
niche preemption, 14
operador de inversão
  cycle crossover (CX), 37
  order crossover (OX), 36
  partially matched crossover (PMX), 36
ordem de um *arranjo*, 46
preempção de nicho, 14
reprodução de linhagem, 40
reprodução interna com cruzamento intermitente, 40
selecção
  amostragem estocástica universal, 25
  bias, 25
  direcção, 25
  diversidade da população, 25
  efeciência, 25
  extensão, 25
  pressão, 25
  roleta redonda, 25
  spread, 25
  stochastic universal sampling, 25
sharing, 57
splice, 54
tempo de vida bilinear, 52
tempo de vida linear, 52
tempo de vida proporcional, 52