

Cecília Maria do Rio Fernandes Moreira Reis

Síntese de Sistemas Digitais por Computação Evolutiva

Tese submetida à Universidade de Trás-os-Montes e Alto Douro
para a obtenção do grau de Doutor em Ciências da Engenharia
de acordo com o Dec.-Lei 216/92 de 13 de Outubro.



UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
VILA REAL, 2007

Tese realizada sob a orientação do
Professor Doutor José António Tenreiro Machado,
do Departamento de Engenharia Electrotécnica do
Instituto Superior de Engenharia do Instituto Politécnico do Porto

e co-orientação do
Professor Doutor José Boaventura Ribeiro da Cunha,
do Departamento de Engenharias da
Universidade de Trás-os-Montes e Alto Douro

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



Aos meus pais, José e Helena

e

aos meus filhos, Tiago e Catarina

Agradecimentos

Ao Professor José António Tenreiro Machado, pelo incentivo, colaboração, disponibilidade e excelente orientação, sem os quais a realização desta tese de Doutoramento nunca teria sido possível.

Ao Professor José Boaventura Cunha pela disponibilidade, colaboração e estímulo.

Ao Instituto Superior de Engenharia do Porto por todas as facilidades concedidas para a realização deste trabalho e pelos subsídios atribuídos para suportar as despesas de deslocação a conferências.

À Universidade de Trás-os-Montes e Alto Douro pela disponibilização de verbas que permitiram a aquisição de material e a participação em conferências.

Ao programa PRODEP pela concessão de uma bolsa que permitiu a dispensa de serviço lectivo para a realização deste trabalho.

À Fundação Luso-Americana para o Desenvolvimento pela atribuição de uma bolsa para participação numa conferência.

Aos colegas do Departamento e em particular aos do grupo de investigação GRIS pelo incentivo, espírito de colaboração e disponibilidade sempre demonstrada.

À Isabel, companheira de tese, pela sua total disponibilidade.

A dois amigos de longa data, Ana e Lino, com os quais sempre pude contar.

Ao Viriato pelo incentivo, dedicação e compreensão.

Agradecimentos

Aos meus filhos, Tiago e Catarina que me deram ânimo para levar esta tese de Doutorado até ao fim.

Aos meus pais, José e Helena por estarem sempre em sintonia comigo.

Resumo

A crescente complexidade dos circuitos digitais sugere, cada vez mais, o recurso a métodos de síntese automática. Por outro lado, nas últimas décadas têm sido propostos algoritmos inspirados na natureza e em processos biológicos, dando origem à área da inteligência computacional na qual se enquadra a computação evolutiva.

Nesta tese foram revistos vários tópicos da computação evolutiva, nomeadamente os algoritmos evolutivos e os algoritmos da inteligência dos enxames.

A tese contribui para a área da síntese automática de circuitos lógicos combinatórios com três ferramentas baseadas, respectivamente, em algoritmos genéticos, algoritmos meméticos e algoritmos de optimização por enxame de partículas.

Com o objectivo de melhorar o desempenho dos algoritmos implementados foram propostas duas técnicas baseadas na forma de avaliação da função de aptidão. Na primeira, aplicada à função de aptidão clássica, designada por estática, incluiu-se a medida da descontinuidade do erro. Na segunda, designada por dinâmica, propõe-se um novo conceito de função de aptidão: a função de aptidão dinâmica foi implementada através de dois métodos de aplicação de cálculo fraccionário, o primeiro usando séries de Taylor e o segundo usando a aproximação de fracções de Padé.

Ambas as abordagens permitiram melhores resultados face aos obtidos com a função de aptidão estática, nomeadamente no que diz respeito ao número de

gerações necessárias para a obtenção de uma solução. Foi também estudada a aplicação das duas técnicas em conjunto, isto é, função de aptidão dinâmica e inclusão da medida de descontinuidade do erro na função de aptidão. Com esta combinação obtiveram-se resultados superiores aos conseguidos quando se aplicam separadamente.

O algoritmo memético implementado tem por base um algoritmos genético no qual é inserido um algoritmo de pesquisa local o que permite, para cada potencial solução do problema, efectuar uma procura minuciosa no espaço de pesquisa vizinho dessa solução. Esta técnica reduz o número de gerações necessárias para obtenção de uma solução bem como o respectivo desvio padrão. É, por este motivo, uma boa alternativa dentro dos algoritmos evolutivos.

O algoritmo de optimização por enxame de partículas demonstrou ser uma técnica de computação evolutiva bem adaptada ao desenho de circuitos digitais dado o seu muito bom desempenho em termos de tempo de processamento, quer pelo reduzido número de gerações necessárias quer pela simplicidade dos cálculos, que permite uma rápida convergência mesmo quando o número de gerações é mais elevado.

Esta tese apresenta também uma metodologia de classificação, em termos de complexidade, de circuitos lógicos combinatórios.

Abstract

The growing complexity of digital circuits suggests, more and more, that automatic synthesis methods must be used. However, in the last decades, nature and biological inspired algorithms have been proposed, giving rise, in the computational intelligence field, to the area that includes evolutionary computation.

This thesis reviews some topics of evolutionary computation, namely the evolutionary algorithms and the swarm intelligence algorithms.

The thesis contributes to the automatic synthesis of combinational logic circuits area with three tools based on, respectively, genetic algorithms, memetic algorithms and particle swarm optimization.

In order to improve the behaviour of the implemented algorithms, two techniques based on the fitness function evaluation have been proposed. The former, hereby called static function, resides on the classic fitness function and includes a measure of the error discontinuity. The second, hereby called dynamical function, proposes a new concept of fitness: the dynamical fitness function has been implemented by two methods, both residing on fractional calculus applications, the former using Taylor's series and the second one using fraction approximations according to Padé.

Both approaches allowed better results when compared with the ones obtained with the static fitness function, especially in what concerns to the number of generations to achieve a solution. It was also studied the application of the two techniques together, that is, the dynamic fitness function and the

discontinuity error measure in the fitness function. This combination gives rise to superior results when compared to the ones obtained by application of the two techniques in separate.

The implemented memetic algorithm is based on a genetic algorithm plus a local search algorithm that allows, for each potential solution, a local refinement of the neighbour search space of that particular solution. This technique reduces the necessary number of generations to obtain the solution as well as the respective standard deviation. For this motive it is a good alternative within the evolutionary algorithms.

The particle swarm optimization algorithm demonstrated to be a well-adapted evolutionary computation technique in the digital circuits design area due to its good performance in terms of processing time, to the reduced number of necessary generations and to the calculus simplicity, that allows a fast convergence even when the number of generations is higher.

This thesis also shows a classification methodology, in terms of complexity, of combinational logic circuits.

Índice

AGRADECIMENTOS	i
RESUMO	iii
ABSTRACT	v
ÍNDICE	vii
ÍNDICE DE FIGURAS	xi
ÍNDICE DE TABELAS	xviii

Capítulo 1

INTRODUÇÃO	1
INTRODUÇÃO	1
1. Sistemas Digitais e Computação Evolutiva.....	2
2. Motivação e objectivos	3
3. Contribuições científicas	4
4. Estrutura da tese	5
REFERÊNCIAS	7

Capítulo 2

COMPUTAÇÃO EVOLUTIVA	11
INTRODUÇÃO	11
1. Algoritmos Evolutivos	12
1.1. Introdução.....	12
1.2. Estrutura de um AE.....	14
1.2.1. Codificação das soluções	15
1.2.2. Selecção e operadores de pesquisa	15
1.2.2.1. Função de aptidão	16
1.2.2.2. Operador de cruzamento	16
1.2.2.3. Operador de mutação	17
1.2.2.4. Selecção.....	17
1.2.3. Componentes de um AE.....	18
1.3. AE básico	18
1.3.1. Condição de conclusão	19

1.3.2. Resultado de um AE	19
1.4. Aplicações dos AEs.....	20
1.4.1. Aplicações gerais	21
2. Inteligência dos Enxames	23
3. Algoritmos Genéticos.....	27
3.1. Evolução e selecção natural.....	27
3.2. Princípios básicos dos AGs.....	32
3.3. Características Gerais dos AGs	36
3.4. Fundamentos Matemáticos dos AG's	41
3.4.1. Análise do operador de selecção no teorema de padrões	42
3.4.2. Análise do operador de cruzamento no teorema de padrões.....	44
3.4.3. Análise do operador de mutação no teorema de padrões.....	45
3.5. Funcionamento dos AG's	46
3.5.1. Processo de Selecção	46
3.5.2. Operadores Genéticos.....	47
3.5.3. Parâmetros Genéticos.....	49
3.5.4. Tamanho da população	50
3.5.4.1. Fornecimento de BCs	50
3.5.4.2. Tomada de decisão de BCs.....	52
3.5.4.3. Modelo da Ruína do Jogador.....	53
4. Algoritmos Meméticos	56
4.1. Introdução.....	56
4.2. Características dos AMs.....	58
4.3. Pesquisa Local	62
4.3.1. Espaço de pesquisa e vizinhanças.....	63
5. Optimização por Enxame de Partículas.....	65
5.1. Introdução.....	65
5.2. Comportamento social	66
5.3. A Etiologia da Optimização por Enxame de Partículas.....	68
5.3.1. Pesquisa Multidimensional.....	72
5.3.2. Versão simplificada actual	73
5.4. Enxames e Partículas	73
5.5. Algoritmo básico.....	74
6. Resumo do capítulo	78
REFERÊNCIAS	79

Capítulo 3

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS	85
INTRODUÇÃO	85
1. Electrónica evolutiva	86
2. Formulação do problema.....	88
2.1. Codificação dos circuitos	90
2.2. Operadores genéticos.....	91
2.3. Função de aptidão	92
3. Implementação dos circuitos.....	93
4. Problema de escala	103
5. Tamanho da População e Tempo de Processamento.....	108
5.1. Tamanho da população.....	109
5.2. Experiências desenvolvidas.....	110
6. Resumo do capítulo.....	118
REFERÊNCIAS	119

Capítulo 4

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS E CÁLCULO FRACCIONÁRIO... 123

INTRODUÇÃO	123
1. Cálculo Fraccionário	124
2. Funções de aptidão estática e dinâmica	125
3. Experiências desenvolvidas	128
3.1. Utilizando a função de aptidão estática F_e	129
3.2. Utilizando a função de aptidão dinâmica F_d	131
3.3. Planos de fase	142
3.3.1. Com a função de aptidão estática (F_e).....	143
3.3.2. Com a função de aptidão dinâmica (F_d)	148
3.4. Outros circuitos.....	152
3.5. Calculando F_d usando uma aproximação de Padé.....	160
4. Resumo do capítulo	163
REFERÊNCIAS	165

Capítulo 5

SÍNTESE DE CIRCUITOS COM ALGORITMOS MEMÉTICOS 167

INTRODUÇÃO	167
1. O Algoritmo Memético	168
1.1. Pesquisa local	170
2. Experiências desenvolvidas	172
2.1. Função de aptidão sem medida da descontinuidade do erro	174
2.2. Função de aptidão com medida da descontinuidade do erro	180
3. Análise da convergência	182
4. AM versus AG.....	190
4.1. Convergência.....	193
4.2. Escala	195
5. Resumo do capítulo.....	197
REFERÊNCIAS	198

Capítulo 6

SÍNTESE DE CIRCUITOS COM OPTIMIZAÇÃO POR ENXAMES DE PARTÍCULAS..... 201

INTRODUÇÃO	201
1. Optimização por Enxames de Partículas	202
1.1. Parâmetros.....	202
1.2. Topologias	204
1.3. Algoritmo.....	205
2. Algoritmo OEP para síntese de circuitos.....	206
2.1. Codificação dos circuitos e parâmetros OEP.....	206
2.2. Funções de aptidão	208
3. Experiências desenvolvidas	210

3.1. Com a função de aptidão estática F_e	210
3.2. Com a função de aptidão dinâmica F_d	216
3.3. Função de aptidão estática versus dinâmica.....	217
3.4. Comparação dos algoritmos genético, memético e OEP.....	219
3.4.1. Índices $\mu(N)$, $\sigma(N)$ e $\mu(TPS)$	219
3.4.2. Tamanho da população.....	225
3.4.3. Complexidade dos circuitos.....	228
3.4.4. Problema de escala.....	230
3.4.5. Função de aptidão sequencial.....	232
3.4.5.1. Problema de Escala.....	235
4. Resumo do capítulo.....	239
REFERÊNCIAS.....	241

Capítulo 7

CONCLUSÕES.....	243
INTRODUÇÃO.....	243
1. Trabalho realizado.....	243
2. Síntese conclusiva.....	245
3. Desenvolvimentos futuros.....	247

Apêndice A

SISTEMAS DIGITAIS.....	249
INTRODUÇÃO.....	249
1. Circuitos Digitais.....	250
1.1. Estrutura dos sistemas digitais.....	252
2. Álgebra de Boole.....	256
3. Portas Lógicas.....	258
3.1. Portas lógicas AND, OR, XOR e NOT.....	258
4. Circuitos Combinatórios.....	261
4.1. Circuitos de Comunicação.....	263
4.1.1. Codificadores.....	263
4.1.2. Descodificadores.....	264
4.1.3. Multiplexadores.....	265
4.1.4. Demultiplexador.....	267
4.2. Circuitos aritméticos.....	268
4.2.1. Semi-somadores.....	269
4.2.2. Somadores.....	270
4.2.3. Subtractores.....	271
4.2.4. Comparadores.....	273
REFERÊNCIAS.....	274

Índice de Figuras

Capítulo 2

COMPUTAÇÃO EVOLUTIVA

Figura 2. 1 - Algoritmo evolutivo básico.....	19
Figura 2. 2 - Colônia de formigas	23
Figura 2. 3 - Bando de pássaros	24
Figura 2. 4 - Manada de animais.....	24
Figura 2. 5 - Cardume de peixes	24
Figura 2. 6 - Comportamento das formigas após o aparecimento de um obstáculo entre o ninho e uma fonte de alimento.	25
Figura 2. 7 - Capa da primeira publicação do livro “The origin of Species” de Charles Darwin	29
Figura 2. 8 - O processo da selecção natural	30
Figura 2. 9 - Método da Roleta	37
Figura 2. 10 - Exemplo de cruzamento num ponto.....	49
Figura 2. 11 - Algoritmo memético.....	59
Figura 2. 12 - Operadores de recombinação e mutação agindo como estratégias de diversificação nos algoritmos meméticos	60
Figura 2. 13 - Algoritmo genérico representativo de uma pesquisa local	62
Figura 2. 14 - Voo típico em forma de V de um bando de pássaros	67
Figura 2. 15 - Algoritmo PSO.....	76

Capítulo 3

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS

Figura 3. 1- Matriz 3 x 3 que representa um circuito.	90
Figura 3. 2 - Cromossoma da matriz da figura 3.1.....	91
Figura 3. 3 - Função de aptidão F versus número de gerações N para obter a solução para o circuito M21, utilizando o AG.....	94
Figura 3. 4 - Circuito Multiplexador 2 para 1 gerado pelo AG.....	95
Figura 3. 5 - Função de aptidão F versus número de gerações N para obter a solução para o circuito SOM1, utilizando o AG.....	96
Figura 3. 6 - Circuito Somador de um <i>bit</i> gerado pelo AG.....	96
Figura 3. 7 - Função de aptidão F versus número de gerações N para obter a solução para o circuito TP4, utilizando o AG.....	97
Figura 3. 8 - Circuito Teste de Paridade de quatro <i>bits</i> gerado pelo AG.....	98
Figura 3. 9 - Função de aptidão F versus número de gerações N para obter a solução para o MUL2, utilizando o AG	99
Figura 3. 10 - Circuito Multiplicador de dois <i>bits</i> gerado pelo AG.....	99
Figura 3. 11 - Média do número de gerações para alcançar a solução $\mu(N)$ para os conjuntos de portas lógicas em avaliação, utilizando o AG.....	101
Figura 3. 12 - Média da função de aptidão dos circuitos obtidos $\mu(F)$ para os conjuntos de portas lógicas em avaliação, utilizando o AG.....	101
Figura 3. 13 - Média da função de aptidão $\mu(F)$ versus a média do número de gerações para obter a solução $\mu(N)$, utilizando o AG.	102
Figura 3. 14 - Média do número de gerações para obter a solução $\mu(N)$ para os circuitos de teste de paridade de 2, 3, 4, 5 e 6 <i>bits</i> para os conjuntos de portas lógicas em avaliação. 103	

Figura 3. 15 - Desvio padrão do número de gerações para obter a solução $\mu(N)$ para os circuitos de teste de paridade de 2, 3, 4, 5 e 6 bits para os conjuntos de portas lógicas em avaliação.	104
Figura 3. 16 - Média da função de aptidão final $\mu(F)$ para os circuitos de teste de paridade de 2, 3, 4, 5 e 6 bits para os conjuntos de portas lógicas em avaliação.	104
Figura 3. 17 - Média do número de gerações para obter a solução $\mu(N)$ para os circuitos somadores completos de 1 e 2 bits para os conjuntos de portas lógicas em avaliação.	105
Figura 3. 18 - Desvio padrão do número de gerações para obter a solução $\mu(N)$ para os circuitos somadores completos de 1 e 2 bits para os conjuntos de portas lógicas em avaliação.	105
Figura 3. 19 - Média da função de aptidão final $\mu(F)$ para os circuitos somadores completos de 1 e 2 bits para os conjuntos de portas lógicas em avaliação.	106
Figura 3. 20 - $\mu(F)$ versus $\mu(N)$ para as famílias dos circuitos de teste de paridade e somador completo, para os conjuntos de portas lógicas 2, 3, 4 e 6.	107
Figura 3. 21 - TPS versus (P, TM) com $TC = 95\%$ usando desde o $Gset 1a$ até ao $Gset 6$ para o circuito M21.	114
Figura 3. 22 - TPS para alcançar a solução versus TM , com $TC = 95\%$ e $P = 70$, para o circuito M21.	116
Figura 3. 23 - TP_1 versus P com $TC = 95\%$ e $TM=5\%$ para o circuito M21.	117
Figura 3. 24 - TP_1 versus TM com $CR = 95\%$ e $P = 70$ para o circuito M21.	117

Capítulo 4

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS E CÁLCULO FRACIONÁRIO

Figura 4. 1- Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $\delta = \{0, 0.25, 0.5, 0.75, 1\}$ com os conjuntos $Gset 2$ e $Gset 4$, para o circuito M21.	130
Figura 4. 2 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $\delta = \{0, 0.25, 0.5, 0.75, 1\}$ com os conjuntos $Gset 2$ e $Gset 4$, para o circuito TP4.	130
Figura 4. 3 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $\delta = \{0, 0.25, 0.5, 0.75, 1\}$ com os conjuntos $Gset 2$ e $Gset 4$, para o circuito SOM1.	131
Figura 4. 4 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito M21.	132
Figura 4. 5 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito M21.	132
Figura 4. 6 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 4$, para o circuito M21.	133
Figura 4. 7 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto $Gset 4$, para o circuito M21.	133
Figura 4. 8 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito TP4.	133
Figura 4. 9 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito TP4.	134
Figura 4. 10 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 4$, para o circuito TP4.	134
Figura 4. 11 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto $Gset 4$, para o circuito TP4.	134
Figura 4. 12 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito SOM1.	135
Figura 4. 13 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito SOM1.	135

Figura 4. 14 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta=0$) com o conjunto <i>Gset 4</i> , para o circuito SOM1.....	135
Figura 4. 15 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta=0$) com o conjunto <i>Gset 4</i> , para o circuito SOM1.....	136
Figura 4. 16 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^\lambda D^\mu$, para $K = K_D = K_I$ e ($\delta=0$) com o conjunto <i>Gset 2</i> , para o circuito M21. 138	
Figura 4. 17 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^\lambda D^\mu$, para $K = K_D = K_I$ e ($\delta=0$) com o conjunto <i>Gset 2</i> , para o circuito TP4.. 138	
Figura 4. 18 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^\lambda D^\mu$, para $K = K_D = K_I$ e ($\delta=0$) com o conjunto <i>Gset 2</i> , para o circuito SOM1.138	
Figura 4. 19 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^\lambda D^\mu$, para $K = K_D = K_I$ e ($\delta=0$) com o conjunto <i>Gset 4</i> , para o circuito M21. 139	
Figura 4. 20 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^\lambda D^\mu$, para $K = K_D = K_I$ e ($\delta=0$) com o conjunto <i>Gset 4</i> , para o circuito TP4.. 139	
Figura 4. 21 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^\lambda D^\mu$, para $K = K_D = K_I$ e ($\delta=0$) com o conjunto <i>Gset 4</i> , para o circuito SOM1.139	
Figura 4. 22 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto <i>Gset 2</i> , para o circuito M21.....	140
Figura 4. 23 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto <i>Gset 2</i> , para o circuito TP4.....	141
Figura 4. 24 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto <i>Gset 2</i> , para o circuito SOM1.....	141
Figura 4. 25 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto <i>Gset 4</i> , para o circuito M21.....	141
Figura 4. 26 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto <i>Gset 4</i> , para o circuito TP4.....	142
Figura 4. 27 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto <i>Gset 4</i> , para o circuito SOM1.....	142
Figura 4. 28 - Plano de fase de uma execução do AG, com os conjuntos <i>Gset 2</i> e <i>Gset 4</i> , usando F_e , para o circuito M21.....	144
Figura 4. 29 - Plano de fase de uma execução do AG, com os conjuntos <i>Gset 2</i> e <i>Gset 4</i> , usando F_e , para o circuito TP4.....	145
Figura 4. 30 - Plano de fase de uma execução do AG, com os conjuntos <i>Gset 2</i> e <i>Gset 4</i> , usando F_e , para o circuito SOM1.....	146
Figura 4. 31 - Plano de fase calculado com a derivada da média da função de aptidão de uma execução do AG, com o conjunto <i>Gset 4</i> , usando F_e , para o circuito SOM1.....	147
Figura 4. 32 - Plano de fase calculado com a derivada da melhor função de aptidão de uma execução do AG, com o conjunto <i>Gset 4</i> , usando F_e , para o circuito SOM1.....	148
Figura 4. 33 - Plano de fase de uma execução do AG, com os conjuntos <i>Gset 2</i> e <i>Gset 4</i> , usando F_d com o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para o circuito M21.....	149

Figura 4. 34 - Plano de fase de uma execução do AG, com os conjuntos <i>Gset 2</i> e <i>Gset 4</i> , usando F_d com o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para o circuito TP4.	150
Figura 4. 35 - Plano de fase de uma execução do AG, com os conjuntos <i>Gset 2</i> e <i>Gset 4</i> , usando F_d com o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para o circuito SOM1.	151
Figura 4. 36 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto <i>Gset 2</i> , para o circuito TP5.....	153
Figura 4. 37 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto <i>Gset 4</i> , para o circuito TP5.....	154
Figura 4. 38 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto <i>Gset 2</i> , para o circuito SUB1.....	155
Figura 4. 39 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto <i>Gset 4</i> , para o circuito SUB1.....	156
Figura 4. 40 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto <i>Gset 2</i> , para o circuito MUL2.	157
Figura 4. 41 – Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto <i>Gset 4</i> , para o circuito MUL2.	158
Figura 4. 42 - Circuito TP5 gerado pelo AG.	159
Figura 4. 43 - Circuito SUB1 gerado pelo AG.	159
Figura 4. 44 - Circuito MUL 2 gerado pelo AG.	159
Figura 4. 45 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto <i>Gset 2</i> , para o circuito M21.	161
Figura 4. 46 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto <i>Gset 4</i> , para o circuito M21.	161
Figura 4. 47 - Média do número de gerações para obter a solução $\mu(N)$ e desvio padrão $\sigma(N)$ com o conjunto <i>Gset 2</i> , para o circuito TP4.	162
Figura 4. 48 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ p do número de gerações para obter a solução com o conjunto <i>Gset 4</i> , para o circuito TP4.	162
Figura 4. 49 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto <i>Gset 2</i> , para o circuito SOM1.	162
Figura 4. 50 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto <i>Gset 4</i> , para o circuito SOM1.	163

Capítulo 5

SÍNTESE DE CIRCUITOS COM ALGORITMOS MEMÉTICOS

Figura 5. 1- Algoritmo memético.....	169
Figura 5. 2 - Algoritmo genético e algoritmo de pesquisa local	169
Figura 5. 3 - Algoritmo de pesquisa local do tipo de porta (APLTP).....	171
Figura 5. 4 - Algoritmo APLTP com início da pesquisa local na primeira célula do cromossoma para o circuito M21, com o conjunto de portas lógicas <i>Gset 2</i>	171
Figura 5. 5 - Algoritmo APLTP com início da pesquisa local na última célula do cromossoma para o circuito M21, com o conjunto de portas lógicas <i>Gset 2</i>	171
Figura 5. 6- Função de aptidão F ($\delta = 0$) versus o número de gerações N necessárias para obter a solução para o circuito M21, utilizando o AM.....	174
Figura 5. 7- Função de aptidão F ($\delta = 0$) versus número de gerações N necessária para obter a solução para o circuito SOM1, utilizando o AM.	175
Figura 5. 8- Função de aptidão F ($\delta = 0$) versus o número de gerações N para obter a solução para o circuito TP4, utilizando o AM.....	176
Figura 5. 9- Função de aptidão F ($\delta = 0$) versus o número de gerações N necessárias para obter a solução para o MUL2, utilizando o AM.	177

Figura 5. 10- Média do número de gerações para alcançar a solução $\mu(N)$ para os conjuntos de portas lógicas em avaliação, utilizando o AM.....	178
Figura 5. 11 - Média da função de aptidão dos circuitos obtidos $\mu(F)$ para os conjuntos de portas lógicas em avaliação, utilizando o AG.....	179
Figura 5. 12 – Média da função de aptidão $\mu(F)$ versus o número de gerações necessárias para obter a solução N , utilizando o AM.....	179
Figura 5. 13 – Desvio padrão $\sigma(N)$ versus média do número de gerações $\mu(N)$ para obter a solução, para os circuitos M21, TP4 e SOM1, com $P = 10$ e $\delta = 0,5$ (para os casos do AM com medida da descontinuidade do erro).	181
Figura 5. 14 – Desvio padrão $\sigma(N)$ versus média do número de gerações $\mu(N)$ para obter a solução, para o circuito MUL2, com $P = 1000$ e $\delta = 0,5$ (para os casos do AM com medida da descontinuidade do erro).....	181
Figura 5. 15 – Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito M21, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	183
Figura 5. 16 – Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito SOM1, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	183
Figura 5. 17 – Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito TP4, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	184
Figura 5. 18 – Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito MUL2, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	184
Figura 5. 19 – Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito M21, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	186
Figura 5. 20 – Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito SOM1, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	187
Figura 5. 21 – Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito TP4, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	187
Figura 5. 22 – Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito MUL2, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, utilizando o AM.....	188
Figura 5. 23 – Média da função de aptidão $\mu(F)$ versus o número de gerações necessárias para obter a solução $\mu(N)$, para $P = 3000$	193
Figura 5. 24 – Tempo de processamento TPS versus o desvio padrão do número de gerações para obter a solução $\sigma(N)$ e a média do número de gerações para obter a solução $\mu(N)$, para os algoritmos AM e AG, com o conjunto de portas lógicas $Gset 2$ e o circuito TP4.....	194
Figura 5. 25 – Projecção no plano horizontal de TPS versus o desvio padrão do número de gerações para obter a solução $\sigma(N)$ e a média do número de gerações para obter a solução $\mu(N)$, para os algoritmos AM e AG, com o conjunto de portas lógicas $Gset 2$ e o circuito TP4.	195
Figura 5. 26 - $\mu(F)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para os algoritmos AG e AM, com os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$, para $P = 3000$	196

Capítulo 6

SÍNTESE DE CIRCUITOS COM OPTIMIZAÇÃO POR ENXAMES DE PARTÍCULAS

Figura 6. 1- Processo de Optimização por Enxame de Partículas 205

Figura 6. 2- $\sigma(N)$ versus $\mu(N)$ com $P = 3000$ e F_e ($\delta = 0$) para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para o algoritmo OEP..... 211

Figura 6.3- $\mu(TPS)$ versus $\mu(N)$ com $P = 3000$ e F_e ($\delta = 0$) para os circuitos { M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para o algoritmo OEP..... 211

Figura 6. 4 - $\mu(N)$ para o circuito SOM1 versus $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}..... 213

Figura 6. 5 - $\mu(N)$ para o circuito SUB1 versus $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}..... 213

Figura 6.6 - $\sigma(N)$ para o circuito SOM1 versus $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}..... 214

Figura 6. 7 - $\sigma(N)$ para o circuito SUB1 versus $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}..... 214

Figura 6. 8 - $\mu(N)$ e $\sigma(N)$ versus $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2} para os circuitos SOM1 e SUB1..... 215

Figura 6.9 - $\sigma(N)$ versus $\mu(N)$ para o algoritmo OEP, $P = 3000$ e F_d 216

Figura 6.10 - $\mu(TPS)$ versus $\mu(N)$ para o algoritmo OEP, $P = 3000$ e F_d 217

Figura 6. 11 - $\mu(N)$ e $\sigma(N)$ para o algoritmo OEP e $P = 3000$, com F_e e F_d 218

Figura 6. 12- $\sigma(N)$ versus $\mu(N)$ com $P = 3000$ e F_e ($\delta = 0$) para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para os algoritmos AG e AM. 220

Figura 6. 13- $\mu(TPS)$ versus $\mu(N)$ com $P = 3000$ e F_e ($\delta = 0$) para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para os algoritmos AG e AM. 222

Figura 6. 14- Linhas de tendência para $\sigma(N)$ versus $\mu(N)$ e $\mu(TPS)$ versus $\mu(N)$ para os algoritmos AG, AM e OEP. 224

Figura 6. 15- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito M21 com $P = \{100, 500, 1000, 3000\}$ 226

Figura 6. 16- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito SOM1 com $P = \{100, 500, 1000, 3000\}$ e os algoritmos {AG, AM, OEP}..... 226

Figura 6. 17- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito SUB1 com $P = \{100, 500, 1000, 3000\}$ e os algoritmos {AG, AM, OEP}..... 227

Figura 6. 18- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito TP4 com $P = \{100, 500, 1000, 3000\}$ e os algoritmos {AG, AM, OEP}..... 227

Figura 6. 19- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o AG com $P = \{100, 500, 1000, 3000\}$ e para os circuitos {M21, SOM1, SUB1, TP4}. 229

Figura 6. 20- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o AM com $P = \{100, 500, 1000, 3000\}$ e para os circuitos {M21, SOM1, SUB1, TP4}. 229

Figura 6. 21- $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o OEP com $P = \{100, 500, 1000, 3000\}$ e para os circuitos {M21, SOM1, SUB1, TP4}. 230

Figura 6.22 - $\sigma(N)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para os algoritmos AG, AM e OEP e para os conjuntos de portas lógicas {2, 3, 4, 6}e $P = 3000$ 231

Figura 6.23 - $\mu(TPS)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para os algoritmos AG, AM e OEP e para os conjuntos de portas lógicas {2, 3, 4, 6}e $P = 3000$ 231

Figura 6. 24 - Desvio padrão $\sigma(N)$ versus a média do número de gerações $\mu(N)$ para obter a solução para o circuito SOM1, com os algoritmos {AG, AM, OEP} usando a função de aptidão com f_1 e com $f_1 + f_2$ 232

Figura 6. 25 - Desvio padrão $\sigma(N)$ versus a média do número de gerações $\mu(N)$ para obter a solução para o circuito SUB1, com os algoritmos {AG, AM, OEP} usando a função de aptidão com f_1 e com $f_1 + f_2$ 233

Figura 6. 26 - $\mu(N)$ para a família de circuitos de teste de paridade de $\{2, 3, 4, 5, 6\}$ bit com o AG, aplicando a função de aptidão com f_1 e com $f_1 + f_2$	236
Figura 6. 27 - $\mu(N)$ para a família de circuitos de teste de paridade de $\{2, 3, 4, 5, 6\}$ bit com o AM, aplicando a função de aptidão com f_1 e com $f_1 + f_2$	236
Figura 6. 28 - $\mu(N)$ para a família de circuitos de teste de paridade de $\{2, 3, 4, 5, 6\}$ bit com o OEP, aplicando a função de aptidão com f_1 e com $f_1 + f_2$	237
Figura 6. 29 - $\sigma(N)$ para a família de circuitos de teste de paridade de $\{2, 3, 4, 5, 6\}$ bit com o AG, aplicando a função de aptidão com f_1 e com $f_1 + f_2$	237
Figura 6. 30 - $\sigma(N)$ para a família de circuitos de teste de paridade de $\{2, 3, 4, 5, 6\}$ bit com o AM, aplicando a função de aptidão com f_1 e com $f_1 + f_2$	238
Figura 6. 31 - $\sigma(N)$ para a família de circuitos de teste de paridade de $\{2, 3, 4, 5, 6\}$ bit com o OEP, aplicando a função de aptidão com f_1 e com $f_1 + f_2$	238

Apêndice A

SISTEMAS DIGITAIS

Figura A. 1- ENIAC - 1946. Primeiro computador digital electrónico de grande escala	251
Figura A. 2 - Capa do livro "An Investigation of the Laws of Thought" de George Boole.....	257
Figura A. 3 - Representação genérica de um circuito lógico.....	258
Figura A. 4 - Tabelas de verdade: a) AND b) OR c) XOR d) NOT.....	259
Figura A. 5 - Tabelas de verdade: a) NAND b) NOR c) XNOR.....	260
Figura A. 6 - Diagrama genérico de um circuito combinacional.....	262
Figura A. 7 - Codificador binário 2^n -para- n	263
Figura A. 8 - Tabela de verdade do codificador binário 4-para-2.....	264
Figura A. 9 - Esquemático do circuito codificador binário 4-para-2.....	264
Figura A. 10 - Descodificador binário n -para- 2^n	265
Figura A. 11 - Tabela de verdade do descodificador binário 2-para-4.....	265
Figura A. 12 - Circuito multiplexador.....	266
Figura A. 13 - Tabela de verdade do multiplexador 4-para-1.....	267
Figura A. 14 - Esquemático do circuito multiplexador 4-para-1.....	267
Figura A. 15 - Circuito demultiplexador.....	268
Figura A. 16 - Tabela de verdade do demultiplexador 1-para-4.....	268
Figura A. 17 - Circuito semi-somador.....	269
Figura A. 18 - Tabela de verdade do semi-somador.....	269
Figura A. 19 - Esquemático do circuito semi-somador.....	270
Figura A. 20 - Circuito somador.....	270
Figura A. 21 - Tabela de verdade do somador completo.....	271
Figura A. 22 - Esquemático do circuito somador completo.....	271
Figura A. 23 - Circuito subtrator.....	272
Figura A. 24 - Tabela de verdade do subtrator.....	272
Figura A. 25 - Esquemático do circuito subtrator.....	272
Figura A. 26 - Esquema de um comparador de quatro bits.....	273

Índice de Tabelas

Capítulo 2

COMPUTAÇÃO EVOLUTIVA

Tabela 2. 1 - Terminologia natural/computacional	31
Tabela 2. 2 - Elementos de um AG	34

Capítulo 3

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS

Tabela 3. 1 - Conjuntos de Portas Lógicas: <i>Gsets</i> 2, 3, 4 e 6.....	89
Tabela 3. 2- Resultados para os circuitos M21, SOM1, TP4 e MUL, utilizando o AG	100
Tabela 3. 3 - Coeficientes <i>a</i> e <i>b</i> da equação 3.5	108
Tabela 3. 4 - Conjuntos de Portas Lógicas: <i>Gsets</i> 1a, 1b e 5	110
Tabela 3. 5 - Melhor <i>TPS</i> , utilizando o AG.....	115

Capítulo 4

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS E CÁLCULO FRACIONÁRIO

Tabela 4. 1- Parâmetros (μ , K_D) ou (λ , K_I) para cada melhor solução em termos de $\mu(N)$	136
Tabela 4. 2 - Parâmetros (μ , K_D) ou (λ , K_I) para cada melhor solução em termos de $\sigma(N)$	137

Capítulo 5

SÍNTESE DE CIRCUITOS COM ALGORITMOS MEMÉTICOS

Tabela 5. 1- Resultados para os circuitos M21, SOM1, TP4 e MUL, utilizando o AM	178
Tabela 5. 2 - Parâmetros $\mu(N) \approx \alpha P^\beta$ para “ <i>P</i> baixo”	185
Tabela 5. 3 - Parâmetros $\mu(N) \approx \alpha P^\beta$ para “ <i>P</i> elevado”	186
Tabela 5. 4 - Parâmetros $\sigma(N) \approx \chi P^\delta$ para “ <i>P</i> baixo”	189
Tabela 5. 5 - Parâmetros $\sigma(N) \approx \chi P^\delta$ para “ <i>P</i> elevado”	189
Tabela 5. 6 - Resultados para o circuito M21 com os algoritmos AG e AM	190
Tabela 5. 7 - Resultados para o circuito SOM1 com os algoritmos AG e AM	191
Tabela 5. 8 - Resultados para o circuito TP4 com os algoritmos AG e AM	192
Tabela 5. 9 - Resultados para o circuito MUL2 com os algoritmos AG e AM.....	192
Tabela 5. 10 - Coeficientes (α , β) da equação 5.8	197

Capítulo 6**SÍNTESE DE CIRCUITOS COM OPTIMIZAÇÃO POR ENXAMES DE PARTÍCULAS**

Tabela 6. 1- Parâmetros (a, b) e (c, d) das equações 6.13 e 6.14.....	223
Tabela 6. 2 - Razões $\mu(N)]_{f_1+f_2} / \mu(N)]_{f_1}$ e $\sigma(N)]_{f_1+f_2} / \sigma(N)]_{f_1}$ para as funções de aptidão sequenciais.	234

Apêndice A**SISTEMAS DIGITAIS**

Tabela A. 1- Portas Lógicas	260
-----------------------------------	-----

Índice

Capítulo 1

INTRODUÇÃO

Introdução

Este capítulo tem como objectivo apresentar, de forma clara e sucinta, o tema desta tese, a motivação, os seus objectivos e contribuições científicas.

O capítulo está estruturado da seguinte forma:

A secção 1 começa por fazer o enquadramento dos algoritmos da computação evolutiva, dos algoritmos evolutivos e dos algoritmos da inteligência dos enxames nas suas aplicações à síntese de sistemas digitais;

A secção 2 apresenta os motivos que estiveram na origem da escolha deste tema e delinea os objectivos a atingir com a realização deste trabalho;

A secção 3 enumera as contribuições científicas alcançadas;

Por último, a secção 4 apresenta a descrição da estrutura desta tese.

1. Sistemas Digitais e Computação Evolutiva

O tema desta tese - síntese de sistemas digitais por computação evolutiva - consiste no desenvolvimento de ferramentas de síntese automática de circuitos electrónicos digitais com algoritmos da computação evolutiva.

O termo electrónica digital designa os circuitos electrónicos que usam sinais digitais em substituição dos sinais analógicos. Os circuitos electrónicos digitais são a face visível mais comum da álgebra de Boole e estão na base de todos os circuitos que dão origem aos mais diversos equipamentos digitais tais como, por exemplo, computadores, sistemas de controlo, sistemas de som enfim, de praticamente todos, senão mesmo todos, os equipamentos electrónicos que nos rodeiam.

Nas últimas décadas têm sido propostos algoritmos inspirados na natureza e em processos biológicos, dando origem à área da inteligência computacional na qual se enquadra a computação evolutiva.

Os algoritmos evolutivos são algoritmos de optimização. A sua aplicação ao desenho e síntese de circuitos electrónicos conduziu ao aparecimento de uma nova área de investigação designada por Electrónica Evolutiva ou *Hardware Evolutivo*. A Electrónica Evolutiva dedica-se à síntese automática de sistemas electrónicos em substituição do recurso a modelos, abstrações e técnicas concebidas pelo ser humano, nomeadamente mapas de Karnaugh e métodos de simplificação algébricos.

A síntese automática de circuitos lógicos iniciou-se pela aplicação de algoritmos genéticos. Mais tarde foi também aplicada a este problema a programação genética.

Sendo a área da inteligência artificial uma área em constante evolução e havendo ainda muito a fazer na síntese automática de circuitos, acredita-se que será possível obter resultados cada vez mais satisfatórios com a integração destas duas áreas.

2. *Motivação e objectivos*

A crescente complexidade dos circuitos digitais sugere, cada vez mais, o recurso a métodos de síntese automática. A maioria dos algoritmos para optimização de circuitos digitais usa manipulações algébricas ou diagramas de decisão binária (*Binary Decision Diagrams* - BDDs) (Bryant, 1992, Drechsler, 1998), existindo experiências promissoras com algoritmos genéticos. Estes algoritmos pertencem à nova corrente de algoritmos da computação evolutiva.

A motivação para o desenvolvimento deste trabalho prende-se exactamente com a ideia de dar continuidade à exploração destas novas linhas de investigação, com vista ao desenvolvimento e aperfeiçoamento de ferramentas de síntese automática.

Assim, os principais objectivos deste trabalho são os seguintes:

- Concepção, desenvolvimento e testes de desempenho de uma aplicação com algoritmos genéticos para a síntese automática de circuitos digitais;
- Concepção, desenvolvimento e testes de desempenho de uma aplicação baseada num algoritmo híbrido para a síntese automática de circuitos digitais;

- Concepção, desenvolvimento e testes de desempenho de uma aplicação baseada num algoritmo da inteligência dos enxames para a síntese de circuitos digitais;
- Desenvolvimento de novos conceitos de avaliação da função de aptidão com o objectivo de melhorar o desempenho dos algoritmos implementados;
- Elaboração de um estudo comparativo dos resultados obtidos com as aplicações acima referidas.

3. Contribuições científicas

Uma tese de Doutoramento deve, naturalmente, proporcionar novas contribuições científicas que visam o enriquecimento da área em que se enquadra. Neste contexto, as principais contribuições científicas do presente trabalho enumeram-se, resumidamente, em seguida:

- Aplicação de novos algoritmos ao projecto e desenho de circuitos lógicos combinatórios, nomeadamente desenvolvimento de um algoritmo híbrido - o algoritmo memético - e o desenvolvimento de um algoritmo da inteligência dos enxames (Reis e Machado, 2003a, 2003b, Reis, Machado e Cunha, 2004b, 2005a, 2005b, 2005f, 2005h, 2006a, 2006b, 2006c);
- Análise experimental do tempo de processamento *versus* tamanho da população na aplicação de síntese de circuitos combinatórios com algoritmo genético (Reis, Machado e Cunha, 2004a);

- Desenvolvimento de novos conceitos de funções de aptidão (Reis, Machado e Cunha, 2005c, 2005d, 2005g, 2006d, 2006g);
- Aplicação do cálculo fraccionário no desenvolvimento de novos conceitos de funções de aptidão (Reis, Machado e Cunha, 2004c, 2004d, 2005e);
- Adaptação de algoritmos tipicamente aplicados em domínios contínuos, ao domínio discreto (Reis, Machado e Cunha, 2006e);
- Concepção de uma metodologia de classificação, em termos de complexidade, de circuitos lógicos combinatórios (Reis, Machado e Cunha, 2006f).

4. Estrutura da tese

Esta tese está organizada em sete capítulos e um apêndice, sendo o actual capítulo dedicado à introdução e o capítulo 2 dedicado ao estado da arte do tema aqui abordado. O trabalho desenvolvido é apresentado ao longo dos capítulos 3, 4, 5 e 6, finalizando-se com as conclusões no capítulo 7.

Descrevem-se, em seguida e em maior detalhe, cada um dos capítulos que compõem esta tese.

O capítulo 2 faz a apresentação dos algoritmos do ramo da computação evolutiva que servem de base às ferramentas desenvolvidas para a síntese de circuitos lógicos combinatórios.

O capítulo 3 descreve em pormenor o algoritmo genético implementado para o projecto de circuitos combinatórios em termos da codificação, operadores genéticos e função de aptidão, servindo de suporte aos capítulos seguintes. O capítulo inicia também o estudo do problema de escala na síntese de circuitos digitais e faz um estudo do tamanho da população a adoptar aquando da definição dos parâmetros do AG.

O capítulo 4 estuda e analisa novos conceitos na implementação da função de aptidão que avalia os circuitos gerados pelo AG. Neste seguimento, são propostos dois novos conceitos para a função de aptidão, nomeadamente: função de aptidão estática com medida da descontinuidade do erro; função de aptidão dinâmica com aplicação do cálculo fraccionário.

O capítulo 5 propõe um algoritmo memético para a síntese de circuitos lógicos combinatórios, descrevendo em pormenor a sua implementação, nomeadamente no que diz respeito ao algoritmo de pesquisa local.

O capítulo 6, dedicado à inteligência dos enxames, descreve em detalhe o algoritmo de OEP - Optimização por Enxame de Partículas - desenvolvido para a síntese de circuitos lógicos combinatórios.

O capítulo 7 termina a tese fazendo um breve resumo sobre o trabalho realizado e a respectiva síntese conclusiva, apresentando também possíveis direcções de desenvolvimento futuro.

Por último, o apêndice A serve de referência aos sistemas digitais, dado tratar-se da área de aplicação dos algoritmos desenvolvidos ao longo da presente tese.

Referências

- Bryant, R. E. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams, *ACM Computing Surveys*, Vol. 24, No. 3, pp. 293-318, September, 1992.
- Drechsler, R. e Becker, B. Binary Decision Diagrams: Theory and Implementation, Kluwer Academic Publishers. ISBN 0-7923-8193-9. 1998.
- Reis, C. e Tenreiro Machado, J. A. Synthesis of Combinational Logic Circuits using Genetic Algorithms, 8CLEEE - 8º Congresso Luso-Espanhol de Engenharia Electrotécnica, Vilamoura, Portugal, pp. 1.191-1.196, Julho, 2003a.
- Reis, C. e Tenreiro Machado, J. A. An Evolutionary Approach to the Synthesis of Combinational Circuits, *ICCC'2003 - IEEE International Conference on Computational Cybernetics*, Gold Cost, Lake Balaton, Siófok, Hungary, August, 2003b.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Population Size and Processing Time in a Genetic Algorithm, *ICCC'2004 - IEEE International Conference on Computational Cybernetics*, Vienna University of Technology, Vienna, Austria, August, 2004a.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Evolutionary Design of Combinational Logic Circuits, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Fuji Technology Press, Vol. 8, No. 5, pp. 507-513, September, 2004b.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Synthesis of Digital Circuits with Genetic Algorithms: A Fractional-Order Approach,

- International Journal of Computational Intelligence, Vol. 1, No. 4, pp. 289-294, December, 2004c.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Synthesis of Logic Circuits Using Fractional-Order Dynamic Fitness Functions, Proceedings of the ICCI'2004 - International Conference on Computational Intelligence, Istanbul, Turkey, pp. 77-79, December 2004d.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Logic Circuits Synthesis Through Genetic Algorithms, WSEAS Transactions on Information Science and Applications, Issue 5, Vol. 2, pp.618-623, May, 2005a.
- Cecília Reis, J. A. Tenreiro Machado, and J. Boaventura Cunha, Logic Circuits Synthesis Through Genetic Algorithms, 6th WSEAS International. Conference on Evolutionary Computation, Lisbon, Portugal, pp 257-262, June 2005b.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Fractional Dynamic Fitness Functions for GA-based Circuit Design, Proceedings of GECCO 2005 - Genetic and Evolutionary Computation Conference, Washington, DC, USA, pp. 1571-1572, June, 2005c.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Digital Circuit Design Using Dynamic Fitness Functions, LBP of GECCO 2005 - Genetic and Evolutionary Computation Conference, Washington, DC, USA, June, 2005d.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Evolutionary Design Of Combinational Circuits Using Fractional-Order Fitness Functions, ENOC-2005 - Fifth EUROMECH Nonlinear Dynamics Conference, Eindhoven, Netherlands, pp. 1312-1321, August 2005e.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. A Memetic Algorithm for Logic Circuit Design, 2005 WSEAS International.

- Conference on Dynamical Systems and Control, Venice, Italy, pp. 598-603, November, 2005f.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. New Concepts Towards the Synthesis of Digital Circuits Through Genetic Algorithms; Intelligent Systems at the Service of Mankind - Volume II, pp. 377-388, Ubooks, December, 2005g.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. An Evolutionary Hybrid Approach in the Design of Combinational Digital Circuits, WSEAS Transactions on Systems, Issue 12, Vol. 4, pp.2338-2345, December, 2005h.
- Reis, C., Machado, J. A. T., Figueiredo, L. e Boaventura Cunha, J. A Hybrid Algorithm for Logic Circuit Synthesis, Knowledge and Decision Technologies, pp 297-303. April, 2006a.
- Reis, C., Machado, J. A. T., Figueiredo, L. e Boaventura Cunha, J. A Hybrid Algorithm for Logic Circuit Synthesis, ICKEDS'06 - International Conference on Knowledge Engineering and Decision Support, Lisbon, Portugal, pp 297-303, May, 2006b.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Evolutionary Techniques in Circuit Design and Optimization", WSEAS Transactions on Power Systems, Issue 7, Vol. 1, pp.1337-1342, July, 2006c.
- Reis, C., Tenreiro Machado, J. A., Boaventura Cunha, J. e Figueiredo, L. Fractional-Order Evolutionary Design Of Digital Circuits, FDA'06 - 2nd IFAC Workshop on Fractional Differentiation and its Applications, Porto, Portugal, pp 445-450, July, 2006d.
- Reis, C., Tenreiro Machado, J. A., Galhano, A. e Boaventura Cunha, J. Circuit Synthesis Using Particle Swarm Optimization, IEEE - ICCS 2006, International Conference on Computational Cybernetics, Tallinn, Estonia, pp 149-154, August, 2006e.

Reis, C., Tenreiro Machado e Boaventura Cunha, J. Evolutionary Techniques in Circuit Design and Optimization, the 6th WSEAS International Conference on Simulation, Modelling and Optimization, Lisbon, Portugal, pp 307-312, September, 2006f.

Reis, C., Tenreiro Machado e Boaventura Cunha, J. Arithmetic Circuits Design Using Swarm Intelligence, WACI 2006 - Workshop on Applications of Computational Intelligence, Coimbra, Portugal, November, 2006g.

Capítulo 2

COMPUTAÇÃO EVOLUTIVA

Introdução

A Inteligência Artificial (IA) é o ramo das Ciências da Computação que trata da implementação e estudo de sistemas com inteligência autónoma ou comportamento próprio. A IA tenta imitar a inteligência humana por meio de computadores, tentando reproduzir faculdades humanas como a criatividade, o auto-aperfeiçoamento e o uso da linguagem. O grande objectivo final será a implementação de uma máquina que o utilizador não distinga do ser humano.

A investigação em IA diz respeito à implementação de máquinas para automatizar tarefas que requerem comportamentos inteligentes. Alguns exemplos são o controlo, planeamento e escalonamento, reconhecimento de voz e a robótica. Tornou-se assim uma área da engenharia empenhada na resolução de problemas do mundo real.

A IA divide-se em duas áreas de trabalho, a saber, a IA convencional e a Inteligência Computacional. A IA convencional trata dos métodos actualmente designados por aprendizagem automática (*machine learning*), sendo caracterizada pela análise formal e estatística, e onde se incluem os métodos tais

como, sistemas periciais (*expert systems*), raciocínio baseado em casos (*case base reasoning*) e redes bayesianas (*bayesian networks*). A Inteligência Computacional envolve os domínios do desenvolvimento interactivo e da aprendizagem destacando-se as redes neuronais, os sistemas difusos (*fuzzy systems*) e a computação evolutiva.

Por sua vez a Computação Evolutiva (CE) divide-se nos Algoritmos Evolutivos, onde se encontram os algoritmos genéticos e a programação genética, e na Inteligência dos Enxames (*Swarm Intelligence*), onde se destacam os algoritmos de optimização por colónias de formigas (*Ant Colony Optimization - ACO*) e os algoritmos de optimização por enxames de partículas (*Particle Swarm Optimization - PSO*).

Estes algoritmos são descritos nas secções que se seguem.

1. Algoritmos Evolutivos

1.1. Introdução

Os Algoritmos Evolutivos (AEs) são algoritmos de optimização que propõem um novo paradigma, inspirado na teoria da Selecção Natural (Darwin, 1859), para a resolução de problemas de optimizações em geral. Os AEs compreendem um conjunto de técnicas de pesquisa e optimização apoiadas na evolução natural das espécies. Desta forma, cria-se uma população de indivíduos que vão reproduzir e competir pela sobrevivência. Os melhores indivíduos sobrevivem e transferem as suas características às novas gerações. Nestes algoritmos incluem-se as técnicas (Banzhaf, 1998): Programação Evolutiva, Estratégias

Evolutivas, Algoritmos Genéticos e Programação Genética. Estes métodos são utilizados, cada vez mais, pela comunidade de inteligência artificial para obter modelos de inteligência computacional (Barreto, 1997).

Os Algoritmos Genéticos (AG) e a Programação Genética (PG) são as duas principais frentes de pesquisa em CE. Os Algoritmos Genéticos (AG) foram desenvolvidos no início dos anos 60 por John Holland, na Universidade de Michigan (Holland, 1962), com o objectivo inicial de estudar os fenómenos relacionados com a adaptação das espécies e a selecção natural que ocorre na natureza (Darwin, 1859), bem como desenvolver uma maneira de incorporar estes conceitos na computação (Mitchell, 1997). O método de Holland envolve uma população de cromossomas (candidatos a solução). Os cromossomas são vectores binários e as operações de pesquisa são, tipicamente, o cruzamento, a mutação e a inversão. Os cromossomas são avaliados por uma função de avaliação ou de aptidão.

Os AGs possuem uma larga aplicação em muitas áreas científicas, entre as quais podem ser citados problemas de optimização de soluções, aprendizagem de máquinas, análise de modelos económicos, problemas de engenharia, diversas aplicações na biologia, tais como a simulação de bactérias, os sistemas imunológicos, os ecossistemas e a descoberta de formato e propriedades de moléculas orgânicas (Mitchell, 1997).

A Programação Genética (PG) é uma técnica de geração automática de programas de computador criada por John Koza (Koza, 1989, 1992), inspirada na teoria de AGs de Holland. Em PG é possível criar e manipular geneticamente *software*, aplicando conceitos inspirados na Biologia, para gerar programas computacionais de forma automática. Em PG, um indivíduo é um programa de computador. Os resultados produzidos pela execução do programa são avaliados. Esta avaliação representa o valor da função de aptidão do indivíduo.

Normalmente os programas de computador são representados em estruturas em árvore e os operadores de pesquisa, que permitem a evolução dos programas, são o cruzamento e a mutação.

A diferença essencial entre AG e PG consiste no facto das estruturas manipuladas em PG e operações realizadas pelo algoritmo serem bastante mais complexas. Ambas as técnicas partilham a mesma base teórica, inspirada na competição pela sobrevivência entre indivíduos, porém não mantêm vínculos de dependência ou subordinação.

PG e AGs representam um campo novo de pesquisa dentro da Ciência da Computação. Neste campo muitos problemas continuam em aberto na tentativa de serem encontradas novas soluções e ferramentas. Apesar disso, este paradigma tem-se mostrado bastante poderoso e muitos trabalhos exploram o uso de AGs e PG para solucionar problemas em diferentes áreas do conhecimento, desde tratamento de dados e biologia molecular, até ao projecto de circuitos eléctricos e algoritmos de controlo.

1.2. Estrutura de um AE

Os AEs baseiam-se na aprendizagem colectiva de um conjunto de candidatos a soluções através da imitação do processo natural de evolução. Esta secção apresenta as etapas necessárias à estruturação de um AE, assim como os operadores que têm que ser implementados.

1.2.1. Codificação das soluções

Na maioria dos casos, a pesquisa não é realizada directamente no espaço das soluções do problema, mas sim no espaço das representações. Os membros da população que codificam as soluções de um AE são membros do espaço das representações.

Cada candidato a solução é representado como um indivíduo (ou cromossoma) da população e os novos indivíduos de uma determinada população são criados por operadores cuja intenção consiste em simular o *cruzamento* natural (ou *recombinação*) e a *mutação*. Desta forma, obtém-se uma nova geração de soluções. Os indivíduos são comparados através de uma função de aptidão. O cruzamento e a sobrevivência são guiados pelo valor da função de aptidão.

Se S for o *espaço de soluções* de um dado problema, então o espaço dos indivíduos X será considerado como a representação do *espaço* (ou *representação*) de pesquisa do sistema. Uma solução do problema s é codificada por um indivíduo $C(s) \in X$, onde C é a função de codificação:

$$C : S \rightarrow X^1 \quad (2.1)$$

1.2.2. Selecção e operadores de pesquisa

Considerando $P(t)$ a população de indivíduos na geração t , onde $t = 0, 1, 2, \dots$, a população descendente é gerada através da *selecção* e de operadores de *pesquisa*

¹ Observações:

- i) o espaço de soluções S e o espaço de representações X podem coincidir;
- ii) os membros do espaço de soluções S são referidos como fenótipos;
- iii) os membros do espaço de representações X são normalmente designados por cromossomas, genótipos, genomas ou indivíduos.

(ou *variação*). Podem também ser usados outros operadores, como a *inversão* e operadores específicos de cada problema.

1.2.2.1. Função de aptidão

A avaliação dos indivíduos de uma população é feita com base no valor real da *função de aptidão* f :

$$f : X \rightarrow R \quad (2.2)$$

É usado um mecanismo de *sobrevivência*, ou *substituição*, dos descendentes, ou da população dos pais para seleccionar os indivíduos para a nova geração.

A população inicial $P(0)$ é, regra geral, gerada aleatoriamente. Em alguns casos particulares é possível usar um conhecimento específico sobre o problema e iniciar a pesquisa numa determinada região apropriada do espaço de pesquisa.

Os membros de $P(0)$ são avaliados através da função de aptidão.

1.2.2.2. Operador de cruzamento

O operador de *cruzamento* R é usado para criar novos indivíduos por combinação da informação genética de dois ou mais pais, sendo definido como uma aplicação:

$$R : X^p \rightarrow X^q \quad (2.3)$$

Neste caso, o operador de cruzamento realiza uma transformação (p,q) onde p pais são combinados para obter q descendentes.

1.2.2.3. Operador de mutação

O operador de *mutação* M gera novos indivíduos através de variações (normalmente pequenas) de um único indivíduo de cada vez. O objectivo da mutação é a introdução de diversidade na população de soluções para evitar a estagnação da pesquisa numa região inadequada do espaço de pesquisa. Esta estagnação da pesquisa é conhecida como *convergência prematura* do processo de pesquisa.

O operador de mutação pode ser representado como uma aplicação:

$$M : X \rightarrow X \quad (2.4)$$

1.2.2.4. Selecção

É possível considerar dois tipos de selecção, nomeadamente a *selecção para cruzamento* e a *selecção para substituição*.

O operador de selecção para cruzamento é usado para decidir que membros da população actual $P(t)$ serão usados como pais da nova geração.

O operador de selecção para substituição é usado para saber que indivíduos de $P(t)$ e respectivos descendentes irão efectivamente fazer parte da nova geração $P(t+1)$.

1.2.3. Componentes de um AE

Os principais componentes de um AE podem ser identificados como se segue:

- Esquema de representação das potenciais soluções;
- Modelo da população;
- Módulo de avaliação (realizado pela função de aptidão ou por outros mecanismos);
- Operadores de pesquisa (variação);
- Estratégia de selecção de pais (obtida pelo operador de selecção para cruzamento);
- Estratégia de sobrevivência (selecção ambiental) (obtida pelo operador de selecção para substituição).

1.3. AE básico

Um AE genérico, com selecção, cruzamento e mutação, pode ser delineado tal como se apresenta na figura 2.1.

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

Figura 2. 1 - Algoritmo evolutivo básico

1.3.1. Condição de conclusão

Na grande maioria das situações, a condição de conclusão dos AEs refere-se a um número de gerações previamente especificado. Outra condição de conclusão que pode ser usada é a estabilização da população, quando, ao fim de um determinado número de gerações sucessivas, não ocorre nenhuma modificação na população. Se for possível formular um critério para soluções óptimas ou aceitáveis, então também este pode ser usado para estabelecer um predicado de sucesso.

1.3.2. Resultado de um AE

Os métodos básicos para escolha do resultado de um AE são os especificados a seguir (Dumitrescu *et al.*, 2000):

- i) A solução é codificada pelo melhor indivíduo da população final;

- ii)* A solução é dada pela população final toda;
- iii)* A solução é dada por um subconjunto dos melhores indivíduos da população final;
- iv)* A solução é dada pelo melhor indivíduo encontrado em todas as gerações do algoritmo evolutivo (neste caso é necessário usar uma estratégia elitista).

1.4. Aplicações dos AEs

As aplicações dos AEs estão apenas limitadas pela capacidade de codificação adequada do problema e do estabelecimento do procedimento de avaliação das soluções, ou seja, da função de aptidão ou de outra medida de qualidade.

As aplicações dos AEs incluem praticamente todas as áreas onde são usados os métodos tradicionais de optimização e as técnicas baseadas em IA (especialmente os métodos de aprendizagem automática).

O facto de existir um amplo número de aplicações relevantes torna difícil estabelecer uma selecção que se possa considerar completa. No entanto, com base em alguns autores, apresenta-se um conjunto extenso de aplicações desta área. Davis (Davis, 1991) descreve em detalhe uma grande maioria das primeiras aplicações de CE, enquanto Bäck, Fogel e Michalewicz (Bäck *et al*, 1997) indicam um vasto leque de aplicações.

1.4.1. Aplicações gerais

Os métodos de pesquisa e optimização baseados em CE têm numerosas aplicações práticas em vários domínios distintos. A seguir apresenta-se uma lista de alguns domínios de aplicação importantes.

Sistemas de computação distribuída e paralela

As aplicações de CE lidam com estimação de tempo e com optimização da comunicação inter-processador ou inter-processo em ambientes de computação distribuídos, configuração de processadores e testes automáticos de circuitos numéricos. Uma aplicação muito interessante nesta área consiste na implementação automática e simultânea de programas sequenciais.

Química

As aplicações na área da química tentam optimizar os processos e estratégias de produção. Estes métodos são também usados no projecto tecnológico e na análise e interpretação de dados. Outras aplicações são o controlo de processos químicos quer em laboratório quer a nível industrial, e o controlo ambiental, nomeadamente o controlo de poluição.

Engenharia

O domínio das aplicações em engenharia é vasto e inclui o projecto tecnológico, simulação, teste, controlo de processos em tempo real, telecomunicações (optimização de redes, controlo de tráfego, desenho de dispositivos),

processamento de imagem (reconhecimento de escrita manual, reconhecimento de impressões digitais e reconhecimento de voz) e reconhecimento de cenários complexos.

Negócios

As aplicações de negócios incidem essencialmente na modelização de sistemas económicos complexos e na previsão de sistemas. Uma possível aplicação na área financeira aborda a modelização baseada em séries temporais e na previsão de mercados.

Medicina

Na área da medicina, as aplicações principais são a análise de dados e o diagnóstico automático (*e.g.*, detecção de sinais que indicam epilepsia).

Gestão

Têm sido propostas algumas aplicações práticas no domínio da gestão. A maioria delas aparece no contexto industrial, com ênfase no processo produtivo, o que parece natural, dado que actualmente a produção envolve complexas tarefas de optimização.

Outros domínios

Existem muitas outras aplicações, tais como a alocação de recursos, escalonamento, optimização de sistemas de transporte e localização óptima de objectos.

2. Inteligência dos Enxames

A Inteligência dos Enxames (IEs), do inglês *Swarm Intelligence*, é uma técnica baseada no estudo do comportamento de grupos (colectivo) em sistemas descentralizados e auto-organizados. Na IE incluem-se as metaheurísticas inspiradas em agentes biológicos.

A expressão *swarm intelligence* foi introduzida por Beni e Wang em 1989 (Beni e Wang, 1989) no contexto dos sistemas robóticos celulares. Os sistemas IE são tipicamente constituídos por uma população de agentes simples interagindo localmente uns com os outros e, também, com o ambiente onde se encontram inseridos. Ainda que não exista uma estrutura de controlo centralizada que imponha um determinado comportamento aos agentes, as interacções locais entre os agentes geralmente conduzem a um comportamento global. Pode-se encontrar na natureza vários exemplos deste tipo de sistemas, nomeadamente as colónias de formigas, os bandos de pássaros, as manadas de animais e os cardumes de peixes (figuras 2.2 a 2.5).



Figura 2. 2 - Colónia de formigas



Figura 2. 3 - Bando de pássaros



Figura 2. 4 - Manada de animais



Figura 2. 5 - Cardume de peixes

A *Optimização por Colônias de Formigas*, do inglês *Ant Colony Optimization* (ACO), é um algoritmo metaheurístico de otimização que pode ser usado para encontrar soluções aproximadas em problemas de otimização combinatória, considerados difíceis. O seu princípio baseia-se na simulação do comportamento de um conjunto de agentes que cooperam para resolver um problema de otimização por meio de comunicações muito simples.

As formigas conseguem encontrar com alguma facilidade o caminho desde o ninho até uma fonte de comida mesmo quando têm que ultrapassar obstáculos. Na maioria dos casos esta capacidade é o resultado da comunicação química entre as formigas e um fenómeno emergente causado pela presença de muitas formigas, conforme representado na figura 2.6 (Dorigo e Gambardella, 1997).

O comportamento do modelo computacional não segue exactamente o modelo de colónias de formigas naturais (por exemplo, as formigas reais deixam o rasto durante o seu movimento, e não depois de alcançarem o destino final, como no caso computacional). Tal como nos AGs as colónias de formigas baseiam-se numa população de agentes, e não só num único agente (solução).

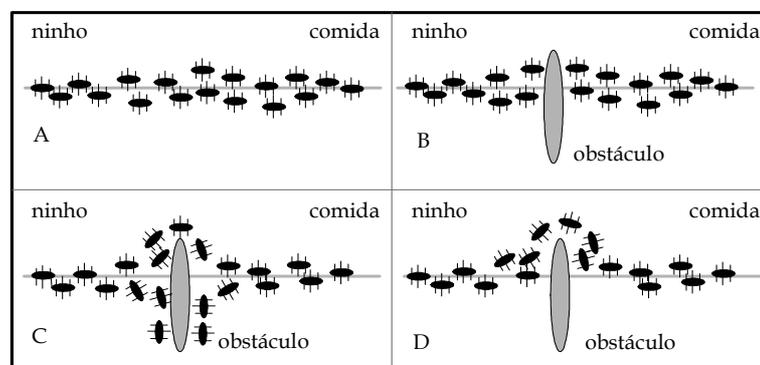


Figura 2. 6 - Comportamento das formigas após o aparecimento de um obstáculo entre o ninho e uma fonte de alimento.

A otimização por Enxame de Partículas, do inglês *Particle Swarm Optimization* - PSO) é um algoritmo de otimização heurística baseado no comportamento do movimento de animais (peixes, pássaros, etc). O PSO foi proposto por Eberhart e Kennedy em 1995 (Kennedy e Erberhart, 1995) e consiste na otimização de uma função objectivo através da troca de informações entre elementos (partículas) do grupo, resultando num algoritmo de otimização não determinístico que resulta eficiente, robusto e de simples implementação computacional.

O movimento de cada partícula em cada iteração corresponde à soma de três parcelas distintas: a primeira é uma parcela relativa à inércia da partícula e que traduz o modo como a partícula se está a movimentar; a segunda é uma parcela relativa à atracção da partícula ao melhor ponto que já encontrou; e a terceira parcela é relativa à atracção da partícula ao melhor ponto que todo grupo (ou uma parte do grupo) já encontrou.

Uma característica interessante do PSO consiste na realização, durante o procedimento de otimização, de uma pesquisa global nas iterações iniciais. Nesta etapa ocorre a prospecção de várias regiões do espaço de pesquisa, viabilizando a localização de óptimos globais. Com o decorrer das iterações a pesquisa passa a ser uma pesquisa local em torno do valor mais promissor, chegando a valores próximos do ponto óptimo. Entretanto, o desempenho do algoritmo é bastante dependente da selecção dos valores dos seus parâmetros.

Apesar de ser recente, o PSO tem encontrado diversas aplicações, sendo exemplos os trabalhos de Cockshott e Hartman (Cockshott e Hartman, 2001) e Costa Jr. (Costa Jr. *et al.*, 2003) onde o PSO é usado na otimização de processos específicos, de Ourique (Ourique *et al.*, 2002) que utilizaram o PSO na análise dinâmica não linear de processos químicos e Parsopoulos e Vrahatis

(Parsopoulos e Vrahatis , 2002) que aplicaram o PSO em problemas multi-objectivo.

3. Algoritmos Genéticos

3.1. Evolução e selecção natural

A teoria da evolução é uma das maiores revoluções intelectuais de toda a história da humanidade, mudando drasticamente a nossa percepção do mundo e o nosso posicionamento face a essa realidade. Charles Darwin apresentou uma teoria de evolução forte e coerente baseada em evidências comprovadas.

Até meados do século XIX, os naturalistas acreditavam que cada espécie tinha sido criada separadamente, por um ser supremo ou através de uma geração espontânea. O trabalho do naturalista Carolus Linnaeus, sobre a classificação biológica de organismos, despertou o interesse pela similaridade entre certas espécies, levando a acreditar na existência de uma certa relação entre elas. Este sistema de classificação biológica ainda é usado actualmente.

Outros trabalhos influenciaram os naturalistas em direcção à teoria da selecção natural, tais como os de Jean Baptiste Lamarck e de Thomas Robert Malthus. Lamarck propôs a evolução orgânica como explicação das similaridades dentro de grupos de organismos e um mecanismo de alterações adaptativas baseado na herança de características adquiridas. Embora Lamarck estivesse incorrecto com a hipótese do mecanismo, o seu exemplo tornou claro a possibilidade de alterações evolutivas das espécies.

Depois de mais de 20 anos de observações e experiências, Charles Darwin apresentou, em 1858, a sua teoria da evolução através de selecção natural,

simultaneamente com outro naturalista inglês Alfred Russel Wallace. No ano seguinte, Darwin publica o seu livro "*On the Origin of Species by Means of Natural Selection*" (a figura 2.7 ilustra a primeira capa deste livro) com a sua teoria completa, sustentada por muitas evidências colhidas durante as suas viagens a bordo do navio *Beagle*. Este trabalho influenciou muito o futuro, não apenas da Biologia, da Botânica e da Zoologia, mas também teve grande influência sobre o pensamento religioso, filosófico, político e económico da época.

A teoria de evolução de Darwin divide-se nas quatro partes principais que se apresentam a seguir:

- Ao longo do tempo os organismos foram sofrendo modificações. Os organismos que vivem actualmente são diferentes dos que viveram no passado. Além disso, muitos dos organismos que viveram no passado estão agora extintos. O mundo não é estático, bem pelo contrário encontra-se em constante evolução. A informação dada pelos fósseis fornece uma ampla evidência nesta matéria;
- Todos os organismos derivam de antepassados comuns através de um processo de ramificação. À medida que o tempo passa as populações vão-se dividindo em espécies diferentes, embora relacionadas, uma vez que descendem de um antepassado comum. Assim, se se recuar no tempo, qualquer par de organismos tem um antepassado comum. Este facto explica as similaridades dos organismos que foram classificados em conjunto - eles são similares porque partilham traços herdados do antepassado que têm em comum. Também explica a razão pela qual as espécies similares tendem a ocorrer na mesma região geográfica;
- A evolução é lenta e gradual, ocorrendo num período longo de tempo. Esta teoria encontra suporte pelo registo que se tem dos

fósseis e é consistente com o facto de nenhum naturalista ter observado o aparecimento súbito de uma nova espécie.

- O mecanismo das alterações evolutivas é a selecção natural. Esta foi a parte da teoria de Darwin mais importante e revolucionária.

A selecção natural é um processo que ocorre ao longo de gerações sucessivas e está ilustrado na figura 2.8.

A teoria da evolução e a computação nasceram praticamente na mesma época: Charles Babbage, um dos fundadores da computação moderna e amigo pessoal de Darwin desenvolveu uma máquina analítica em 1833.

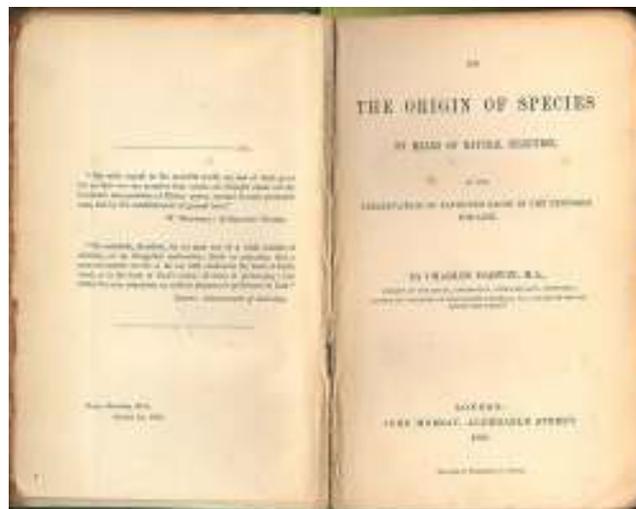


Figura 2. 7 - Capa da primeira publicação do livro "The origin of Species" de Charles Darwin

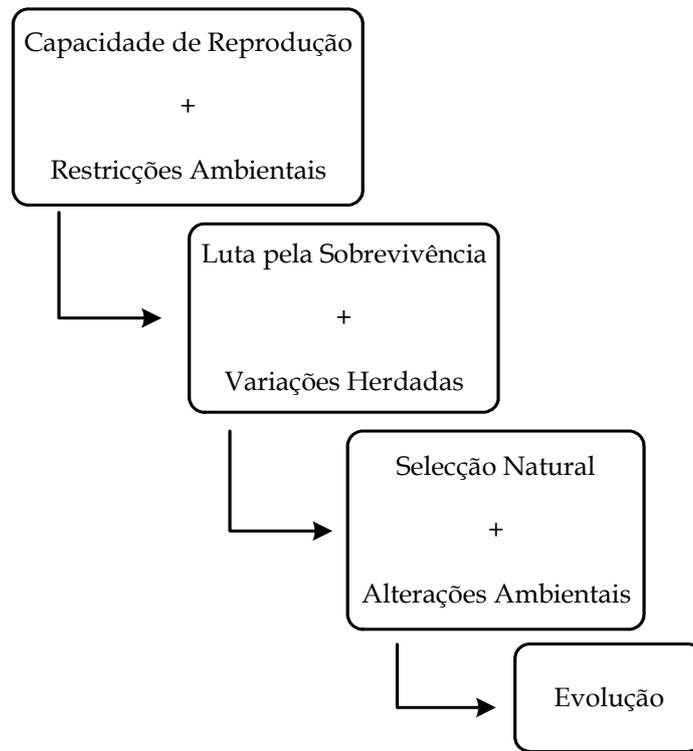


Figura 2. 8 – O processo da selecção natural

O trabalho de Gregor Mendel, desenvolvido em 1865, sobre os princípios básicos de herança genética, foi redescoberto pelos cientistas por volta de 1900 e teve grande influência sobre trabalhos relacionados com a evolução. A moderna teoria da evolução combina a genética e as ideias de Darwin e Wallace sobre a selecção natural, criando o princípio básico de Genética Populacional: a variabilidade entre indivíduos, numa população de organismos que se reproduzem sexualmente, é produzida pela mutação e pela recombinação genética. Este princípio foi desenvolvido durante os anos 30 e 40, por biólogos e matemáticos de importantes centros de pesquisa.

Entre os anos 50 e 60, vários cientistas da computação estudaram sistemas evolutivos na expectativa de que estes poderiam ser usados como uma ferramenta de optimização para problemas na engenharia. Os sistemas

desenvolvidos pretendiam gerar uma população de candidatos a solução para um dado problema.

Tal como referido na secção 1.1, os AGs foram inventados por John Holland nos anos 60 e desenvolvidos por alunos na Universidade de Michigan. O principal objectivo de Holland não foi tanto desenvolver algoritmos para solucionar problemas específicos, mas dedicar-se ao estudo formal do fenómeno de evolução, como ocorre na natureza, e desenvolver maneiras de importá-lo para os sistemas de computação.

Em meados dos anos 70, John Holland acreditou que as peculiaridades da Evolução Natural poderiam ser implementadas de forma algorítmica a fim de alcançar uma versão computacional dos processos de evolução. A tabela 2.1 apresenta a analogia entre os AGs e o sistema natural.

Tabela 2.1 - Terminologia natural/computacional

Termo natural	Termo utilizado nos AGs
Cromossoma	Palavra binária, vector
Gene	Parâmetro, característica
Alelo (<i>allele</i>)	Valor
Lugar (<i>locus</i>)	Posição no vector
Genótipo	Estrutura de codificação
Fenótipo	Estrutura de descodificação
Indivíduo	Solução
Geração	Iteração

3.2. Princípios básicos dos AGs

Muitos dos problemas que a Inteligência Artificial (IA) tenta resolver envolvem a procura de uma solução num espaço vasto de candidatos que se encontra sujeito a restrições de tempo e espaço. Quando, *a priori*, não existe nenhum conhecimento sobre o problema, não se podem usar estratégias estáticas específicas do domínio. É neste contexto que surgem os AGs (Tomassini, 1998) como uma forma de resolver problemas de pesquisa adaptativa nos quais o conhecimento para controlo da procura é obtido dinamicamente.

As espécies naturais lutam permanentemente pela sobrevivência tentando adaptar-se ao ambiente em que estão inseridas. É sabido que, ao longo dos tempos, algumas espécies têm desaparecido enquanto outras novas têm surgido. O ponto de vista de que os seres vivos evoluem de acordo com o princípio da selecção natural, defendido por Charles Darwin no século XIX, é hoje consensualmente aceite. A teoria da selecção natural foi complementada com os estudos sobre genética iniciados por Mendel. Em termos gerais este paradigma estipula que:

- Cada indivíduo transmite aos seus descendentes as suas características (genéticas);
- Numa população existem indivíduos com características diferentes;
- Os indivíduos com melhores características (mais adaptados) tendem a reproduzir-se mais, pelo que uma população caminha no sentido da existência em maior número dos elementos mais adaptados ("mais fortes");
- Ao longo do tempo a acumulação de pequenas variações pode originar o aparecimento de novas espécies com melhor capacidade de sobrevivência;

- Os processos de troca e de aparecimento de material genético novo podem concorrer para uma melhor adaptabilidade.

A reprodução nos seres vivos envolve as células que os constituem. Estas possuem um núcleo que contém cromossomas (aos pares ou não). É nestes últimos que se encontram os genes responsáveis pela definição dos traços característicos dos indivíduos. Existem três mecanismos de base para explicar a dinâmica das populações:

- **Reprodução adaptativa:** as espécies reproduzem-se aumentando o número dos mais aptos (*fitness*);
- **Cruzamento** (*crossover*): novas combinações de genes existentes;
- **Mutação** (*mutation*): aparecimento de novos genes.

Em termos gerais a passagem de uma geração à seguinte faz-se de acordo com o princípio da reprodução adaptativa perturbado pelos mecanismos de cruzamento e mutação.

Os algoritmos genéticos são uma versão computacional simplificada do que se passa na natureza (Goldberg, 1989). Qualquer problema a ser resolvido nesta abordagem passa pela identificação de um conjunto de elementos que se apresentam na tabela 2.2 e pela definição dos seus valores.

As tarefas de pesquisa e optimização possuem vários componentes, entre os quais: o espaço de pesquisa, onde são consideradas todas as possibilidades de solução de um determinado problema e a função de avaliação (ou função de custo, aptidão ou objectivo), que implementa uma maneira de avaliar os membros do espaço de pesquisa. Existem muitos métodos de pesquisa e funções de avaliação. As técnicas de pesquisa e optimização tradicionais iniciam-se com um único candidato que é manipulado iterativamente

utilizando algumas heurísticas (estáticas) directamente associadas ao problema a ser solucionado.

Geralmente estes processos heurísticos não são algorítmicos e a simulação em computadores pode ser muito complexa. Apesar destes métodos não serem suficientemente robustos, isto não implica que eles sejam inúteis. Na prática, eles são amplamente utilizados, com sucesso, em inúmeras aplicações. Por outro lado, as técnicas de computação evolutiva operam, em paralelo, com uma população de candidatos. Assim, elas podem fazer a pesquisa em diferentes áreas do espaço de solução, encontrando um número de membros apropriado para a pesquisa em várias regiões.

Tabela 2. 2- Elementos de um AG

geração	uma população num dado instante do tempo
população	conjunto de indivíduos
indivíduo	definido pelos seus cromossomas (tipicamente um)
cromossoma	conjunto de genes
gene	codificação de uma característica genética podendo tomar diferentes valores
função de aptidão	para medir a qualidade do indivíduo
cruzamento	a probabilidade de haver troca de material genético entre dois indivíduos
mutação	a probabilidade de aparecer material genético novo

Os Algoritmos Genéticos (AGs) diferem dos métodos tradicionais de pesquisa e otimização principalmente em quatro aspectos:

- Os AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;
- Os AGs trabalham com uma população de candidatos e não com um único candidato;
- Os AGs utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
- Os AGs utilizam regras de transição probabilísticas em vez de determinísticas.

Os AGs são muito eficientes para pesquisa de soluções óptimas, ou aproximadamente óptimas, numa grande variedade de problemas, pois não impõem muitas das limitações encontradas nos métodos de pesquisa tradicionais.

A figura 2.1, apresentada na secção 1.3 como algoritmo evolutivo básico, ilustra também o funcionamento geral de um AG. Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que são possíveis soluções do problema. Durante o processo evolutivo esta população é avaliada, sendo atribuída uma classificação a cada indivíduo, reflectindo a sua capacidade de adaptação a determinado ambiente (Davis, 1991, Michalewicz, 1992).

Os indivíduos mantidos pela selecção podem sofrer modificações das suas características fundamentais através de mutações e do cruzamento gerando descendentes para a geração seguinte. Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada.

Apesar da simplicidade do ponto de vista biológico, estes algoritmos são suficientemente ricos para fornecer mecanismos de pesquisa poderosos e robustos.

3.3. Características Gerais dos AGs

Os AGs são algoritmos de otimização global, baseados nos mecanismos de selecção natural e da genética. Eles empregam uma estratégia de pesquisa paralela e estruturada, mas aleatória, que se baseia no reforço da pesquisa de indivíduos de "alta aptidão", ou seja, em candidatos a soluções nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos).

Apesar de aleatórios, os AGs não seguem um percurso não direccionado, pois exploram informações históricas para encontrar novos indivíduos com melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é denominada de geração.

Durante cada iteração, os princípios de selecção e reprodução são aplicados a uma população de candidatos que pode variar, dependendo da complexidade do problema e dos recursos computacionais disponíveis. Através da selecção, determina-se quais os indivíduos que irão entrar no processo de reprodução, gerando um número determinado de descendentes para a geração seguinte, com uma probabilidade determinada pelo seu índice de aptidão. Por outras palavras, os indivíduos com maior adaptação relativa têm maiores probabilidades de se reproduzirem.

O ponto de partida para a utilização de Algoritmos Genéticos, como ferramenta para solução de problemas, é a representação. A maioria das

representações são genótípicas, isto é, utilizam vectores de tamanho finito num alfabeto finito.

Tradicionalmente, os indivíduos são representados genotipicamente por vectores binários, onde cada elemento de um vector denota a presença (1) ou ausência (0) de uma determinada característica: o seu genótipo.

Os elementos podem ser combinados formando as características reais do indivíduo, ou o seu fenótipo. Teoricamente, esta representação é independente do problema, pois uma vez encontrada a representação em vectores binários, as operações padrão podem ser utilizadas, facilitando o seu emprego em diferentes classes de problemas.

O princípio básico do funcionamento dos AGs baseia-se no critério de selecção, o que vai fazer com que, depois de várias gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A maioria dos métodos de selecção são projectados para escolher preferencialmente indivíduos com maiores valores de aptidão, embora não exclusivamente, a fim de manter a diversidade da população. Um método de selecção muito utilizado é o Método da Roleta, onde os indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta, tal como ilustra o exemplo da figura 2.9.

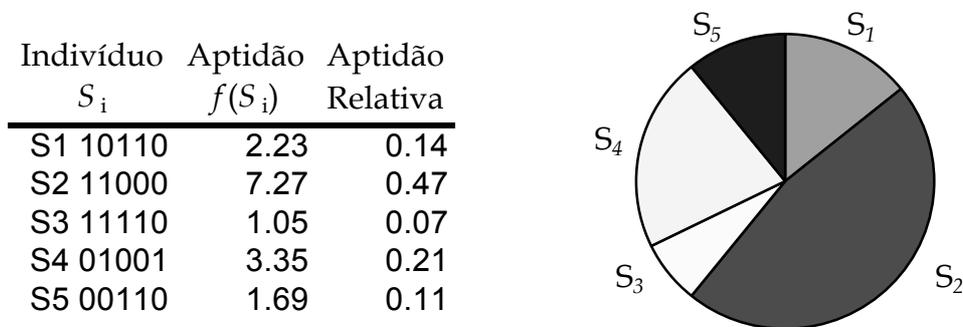


Figura 2. 9 - Método da Roleta

Neste método cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor da roleta. Finalmente, a roleta gira um determinado número de vezes, dependendo do tamanho da população, e são escolhidos, como indivíduos que participarão da próxima geração, os sorteados na roleta.

Existem ainda os métodos de selecção baseada no posto, a selecção por torneio (*tournament selection*) e a selecção de Boltzmann.

Na selecção por nível (posto) são usados os índices dos indivíduos, ordenados de acordo com os valores de aptidão, para calcular a probabilidade de selecção correspondente.

Foram também propostas as projecções:

lineares:

$$p(\text{nível}) = q - (\text{nível} - 1)r \quad (2.5)$$

não lineares:

$$p(\text{nível}) = q(1 - q)^{\text{nível}-1} \quad (2.6)$$

Ambas as funções determinam a probabilidade de um indivíduo ser escolhido numa amostra. Considera-se que o indivíduo se encontra cotado numa escala de nível, onde $\text{nível} = 1$ quando o indivíduo em questão é o melhor da população e $\text{nível} = 0$ quando é o pior da população.

As duas funções satisfazem a equação:

$$\sum_{i=1}^{\mu} p_i = 1 \quad (2.7)$$

ou seja:

$$q = 0,5r(\mu - 1) + \mu^{-1} \quad (2.8)$$

onde r é um parâmetro escolhido pelo utilizador e μ é o número de indivíduos da população.

O parâmetro r está relacionado com a pressão do método de selecção. Para $r = 0$ não existe pressão no método, ou seja, todos os indivíduos têm a mesma probabilidade de serem seleccionados, e para $r = 2 / (\text{nível}^2 - \text{nível})$ a selecção do método é máxima.

Na selecção por torneio é retirada uma amostra de q ($q > 1$) cromossomas da população, seguindo uma lei uniforme aleatória. O melhor cromossoma da amostra é copiado para a geração seguinte e a operação é repetida até que o número de cromossomas seleccionados preencha a nova geração. Este método é de fácil implementação computacional e permite apurar o peso da selecção pelo aumento ou diminuição do tamanho do torneio (q).

Por último, a selecção de Boltzmann tem semelhanças com o método anterior considerando $q = 2$, neste caso os dois cromossomas competem entre si de acordo com a fórmula:

$$\frac{T}{1 + e^{f(i) - f(j)}} \quad (2.9)$$

onde T é a temperatura e $f(k)$ é a função de aptidão do cromossoma k .

Para que, a partir de uma dada população, se consiga gerar populações sucessivas com melhores funções de aptidão, é necessário utilizar um conjunto de operações ou operadores. Estes operadores são o cruzamento (*crossover*) e a mutação. Eles são utilizados para assegurar que a geração seguinte seja totalmente nova, mas possui, de alguma forma, características dos seus pais. Por outras palavras, a população diversifica-se mantendo as características de adaptação adquiridas pelas gerações anteriores.

Para prevenir que os melhores indivíduos não desapareçam da população através da manipulação dos operadores genéticos, eles podem ser automaticamente colocados na geração seguinte, através da reprodução elitista. Numa estratégia elitista os indivíduos com melhores funções de aptidão são mantidos na geração seguinte. O melhor cromossoma de $P(t)$ é transferido para $P(t+1)$ após o cruzamento e a mutação, o que reduz o efeito aleatório do processo de selecção e garante que o melhor indivíduo da próxima geração é melhor ou igual ao da geração anterior.

A estrutura de um AG simples é fornecida pelo algoritmo da figura 2.1, onde $P(t)$ é a população na geração t .

Estes algoritmos, apesar de serem computacionalmente muito simples, são bastante poderosos. Além disso, eles não estão limitados por suposições sobre o espaço de pesquisa, relativas a continuidade, existência de derivadas, e outras.

A pesquisa em problemas reais lida com discontinuidades, ruído e outros problemas. Métodos que dependam fortemente de restrições de continuidade e da existência de derivadas são adequados apenas para problemas num domínio limitado.

3.4. Fundamentos Matemáticos dos AG's

Pode-se compreender melhor o princípio de funcionamento dos AGs a partir da Teoria dos Padrões (*schemata*) formulada por John Holland em 1975 (Holland, 1975).

John Holland definiu *schema* como um padrão que descreve um conjunto de cromossomas com similaridades em algumas posições. Assim, para um espaço de pesquisa representado por K^L existem $(K+1)^L$ padrões, sendo K o número de símbolos do alfabeto (*i. e.* a cardinalidade do alfabeto) e L o comprimento do cromossoma.

Então $K = 2$ e $L = 3$ definem um espaço de pesquisa de 8 pontos e o padrão $H=11\Sigma$ descreve o conjunto de elementos 111 e 110. Para se compreender melhor o porquê dos AGs funcionarem, basta analisar o efeito dos processos de selecção, recombinação e mutação sobre os padrões. Ou seja, interessa saber o que acontece, iteração a iteração, com os representantes de determinado grupo, aqueles indivíduos que possuem o padrão H . Nesta análise utilizam-se duas definições:

- $\phi(H)$: ordem ou especificidade de um padrão (*schema*), como o número de posições fixas diferentes de S ;
- $\delta(H)$: comprimento do padrão (*schema*), como a distância entre as primeira e última posições fixas.

3.4.1. Análise do operador de selecção no teorema de padrões

Seja $m(H, t)$ o número de representações de H numa iteração t do algoritmo com n indivíduos na população. Assim, é possível calcular o número provável de representantes de H na iteração seguinte como:

$$m(H, t + 1) = n \frac{\sum_{i \in H} f_i}{\sum_{j=1}^n f_j} \quad (2.10)$$

Define-se $f(H)$ como a aptidão média do padrão (*schema*) H :

$$f(H) = \frac{\sum_{i \in H} f_i}{m(H, t)} \quad (2.11)$$

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

$$m(H, t + 1) = m(H, t) n \frac{f(H)}{\sum_{j=1}^n f_j} \quad (2.12)$$

A aptidão média da população, dada de seguida pela equação 2.13, permite fazer uma última transformação na fórmula de $m(H, t+1)$.

$$\bar{f} = \frac{\sum_{j=1}^n f_j}{n} \quad (2.13)$$

$$m(H, t+1) = m(H, t) \frac{f(H)}{\bar{f}} \quad (2.14)$$

A interpretação desta equação reflecte o efeito da selecção nos AGs, nomeadamente:

- 1) Padrões (*schemata*) com aptidão acima da média tendem a proliferar nas gerações seguintes;
- 2) Padrões com aptidão abaixo da média tendem a desaparecer.

Para se estimar a eficiência do processo de evolução supõe-se um padrão H com aptidão acima (ou abaixo) da média de um factor C estático a partir de $t = 0$. Assim tem-se:

$$m(H, t+1) = m(H, t) \frac{\bar{f} + C\bar{f}}{\bar{f}} = m(H, t)(1 + C) \quad (2.15)$$

Portanto:

$$m(H, t+1) = m(H, 0)(1 + C)^t \quad (2.16)$$

o que significa que o número de representantes de H em gerações sucessivas cresce (ou decresce) exponencialmente durante a evolução.

3.4.2. Análise do operador de cruzamento no teorema de padrões

Aqui interessa analisar o impacto produzido por um corte num indivíduo que destrua um padrão e, por conseguinte, não permita a transição aos descendentes. Seja $p_d(H)$ a probabilidade de um determinado padrão H ser destruído quando é aplicado o cruzamento num ponto:

$$p_d(H) = \frac{\delta(H)}{L-1} \quad (2.17)$$

Portanto,

$$p_s(H) = 1 - \frac{\delta(H)}{L-1} \quad (2.18)$$

é a probabilidade de sobrevivência do padrão H .

Sendo p_c a taxa de cruzamento, e considerando-se que o par progenitor de um cromossoma pode recuperar parte de um padrão destruído pelo cruzamento, tem-se a inequação:

$$p_s \geq 1 - p_c \frac{\delta(H)}{L-1} \quad (2.19)$$

o que significa que os padrões curtos têm maior probabilidade de sobrevivência (*i.e.* de se manterem intactos) após o cruzamento.

3.4.3. Análise do operador de mutação no teorema de padrões

Se p_m for a probabilidade de mutação, então a probabilidade de sobrevivência de um cromossoma é dada por:

$$p_s = (1 - p_m)^{O(H)} \quad (2.20)$$

Para taxas de mutação $p_m \ll 1$:

$$p_s \cong 1 - p_m O(H) \quad (2.21)$$

o que significa que cromossomas de baixa ordem têm maior probabilidade de não serem destruídos pelo operador de mutação.

Combinando o efeito dos três operadores, selecção, cruzamento e mutação, tem-se:

$$m(H, t + 1) \geq m(H, t) \frac{f(H)}{f} \left[1 - p_c \times \frac{\delta(H)}{L - 1} \right] \left[1 - p_m \times O(H) \right] \quad (2.22)$$

A interpretação da expressão 2.22 resume o teorema fundamental dos AGs, que afirma:

“padrões curtos e de baixa ordem têm tendência a proliferar ou a desaparecer nas gerações seguintes, de acordo com a aptidão média”

3.5. Funcionamento dos AG's

Esta secção apresenta a base do funcionamento dos AGs sendo a adaptabilidade o conceito genérico que envolve o funcionamento dos algoritmos evolutivos.

A adaptabilidade é o resultado da função de avaliação ($f(x)$) que define a adaptação do indivíduo ao ambiente.

3.5.1. Processo de Selecção

O Processo de selecção consiste nos seguintes passos: Calcula-se o valor de adaptabilidade $f(v_i)$ para cada cromossoma v_i , onde ($i = 1, \dots, T_{pop}$);

Encontra-se a adaptabilidade total da população;

$$F = \sum f(v_i) \quad (2.23)$$

Calcula-se a probabilidade de selecção p_i para cada cromossoma v_i , onde ($i = 1, \dots, T_{pop}$);

$$p_i = f(v_i) / F \quad (2.24)$$

Calcula-se a probabilidade cumulativa q_i para cada cromossoma v_i , onde ($i = 1, \dots, T_{pop}$);

$$q_i = \sum P_j \quad (2.25)$$

Gera-se um número aleatório r no intervalo $[0,1]$. Se $r < q_i$ então selecciona-se o primeiro cromossoma (v_1); senão selecciona-se o i -ésimo cromossoma v_i tal que $q_{(i-1)} < r \leq q_i$.

3.5.2. Operadores Genéticos

O princípio fundamental dos operadores genéticos consiste em transformar a população através de sucessivas gerações, estendendo a pesquisa até chegar a um resultado satisfatório. Os operadores genéticos são necessários para que a população se diversifique mantendo, no entanto, as características de adaptação adquiridas pelas gerações anteriores.

O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, como é ilustrado a seguir, fornecendo, assim, meios para introdução de novos elementos na população. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de pesquisa nunca será nula, além de contornar o problema de mínimos locais pois, com este mecanismo, altera-se levemente a direcção da pesquisa. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação P_m . Geralmente a taxa de mutação é pequena.

Exemplo de mutação:

Antes da Mutação	1 1 1 0 0
Depois da Mutação	1 1 0 0 0

O cruzamento é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. É considerado o operador genético predominante, e por isso é aplicado com probabilidade dada pela taxa de cruzamento P_c , que deve ser maior que a taxa de mutação.

Este operador pode, ainda, ser utilizado de várias maneiras, sendo as mais usuais:

- **Um-ponto:** um ponto de cruzamento é escolhido e a partir deste ponto são trocadas as informações genéticas dos pais. As informações anteriores a este ponto num dos pais são ligadas às informações posteriores a este ponto no outro pai, como é mostrado no exemplo da figura 2.10.
- **Multi-pontos:** é uma generalização desta ideia de troca de material genético através de pontos, onde pode ser utilizado mais do que um ponto de cruzamento.
- **Uniforme:** não utiliza pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

Um exemplo de cruzamento de um ponto:

- a) São escolhidos dois indivíduos.
- b) É escolhido um ponto (4) de cruzamento.
- c) São recombinaadas as características, gerando dois novos indivíduos.

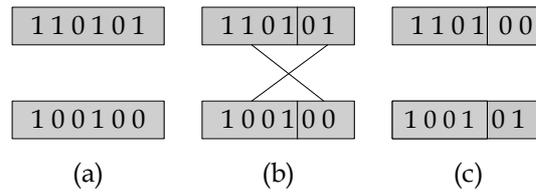


Figura 2. 10 - Exemplo de cruzamento num ponto

3.5.3. Parâmetros Genéticos

É importante analisar também de que maneira certos parâmetros influenciam o comportamento dos Algoritmos Genéticos, para que seja possível estabelecê-los conforme as necessidades do problema e dos recursos disponíveis.

Tamanho da População. O tamanho da população afecta o desempenho global e a eficiência dos AGs. Com uma população pequena o desempenho pode ser baixo, pois, deste modo, a população fornece uma pequena cobertura do espaço de pesquisa do problema. Uma grande população fornece geralmente uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais em vez de soluções globais. No entanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais, ou que o algoritmo trabalhe por um período de tempo muito maior.

Taxa de Cruzamento. Quanto maior for esta taxa, mais rapidamente serão introduzidas novas estruturas na população. Todavia, se esta taxa for muito alta, significa que a maior parte da população será substituída o que implica que se existirem estruturas com boas aptidões elas poderão desaparecer da população, também rapidamente. Com um valor baixo, o algoritmo pode tornar-se muito lento.

Taxa de Mutação. Uma baixa taxa de mutação previne que uma dada posição fique estagnada num valor, além de possibilitar que se chegue a qualquer ponto do espaço de pesquisa. Com uma taxa muito alta a pesquisa torna-se essencialmente aleatória.

Intervalo de Geração. Controla a percentagem da população que será substituída durante a próxima geração. Com um valor alto, a maior parte da população será substituída, mas com valores muito altos pode ocorrer perda de estruturas de alta aptidão. Com um valor baixo, o algoritmo pode tornar-se muito lento.

3.5.4. Tamanho da população

Como já foi referido anteriormente, a definição do tamanho da população é crucial para o sucesso do AG. Por este motivo, alguns investigadores têm vindo a analisar esta questão e a propor formas de a resolver (Goldberg *et. al.*, 1992; Harik *et. al.*, 1997 e Miller, 1997). O conceito de Bloco Construtor (BC) está na base do entendimento e solução deste problema.

3.5.4.1. Fornecimento de BCs

O assunto do fornecimento de BCs ao AG foi levantado por Holland (Holland, 1973). Este estudo foi seguido por vários outros investigadores, como De Jong (De Jong, 1975), Grefenstette (Grefenstette, 1986), Goldberg (Goldberg, 1989 e 2002), Alander (Alander,1992) e Reeves (Reeves, 1993), que abordaram o melhoramento da teoria da definição do tamanho da população. Recentemente,

foi estimada a população mínima constituída por material com suficientes BCs e verificada experimentalmente por Goldberg, Sastry and Latoza (Goldberg *et. al.*, 2002).

A questão de como se selecciona um tamanho de população adequado para cada domínio particular é bastante complexa.

Se a população for demasiado pequena, não é provável que o AG encontre uma solução boa ou aceitável para o problema em causa, executando apenas algumas gerações. Por outro lado, se a população for aumentada, de forma a incorporar soluções de elevada qualidade, o AG vai despende tempo a processar indivíduos sem interesse ou até mesmo desnecessários, o que pode resultar numa diminuição de desempenho inaceitável. O problema resume-se em encontrar o tamanho de população ideal, *i. e.* suficientemente grande para permitir uma exploração correcta do espaço de pesquisa sem desperdiçar recursos computacionais.

Existem duas abordagens à questão do fornecimento de BCs, uma espacial e a outra temporal. A espacial procura ajustar o tamanho da população adequadamente para garantir, probabilisticamente, que estão presentes logo de início BCs suficientes em número e em diversidade, fornecendo assim ao AG o material que vai permitir a pesquisa subsequente. A abordagem temporal assume que a existência de mutação (através do operador de mutação), ou de outro mecanismo de diversificação, devolve BCs diversificados numa escala de tempo apropriada (Goldberg, 2002).

3.5.4.2. Tomada de decisão de BCs

O problema da tomada de decisão quanto aos BCs reside no reconhecimento de que o AG pode cometer erros quando opta entre um BC e o seu concorrente. O procedimento de escolha entre o BC melhor e o segundo melhor numa dada partição foi discutido por Goldberg, Deb e Clark em 1992 (Goldberg *et. al.*, 1992). Vai rever-se essa discussão de forma a obter-se uma probabilidade de decisão dependente do domínio.

Considerem-se dois BCs, H_1 (com função de aptidão média f_{H_1} e variância $\sigma_{H_1}^2$) e H_2 (com função de aptidão média f_{H_2} e variância $\sigma_{H_2}^2$), em que as distribuições das funções de aptidão são normais e H_1 tem a função de aptidão mais elevada. Então espera-se seleccionar mais representações de H_1 do que H_2 . No entanto, numa competição um-a-um existe a possibilidade de, erradamente, ser escolhida a representação H_2 . Para calcular esta probabilidade é preciso acumular todos os valores possíveis da área da região onde as distribuições das duas funções de aptidão dos dois padrões se sobrepõem. Esta computação é designada por convolução e, no caso de variáveis normais, a própria convolução é normalmente distribuída e tem propriedades conhecidas. A média da convolução é a diferença das médias das duas distribuições individuais e a variância é o somatório das variâncias individuais (Goldberg e Rudnick, 1991).

Definindo a diferença de sinal d , como $d = f_{H_1} - f_{H_2}$, a probabilidade de cometer um erro numa única tentativa é calculada por:

$$p = \Phi \left(\frac{d}{\sqrt{\sigma_{H_1}^2 + \sigma_{H_2}^2}} \right) \quad (2.26)$$

onde Φ é a função da distribuição acumulada para uma distribuição normal com média de zero e desvio padrão unitário.

Dado que o problema consiste em múltiplas competições de BCs executadas em paralelo, é necessário especializar este resultado para entrar em linha de conta com o ruído introduzido por outros BCs. Segundo Goldberg, Deb e Clark em 1992 (Goldberg *et. al.*, 1992), assume-se que a função de aptidão é o somatório das m sub-funções independentes f_i , todas de tamanho K igual ao da partição mais ilusória. A variância total da função pode ser calculada por:

$$\sigma_f^2 = \sum_{i=1}^m \sigma_{f_i}^2 \quad (2.27)$$

A média da variância dos BCs é calculada por: $\sigma_{BC}^2 = \sigma_f^2 / m$. Para cada competição entre BCs, existe um ruído que resulta do outro BC ($m' = m - 1$): $\sigma^2 = m' \sigma_{BC}^2$. Desta forma, a probabilidade de tomar a decisão correcta entre BCs é dada por:

$$p = \mathfrak{N}\left(\frac{d}{\sqrt{2m' \sigma_{BC}^2}}\right) \quad (2.28)$$

3.5.4.3. Modelo da Ruína do Jogador

Os passeios aleatórios (*random walks*) são ferramentas matemáticas que podem ser usadas para prever o resultado de certos processos estocásticos. O passeio aleatório mais básico lida com uma partícula que se movimenta, com

certas probabilidades, para a esquerda e para a direita num espaço uni-dimensional. O tamanho do passo é fixo e muitas vezes o movimento da partícula é restringido colocando barreiras em alguns pontos do espaço. Neste caso, considera-se um passeio uni-dimensional com duas barreiras absorventes que capturam a partícula assim que esta alcance uma delas.

O problema da ruína do jogador é um exemplo clássico de um passeio aleatório com barreiras absorventes e foi estudado exaustivamente. Neste problema, o capital do jogador (apostador) é representado pela posição, x , da partícula no espaço uni-dimensional. Inicialmente, a partícula é posicionada em $x = a$, onde a representa o capital inicial do jogador. O apostador joga contra um adversário que tem capital inicial $n - a$, e existem barreiras absorventes em $x = 0$ (representando bancarrota) e em $x = n$ (representando ganhar todo o capital do adversário). A cada etapa do jogo, o jogador tem probabilidade p de aumentar o seu capital de uma unidade e a probabilidade $q = 1 - p$ de perder uma unidade a favor do seu adversário.

Para modelizar o processo de decisão nos AGs como um passeio aleatório a concentração é focada em, apenas, um BC. O espaço é demarcado por barreiras absorventes em $x = 0$ e em $x = n$, que representam respectivamente as soluções errada e correcta. A posição inicial da partícula, x_0 , é o número de réplicas esperado do melhor BC numa população inicializada aleatoriamente, que é igual a $x_0 = n/2^k$, onde k é a ordem do BC.

Este modelo abandona a noção de gerações e considera que as decisões ocorrem uma de cada vez até todos os n BCs terem convergido para o mesmo valor. Também se assume que o AG usa selecção por torneio.

A abordagem usada para calcular a probabilidade da partícula ser capturada em $x = n$, é a de usar um resultado bem conhecido para a probabilidade de captura.

Da literatura dos passeios aleatórios (Feller, 1966) sabe-se que a probabilidade de uma partícula ser capturada em $x = n$ é:

$$P_n = \frac{1 - (q/p)^{x_0}}{1 - (q/p)^n} \quad (2.29)$$

onde $q = 1 - p$ é a probabilidade de tomar a decisão errada entre dois BCs.

É simples encontrar a expressão para o cálculo do tamanho da população a partir da equação 2.29. Note-se que, neste caso, $p > 1 - p$ e x_0 é normalmente pequeno comparado com o tamanho da população. Substituindo $x_0 = n/2^k$ e $q = 1 - p$ simplifica-se P_n a:

$$P_n \approx 1 - \left(\frac{1-p}{p} \right)^{n/2^k} \quad (2.30)$$

Se $\alpha = 1 - P_n$ for a probabilidade de falha, então resolvendo a equação 2.29 em relação a n , temos:

$$n = 2^k \ln(\alpha) / \ln\left(\frac{1-p}{p} \right) \quad (2.31)$$

onde p , a probabilidade de tomar a decisão correcta entre dois BCs, é dada pela equação 2.28.

A equação 2.31 pode ser aproximada, sabendo que p se pode também aproximar usando os primeiros dois termos da expansão em série de potências para a distribuição normal (Abramovitz e Stegun, 1972):

$$p \approx \frac{1}{2} + \frac{1}{2}x \quad (2.32)$$

onde $x = d / (\sigma_{BC} \sqrt{\pi n})$. Substituindo a aproximação de p na equação 2.31, tem-se:

$$n = 2^k \ln(\alpha) / \ln\left(\frac{1-x}{1+x}\right) \quad (2.33)$$

Como x é um número pequeno, pode fazer-se a aproximação $\ln(1-x) \approx -x$ e $\ln(1+x) \approx x$. Substituindo estas aproximações e o valor de x , obtém-se:

$$n = 2^{k-1} \ln(\alpha) \frac{\sigma_{BC} \sqrt{\pi n}}{d} \quad (2.34)$$

Usando esta aproximação pode constatar-se que a população cresce com o aumento da variância de BCs e à medida que o tamanho do problema aumenta. Por outro lado, um maior sinal de diferença d torna o problema mais fácil e, conseqüentemente, a população necessária torna-se mais pequena.

4. Algoritmos Meméticos

4.1. Introdução

No campo da computação evolutiva o refinamento do conhecimento pode ser incorporado como uma etapa de apoio ao processo evolutivo. Em 1992, Moscato e Norman (Moscato e Norman, 1992) introduziram o termo “algoritmo

memético” (AM) para descrever um processo evolutivo que possui uma pesquisa local como parte decisiva na evolução. Esta pesquisa pode ser caracterizada como sendo um refinamento local dentro de um espaço de pesquisa. O termo *meme* foi idealizado por Dawkins (Dawkins, 1976) como sendo uma unidade de informação que se reproduz durante um processo argumentativo e de transmissão de conhecimento (Radcliffe e Surry, 1994). Portanto, pode-se dizer que, enquanto o algoritmo memético está relacionado com a evolução cultural, o algoritmo genético está baseado na evolução biológica dos indivíduos.

Encontra-se no processo de transmissão aos descendentes uma diferença marcante entre genes e *memes*. Quando o *meme* é transmitido, ele é adaptado pela entidade que o recebe com base no seu conhecimento e para melhor atender às suas necessidades. Quanto aos genes, no processo de evolução eles são transmitidos de uma maneira tal que o descendente gerado herda muitas habilidades e características presentes nos seus progenitores.

Desta maneira, o algoritmo genético é inspirado na tentativa de emulação computacional da evolução biológica e o algoritmo memético tenta fazer o mesmo em relação à evolução cultural. É interessante lembrar que na evolução biológica a informação está codificada nos genes por uma sequência de nucleotídeos, de modo que a transmissão é influenciada pela presença de mutação, recombinação e pela seleção natural em relação aos indivíduos geneticamente melhor adaptados. Na evolução cultural, a informação envolvida está nos *memes*, por isso as alterações surgem pela combinação, criação e reorganização das representações mentais (conscientes ou não) e pela possível ineficácia dos mecanismos de transmissão de informação. A replicação (fenótipo) ocorre quando essas representações mentais são transformadas em ações passíveis de imitação ou expressas através de alguma linguagem. A incorporação dessa nova informação por parte de algum indivíduo certamente

alterará a pressão selectiva e a influência vinculada às limitações impostas pelo ambiente.

4.2. Características dos AMs

De acordo com Moscato (Moscato, 1999), o uso genérico da denominação de AM é feito para a identificação de uma classe de meta-heurísticas, constituindo um dos procedimentos de maior sucesso para problemas de optimização combinatória.

A característica principal, presente em muitas implementações que utilizam algoritmos meméticos, é o uso de processos de pesquisa dedicados. Esses processos pretendem utilizar toda a informação disponível sobre o problema, de modo que este conhecimento seja incorporado sob a forma de heurísticas, técnicas de pesquisa local, operadores especializados de recombinação e de muitas outras maneiras (Moscato e Cotta, 2003).

Em essência, os AMs podem ser interpretados como um conjunto de estratégias que implementam a competição e a cooperação entre diferentes mecanismos de optimização (Moscato e Norman, 1992). Sendo assim, o sucesso obtido pode ser explicado como sendo uma consequência directa da sinergia dos diferentes processos de pesquisa utilizados. Na figura 2.11 está ilustrado o algoritmo memético.

A ideia geral dos AMs é a utilização dos operadores evolutivos, que determinam regiões promissoras no espaço de pesquisa, combinados com pesquisa local nestas regiões (este processo tem sido aplicado com sucesso em vários problemas de optimização (Merz e Freisleben, 1999)). Pode-se também

dizer que os AMs correspondem à união de um método de pesquisa global e de uma heurística local aplicada a cada indivíduo, de modo que um AM é, na verdade, um tipo especial de *hillclimbing*.

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

Figura 2. 11 - Algoritmo memético

Num ambiente que empregue um AM, os operadores de recombinação e de mutação agem como estratégias de diversificação (Merz e Freisleben, 1999). Os indivíduos da população podem estar localizados numa região do espaço de pesquisa contendo um ótimo local, chamada base de atracção do ótimo local.

Utilizando a informação contida na população, podem ser descobertos novos pontos de partida após a pesquisa local. Os operadores de recombinação e de mutação podem gerar indivíduos da população que estejam localizados em bases de atracção de ótimos locais ainda não explorados, de modo que um novo pico deva ser alcançado (maximização) ou um vale deva ser explorado (minimização). A figura 2.12 ilustra estes eventos, no caso da maximização, utilizando o conceito de superfície de adaptação (*fitness*). Após a recombinação, o filho gerado pode possuir uma função de aptidão baixa, mas um grande

potencial para crescimento, de modo que uma pesquisa local pode levar o descendente a assumir um valor da função de aptidão elevada.

Por sua vez, a mutação, pode levar a um pequeno aumento (ou decréscimo) da função de aptidão, por representar uma perturbação local junto à representação do indivíduo.

Na presença de restrições, os operadores genéticos de recombinação e de mutação podem produzir soluções que estão fora da região factível do espaço de pesquisa. Entretanto, um algoritmo de reparação pode ser elaborado para remeter o descendente gerado para uma região factível (Radcliffe e Surry, 1994). Embora também pudesse ser caracterizado como um processo de pesquisa local, este algoritmo de pesquisa será denominado algoritmo de reparação. A razão para a distinção entre uma pesquisa local, realizada por um algoritmo memético, e a pesquisa realizada por um algoritmo de reparação, é que este último não investe no aumento da qualidade da solução, mas apenas na garantia de que se trata de uma solução possível para o problema.

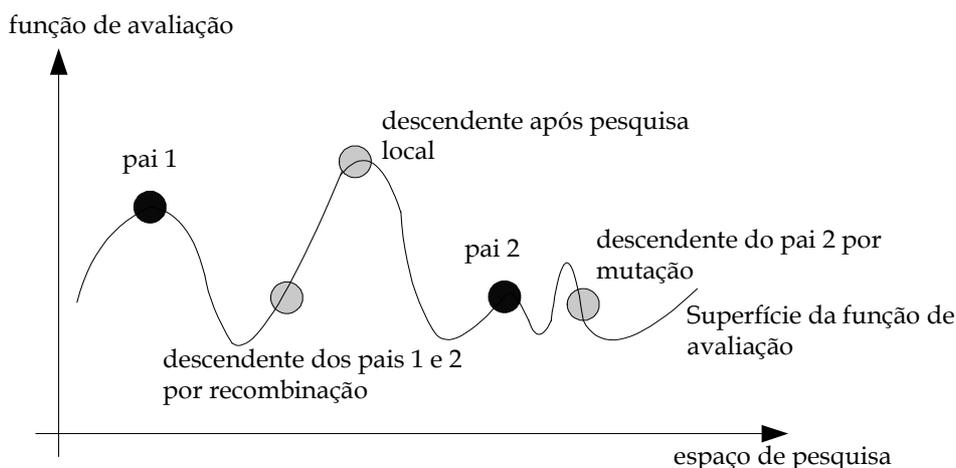


Figura 2. 12 - Operadores de recombinação e mutação agindo como estratégias de diversificação nos algoritmos meméticos

No caso de problemas de otimização sem restrições, a geração da população inicial é realizada em duas etapas. A primeira etapa, consiste na geração de indivíduos que procuram explorar amplamente o espaço de pesquisa, caracterizando-os como soluções candidatas. Na segunda etapa, são implementados procedimentos de pesquisa local para a obtenção de um ótimo associado à região do espaço onde se encontra cada indivíduo. Esse mesmo processo de pesquisa é empregado após a aplicação de qualquer operador genético (Merz e Freisleben, 1999), procurando remeter o descendente gerado a um ótimo local.

No caso de problemas de otimização com restrições, na geração da população inicial podem surgir indivíduos não factíveis, não tanto porque possuam um genótipo mal formado ou porque o algoritmo de solução utilizado contenha erros, mas sim por violarem restrições impostas ao problema. Portanto, há a necessidade de um dispositivo de selecção que seja capaz de discernir entre indivíduos factíveis ou não. Em algumas aplicações é possível implementar algoritmos de reparação para levar indivíduos a serem soluções admissíveis (eventualmente à admissibilidade mais próxima). Uma vez tendo essa necessidade atendida, o emprego de uma pesquisa local sobre os indivíduos factíveis é uma consequência natural, fazendo com que eles possam atingir um ótimo local dentro da região factível. Portanto, resulta como abordagem de solução um algoritmo memético (ou algoritmo genético associado a mecanismos de pesquisa local) aliado a etapas de reparação, tanto para conduzir os indivíduos à admissibilidade, como para aumentar a sua função de avaliação.

4.3. Pesquisa Local

Aarts e Verhoeven (Aarts e Verhoeven, 1997) consideram a pesquisa local como sendo uma aproximação geral para problemas de otimização combinatória baseados na exploração de vizinhanças.

De uma forma genérica, um algoritmo de pesquisa local começa com um indivíduo $s_0 \in S$ e tenta, continuamente, encontrar melhores elementos nos vizinhos. Por outras palavras, a finalidade é remeter esse indivíduo s_0 para um local onde a função de avaliação tenha um valor melhor que o actual (figura 2.12). Essa pesquisa deverá ser realizada até que uma condição de conclusão seja satisfeita, sendo que essa condição deve ser atendida sempre que o processo de pesquisa não tenha mais a capacidade de melhorar a solução actual. A figura 2.13 apresenta o algoritmo representativo de um processo de pesquisa local.

```
1. início
2.  repetir
3.     final = verdade;
4.     para  $i = 1$  até número de indivíduos faça;
5.         início
6.             elemento = indivíduo( $i$ );
7.             aplicar pesquisa local (elemento);
8.             se função aptidão é melhor então final=falso;
9.         fim
10.    até final
11. fim
```

Figura 2. 13 - Algoritmo genérico representativo de uma pesquisa local

Moscato (Moscato, 1999) apresenta uma definição formal de pesquisa local que se apresenta em seguida:

Um problema computacional P tem domínio de entrada I_p , com $x \in I_p$, podendo ser estabelecido um conjunto $resp_p(x)$ de respostas correspondentes. Entretanto, é preciso garantir que exista um subconjunto de soluções $sol_p(x) \subseteq resp_p(x)$ que identifique as respostas factíveis de P . Um algoritmo soluciona um problema P se, para a entrada $x \in I_p$, apresentar como saída qualquer $y \in sol_p(x)$ - solução factível - ou, no caso de $sol_p(x) = \{\}$, indicando que não existe y . A otimização combinatoria é um tipo especial de problema de pesquisa, em que cada $x \in I_p$ tem um conjunto $sol_p(x)$ de cardinalidade finita e cada solução $y \in sol_p(x)$ tem um valor de *fitness* $m_p(x,y)$. A pesquisa, neste tipo de problema, será responsável por encontrar uma solução factível $y^* \in sol_p(x)$ que maximize a *fitness*, $m_p(x,y)$.

Quando um bom algoritmo de pesquisa local é aplicado a um problema de otimização com múltiplos ótimos locais, o algoritmo converge para um ótimo local e para a sua execução.

O ótimo local encontrado depende do ponto de partida (solução inicial) do algoritmo de pesquisa local. O algoritmo de pesquisa local encontrará a solução ótima global se o ponto de partida (solução inicial) se encontrar na vizinhança do ótimo global (Papadimitriou e Steiglitz, 1982).

4.3.1. Espaço de pesquisa e vizinhanças

Dada uma solução admissível $f \in F$ num problema particular, define-se vizinhança como um conjunto $N(f)$ de pontos que estão, segundo um determinado critério, "próximos" do ponto f .

Definição 4.1 (Instância). Instância de um problema de otimização é um par (F, c) , onde F é o domínio (*i. e.* o conjunto dos pontos admissíveis) e c é a função de custo, ou função de aptidão.

O problema consiste em encontrar uma solução $f \in F$ para a qual $c(f) \leq c(y), \forall y \in F$. O ponto f é designado de solução óptima global, ou, quando não há risco de confusão, solução óptima.

Definição 4.2 (Problema de otimização). Chama-se problema de otimização ao conjunto de todas as instâncias de um mesmo problema.

Definição 4.3 (Vizinhança). Dado um problema de otimização com instâncias F, c , então uma vizinhança é uma aplicação $N : F \rightarrow 2^F$ definida para cada instância.

Se $F = \mathbb{R}$, o conjunto de pontos dentro de uma determinada distância euclidiana é a definição natural de vizinhança.

Definição 4.4 (Ótimo local). Dada uma instância (F, c) de um problema de otimização e uma vizinhança N , uma solução admissível $f \in F$ é chamada ótimo local com respeito a N sse $c(f) \leq c(g), \forall g \in N(f)$.

Depois de se obter a solução inicial efectua-se a procura local que utiliza uma função de melhoramento, função esta que, dada uma solução como argumento, devolve uma solução vizinha de melhor qualidade, ou então falha. Quando esta rotina falha, está-se na presença de um ótimo local.

Dada uma instância (F, c) de um problema de otimização (minimização), e uma vizinhança N que é explorada a partir de um ponto s , pode-se definir uma subrotina *melhorar*(s) que devolve:

- Uma solução $r \in N(s)$ com $c(r) < c(s)$ se existirem soluções que melhorem o ponto s ;
- *FALHA* no caso contrário.

5. *Optimização por Enxame de Partículas*

5.1. *Introdução*

A técnica de optimização PSO (*Particle Swarm Optimization*), traduzida para o português como Optimização por Enxame de Partículas (OEP), é uma técnica de computação estocástica baseada em populações, desenvolvida por (Kennedy e Eberhart, 1995), que implementa uma metáfora do comportamento social da interacção entre indivíduos (partículas) de um grupo (enxame) (Kennedy e Eberhart, 2001).

A metáfora foi desenvolvida a partir da observação de bandos de pássaros e cardumes de peixes à procura de alimento numa determinada região. Ao observar esses grupos verifica-se que o comportamento do grupo é influenciado pela experiência individual acumulada por cada indivíduo, bem como pelo resultado da experiência acumulada pelo próprio grupo. Isto é, o comportamento de um indivíduo é uma combinação da sua experiência passada (influência cognitiva) e da experiência da sociedade (influência social).

No algoritmo de OEP, designado por algoritmo PSO, cada candidato à solução do problema corresponde a um ponto no espaço de procura. Essas soluções potenciais, denominadas partículas, têm associado um valor que é avaliado individualmente para cada partícula, o qual indica a adequação da partícula como solução para o problema, e uma velocidade que define a direcção do movimento da partícula. Cada partícula vai modificando a sua velocidade com base na melhor posição da partícula e na melhor posição do grupo até ao momento, acabando por encontrar o alimento.

5.2. Comportamento social

Alguns investigadores criaram simulações por computador de várias interpretações do movimento de organismos de grupos de animais, tais como bandos de pássaros e cardumes de peixes. Reynolds (Reynolds, 1987) e Heppner e Grenander (Heppner e Grenander, 1990) apresentaram simulações notáveis sobre bandos de pássaros. Reynolds, em particular, ficou intrigado com a estética da coreografia dos bandos de pássaros (figura 2.14) e Heppner, como zoologista estava interessado em descobrir as regras que permitem que um número grande de pássaros voem em sincronismo, mudem de direcção repentinamente, se dispersem e reagrupem, etc. Estes cientistas lembraram-se que processos locais, como aqueles que são modelizados por autómatos celulares, poderiam encaixar a dinâmica (imprevisível) de grupo do comportamento social dos pássaros. Ambos os modelos se baseiam fortemente na manipulação de distâncias inter-individuais. Assim, o sincronismo do comportamento do grupo é realizado como uma função do esforço dos pássaros para manter uma distância óptima entre eles e os seus vizinhos. Não parece haver grande discrepância lógica se se suposer que algumas regras se aplicam

também ao comportamento social dos animais em geral, incluindo, rebanhos, bandos e cardumes e, até, seres humanos.

O socio-biologista E. O. Wilson (Wilson, 1975) escreveu, em referência aos cardumes de peixes “Pelo menos em teoria, os membros individuais do grupo podem beneficiar das descobertas e experiências anteriores de todos os outros membros do grupo durante a procura de alimentos. Esta vantagem pode tornar-se decisiva compensando as desvantagens da competição por alimentos, sempre que o recurso seja escasso.”(p.209). Esta afirmação sugere que a partilha social de informação entre as espécies oferece uma vantagem evolutiva: esta hipótese foi fundamental no desenvolvimento da OEP.



Figura 2. 14 - Voo típico em forma de V de um bando de pássaros

Um motivo para desenvolver a simulação foi a modelização do comportamento social humano que, como é evidente, não é idêntico ao comportamento dos cardumes e dos bandos. Uma diferença importante é a sua abstracção. Os pássaros e os peixes ajustam os seus movimentos físicos de modo a evitar os predadores, procurar alimentos, otimizar parâmetros ambientais, como a temperatura, etc. Os seres humanos ajustam não só movimentos físicos, mas também cognitivos. Não é normal andar a passo e

virar em salto (embora uma investigação fascinante em conformidade humana mostre que seria possível) mas, pelo contrário, tem-se tendência a ajustar as crenças e atitudes para estar em conformidade com os pares sociais. Esta é a distinção maior no que diz respeito à obtenção de uma simulação por computador, pelo menos por uma razão óbvia: colisão. Dois indivíduos podem ter atitudes e crenças idênticas sem colidirem, mas dois pássaros não podem ocupar a mesma posição no espaço sem haver colisão. Ao discutir o comportamento social humano, parece razoável tratar do conceito de alteração para o movimento análogo dos pássaros/peixes. Isto é consistente com a visão clássica de Aristóteles sobre a alteração qualitativa/quantitativa do movimento. Por isso, apesar do movimento tri-dimensional no espaço físico, evitando as colisões, os seres humanos movem-se num espaço multi-dimensional abstracto, livre de colisões.

5.3. A Etiologia da Optimização por Enxame de Partículas

Uma primeira simulação satisfatória foi escrita de imediato. Esta simulação teve por base duas propostas: a velocidade do vizinho mais próximo e a “*craziness*”. A população de pássaros começou por ser inicializada aleatoriamente, quanto às posições e quanto às velocidades (X e Y). Em cada iteração um ciclo no programa determinava, para cada agente (indivíduo ou pássaro), qual o agente mais próximo e atribuía a esse agente as velocidades X e Y . Essencialmente, esta regra simples criava um sincronismo no movimento dos pássaros. Infelizmente, esta estratégia convergia numa mudança não unânime de direcção do bando. Então, foi introduzida uma variável estocástica designada por “*craziness*”. Em cada iteração é adicionada uma alteração aleatória às velocidades X e/ou Y . Isso

significa que é introduzida uma variação no sistema que dá à simulação um aspecto interessante e mais de acordo com o que se passa na natureza.

As simulações de Heppner sobre pássaros tinham a característica de introduzir uma força dinâmica à simulação. Os seus pássaros voavam em torno de um “poleiro”, isto é, de uma determinada posição que os atraía até finalmente aí aterrarem. Esta situação eliminava a necessidade de ter uma variável do tipo “*craziness*”, dado que a simulação tomava vida por si própria. Por um lado a ideia do poleiro era intrigante, por outro levantou uma questão ainda mais estimulante. Os pássaros de Heppner sabiam onde se encontrava o poleiro, mas na vida real eles aterram em qualquer árvore ou fio de telefone que estão de acordo com as suas necessidades imediatas. Mais importante ainda é que os bandos de pássaros aterram onde há comida. Coloca-se então a questão: como é que eles sabem onde é que há comida? Qualquer pessoa que tenha dado de comer a pássaros sabe que durante muito tempo eles vão encontrar a comida sem terem nenhum conhecimento prévio da sua localização, aspecto, etc. Parece possível que algo na dinâmica do bando permite que os seus membros adquiram o conhecimento uns dos outros, tal como no modelo de Wilson já apresentado.

Uma segunda variação nesta simulação é definida como o “vector de milharal”, um vector de duas dimensões com coordenadas XY . Cada agente era programado para avaliar a sua posição corrente em termos de equação de forma a que na posição $(100,100)$ o valor fosse zero. Cada agente tem presente o seu melhor valor e a respectiva posição XY . Este valor era chamado de $pbest[]$ e as posições $pbestx[]$ e $pbesty[]$ (os parêntesis indicam que se trata de vectores com número de elementos igual ao número de agentes). À medida que cada agente se movimentava através do espaço a avaliar as posições, as suas velocidades X e Y iam sendo ajustadas de uma maneira simples. Se fosse para a direita do seu $pbestx$, então a sua velocidade segundo X (chamada vx) era

ajustada negativamente de uma quantidade aleatória através de um parâmetro do sistema:

$$vx[i] = vx[i] - rand()p_increment \quad (2.35)$$

Se fosse para à esquerda do seu $pbestx$, então a sua velocidade vx era ajustada positivamente:

$$vx[i] = vx[i] + rand()p_increment \quad (2.36)$$

De forma idêntica, as velocidades Y , vy , eram ajustadas para cima ou para baixo, dependendo de o agente estar acima ou abaixo de $pbesty$. Em segundo lugar cada agente conhecia a melhor posição que já tinha sido encontrada até ao momento pelo bando e o seu valor. Mais uma vez as velocidades $vx[i]$ e $vy[i]$ eram ajustadas como se indica a seguir, onde $g_increment$ é também um parâmetro do sistema:

$$\text{se } presentx[i] > pbestx[gbest] \text{ então } vx[i] = vx[i] - rand()g_increment \quad (2.37)$$

$$\text{se } presentx[i] < pbestx[gbest] \text{ então } vx[i] = vx[i] + rand()g_increment \quad (2.38)$$

$$\text{se } presenty[i] > pbesty[gbest] \text{ então } vy[i] = vy[i] - rand()g_increment \quad (2.39)$$

$$\text{se } presenty[i] < pbesty[gbest] \text{ então } vy[i] = vy[i] + rand()g_increment \quad (2.40)$$

Na simulação, a posição (100,100) estava marcada por um círculo e os agentes eram representados por pontos coloridos. Era possível ver o bando a circular até encontrar o milharal. Com os parâmetros $p_increment$ e $g_increment$ relativamente elevados parecia que o bando era sugado violentamente para o milharal. Em poucas iterações todo o bando, normalmente 15 a 30 indivíduos, eram vistos dentro de um pequeno círculo ao redor do objectivo. Com $p_increment$ e $g_increment$ baixos, o bando pura e simplesmente voava à volta do objectivo, aproximando-se dele de uma forma natural, em sub-grupos sincronizados até que finalmente aterrava no alvo.

Quando se tornou claro que o paradigma tinha capacidade para otimizar funções lineares simples e de duas dimensões, foi importante identificar as partes essenciais do paradigma. Por exemplo, os autores viram rapidamente que o algoritmo funcionava bem e de forma realista sem a variável “*craziness*”, pelo que a eliminaram. Em seguida detectaram que a optimização ocorria mais rapidamente quando a velocidade do vizinho mais próximo era removida. O bando passou a ser um enxame bastante capaz de encontrar o milharal. As variáveis $pbest$ e $gbest$ e os respectivos incrementos são necessárias.

Conceptualmente $pbest$ assemelha-se a uma memória auto-biográfica, dado que cada indivíduo recorda a sua própria experiência, e o ajuste da velocidade associada a $pbest$ foi designada por “nostalgia simples”, uma vez que os indivíduos tendem a retornar ao lugar que mais os satisfiz no passado. Por outro lado, $gbest$ é conceptualmente similar a conhecimento publicitário, isto é, grupo normal ou padrão que os indivíduos tendem em seguir. Nas simulações, um valor elevado de $p_increment$ relativamente a $g_increment$ resulta num isolamento excessivo dos indivíduos no espaço do problema, e o contrário ($g_increment$ relativamente elevado) resulta num bando a apressar-se repentinamente em direcção a um mínimo local. Valores aproximadamente

iguais dos dois incrementos parecem resultar numa procura mais eficiente no domínio do problema.

5.3.1. Pesquisa Multidimensional

Embora o algoritmo modelize de forma impressionante um bando à procura de um milharal, os problemas mais interessantes de optimização não são nem lineares nem bi-dimensionais. Como um dos objectivos é modelizar o comportamento social o qual é multi-dimensional e livre de colisões, parece ser um passo simples para alterar $presentx$ e $presenty$ (e, é claro $vx[]$ e $vy[]$) de vectores de uma dimensão para matrizes DN , onde D é o número de dimensões e N o número de agentes. Nesta ordem de ideias, foram realizadas experiências com um problema multi-dimensional não linear: ajuste dos pesos para treinar uma rede neuronal de várias camadas. O algoritmo teve um desempenho muito bom na resolução deste problema; no entanto, havia algo que desagradava do ponto de vista estético e que era difícil de compreender. Os ajustes das velocidades foram baseados em testes simples de inequações: se $presentx > bestx$ então diminuía-se a velocidade, se $presentx < bestx$ então aumentava-se a velocidade. Desenvolvendo mais experiências foi possível melhorar o seu desempenho. Em vez de simplesmente se testar o sinal da inequação, as velocidades eram ajustadas de acordo com as diferenças entre as melhores posições:

$$vx[\] = vy[\] + rand()p_increment(pbestx[\] - presentx[\]) \quad (2.41)$$

onde os parâmetros vx e $presentx$ são agora matrizes de agentes.

5.3.2. Versão simplificada actual

Facilmente se percebeu que não havia uma forma sistemática de saber se os incrementos de p ou g deveriam ser maiores. Por este motivo estes dois termos foram retirados do algoritmo. O factor estocástico foi multiplicado por 2 para dar média de 1, fazendo assim com que os agentes sobrevoassem o alvo metade do tempo. Esta versão superou as versões anteriores. Na optimização por enxames de partículas simplificada corrente, as velocidades são ajustadas aplicando a seguinte fórmula:

$$\begin{aligned}
 vx[\mathbb{I}] = & vy[\mathbb{I}] + 2 \times rand() (pbestx[\mathbb{I}] - presentx[\mathbb{I}]) \\
 & + 2 \times rand() (pbestx[\mathbb{I}] - presentx[\mathbb{I}])
 \end{aligned}
 \tag{2.42}$$

Foram testadas outras variações do algoritmo, mas nenhuma apresentou melhorias de desempenho em relação à actual versão simplificada.

5.4. Enxames e Partículas

Tal como foi descrito nas secções anteriores, tornou-se óbvio durante a simplificação do paradigma que o comportamento da população de agentes é mais um enxame do que de um bando. O termo enxame é usado de acordo com Millonas (Millonas, 1994), que desenvolveu modelos para aplicação em vida artificial articulando cinco princípios básicos da inteligência dos enxames:

- O primeiro é o princípio da proximidade: a população tem que ser capaz de desenvolver cálculos computacionais simples no espaço e no tempo;

- O segundo é o princípio da qualidade: a população tem que ser capaz de responder a factores ambientais de qualidade;
- O terceiro é o princípio da resposta diversificada: a população não deve dirigir as suas actividades por canais excessivamente estreitos;
- O quarto é o princípio da estabilidade: a população não deve alterar o seu comportamento sempre que existem modificações ambientais;
- O quinto é o princípio da adaptabilidade: a população deve ser capaz de alterar o seu comportamento sempre que o seu custo computacional piore.

O conceito da optimização por enxame de partículas segue os cinco princípios apresentados. A população responde aos factores de qualidade *pbest* e *gbest*. A alocação de respostas entre *pbest* e *gbest* garante a diversidade da resposta. A população modifica o seu estado (comportamento) apenas quando o *gbest* se altera. O termo partícula foi escolhido como compromisso, dado que pode ser discutível que os membros da população possam não ter massa, nem volume. Por isso, são designados por pontos, no entanto velocidades e acelerações são mais apropriadas a partículas.

5.5. Algoritmo básico

A optimização por enxames de partículas é um algoritmo extremamente simples que demonstra ser bastante eficiente na optimização de várias funções. Este algoritmo é considerado como uma forma intermédia entre a vida artificial ou algoritmos inspirados na biologia e a pesquisa evolutiva ou processamento neuronal. Esta optimização está obviamente ligada à computação evolutiva. Conceptualmente, encontra-se entre os algoritmos genéticos e a programação

genética, estando altamente dependente de processos estocásticos, tal como a programação evolutiva. O ajustamento sucessivo em direcção a *pbest* e a *gbest* é, em conceito, similar à operação de cruzamento utilizada nos algoritmos genéticos. Da mesma forma, usa o conceito de função de aptidão, tal como todos os paradigmas da computação evolutiva.

O conceito original e único na optimização por enxames de partículas é fazer voar as potenciais soluções no hiper-espaço, acelerando em direcção às melhores soluções. Os outros esquemas de computação evolutiva trabalham directamente com as potenciais soluções as quais são representadas como posições do hiper-espaço.

Muito do sucesso dos enxames de partículas parece assentar no facto dos agentes terem tendência para ultrapassar o seu alvo. O capítulo do livro de Holland sobre “alocação óptima das experiências” revela o balanço delicado entre o teste cuidadoso de regiões conhecidas *versus* a exploração arriscada de regiões desconhecidas. Na versão corrente do paradigma acontece a alocação de experiências próximo do óptimo. Os factores estocásticos permitem a pesquisa de espaços entre regiões que demonstraram ser relativamente boas e de regiões desconhecidas do domínio do problema.

O algoritmo é escrito em muito poucas linhas de código e requer apenas a especificação do problema e alguns parâmetros para o resolver. Este algoritmo pertence à escola filosófica que permite que o conhecimento surja em vez de o querer impor e que procura emular a natureza ao contrário de a controlar. Uma vez mais a natureza fornece uma técnica de processar informação de forma elegante e versátil.

O algoritmo Particle Swarm Optimization (PSO) foi introduzido por James Kennedy e Russell Elberhart em 1995 e emergiu de experiências com algoritmos que modelam o comportamento social observado em muitas espécies de

pássaros (Pomeroy, 2003). Durante a procura de alimento ou do ninho os pássaros baseiam-se na sua própria experiência e na experiência do bando. O PSO é um algoritmo que possui um vector de velocidades e outro de posição, onde a posição de cada partícula é actualizada de acordo com a velocidade actual, o saber adquirido pela partícula e o saber adquirido pelo bando (Eberhart e Dobbins, 1990).

A figura 2.15 apresenta o pseudo-código que caracteriza de uma forma simples o algoritmo PSO.

```
1. início
2.   k=0;
3.   inicializar aleatoriamente P(k);
4.   repetir
5.     avaliação P(k);
6.     modificar a velocidade das partículas
       com base na melhor posição de cada
       partícula e na melhor posição do
       grupo até ao momento;
7.     k=k+1;
8.   até condição de conclusão verificada
9. fim
```

Figura 2. 15 - Algoritmo PSO

No contexto da optimização de uma partícula p no instante de tempo k , ela é representada pela sua posição corrente ($u^p(k)$), a sua melhor posição de sempre ($y^p(k)$) e a sua velocidade de viagem ($v^p(k)$).

Uma nova posição da partícula é actualizada por:

$$u^p(k+1) = u^p(k) + v^p(k+1) \quad (2.43)$$

onde $v^p(k+1)$ é a nova velocidade dada por:

$$v_j^p(k+1) = \delta(k)v_j^p(k) + \mu\omega_{1j}(k)(y_j^p(k) - u_j^p(k)) + v\omega_{2j}(k)(\hat{y}_j(k) - u_j^p(k)) \quad (2.44)$$

para $j = 1, \dots, n$, onde $\delta(k)$ é o factor de inércia, μ é o parâmetro cognitivo e v é o parâmetro social. $\omega_{1j}(k)$ e $\omega_{2j}(k)$ são números aleatórios obtidos da distribuição uniforme $[0,1]$.

$\hat{y}(k)$ é a posição da partícula com melhor valor da função até ao momento, ou seja,

$$\hat{y}(k) = \arg \min_{\alpha \in A} f(\alpha) \quad (2.45)$$

$$A = \{y^1(k), \dots, y^s(k)\} \quad (2.46)$$

onde s é o número de partículas do enxame.

O cálculo da velocidade necessita, ainda, de alguns parâmetros dependentes do problema, nomeadamente: a inércia da partícula (w) que controla a capacidade de exploração do algoritmo (ou seja, um valor alto facilita um comportamento mais global, enquanto um valor baixo facilita um comportamento mais local (Venter e Sobieszczanski-Sobieski, 2002)) e os dois parâmetros de confiança $c1$ e $c2$ que indicam quanto uma partícula confia em si ($c1$) e no bando ($c2$). Os parâmetros de confiança e de inércia devem ser ajustados de acordo com o problema, pois são utilizados para a actualização do vector velocidade. Alguns autores propõem que sejam adoptados $c1 = c2 = 2$ e $0.7 < w < 1.4$. Sugere-se, também, a adopção de valores diferentes para $c1$ e $c2$ desde que satisfaçam a equação $c1 + c2 = 4$. A inércia pode ser actualizada de forma iterativa pela expressão:

$$w_{nova} = f_w w_{anterior} \quad (247)$$

considerando o factor de redução f_w uma constante compreendida entre 0 e 1.

Em suma, o algoritmo PSO trabalha com um conjunto de soluções que se pretende otimizar, as partículas, as quais evoluem progressivamente dentro do espaço de todas as alternativas, mostrando-se competitiva quando comparada com outras meta-heurísticas.

As partículas movem-se segundo três influências (vectores) que se compõem aditivamente: inércia (ou hábito), memória e cooperação. O primeiro vector impele a partícula numa direcção idêntica àquela que seguia. O segundo vector atrai a partícula na direcção da melhor posição por ela ocupada durante o seu período de vida. O terceiro vector atrai a partícula na direcção do melhor ponto do espaço encontrado pelo enxame até ao momento.

6. Resumo do capítulo

O presente capítulo apresenta os algoritmos evolutivos que estão na base das ferramentas de síntese de circuitos digitais propostas e implementadas neste trabalho: os algoritmos genéticos, os algoritmos meméticos e o algoritmo baseado na optimização de enxames de partículas. Para todos eles foram descritos os conceitos essenciais que permitem o seu entendimento.

Referências

- Aarts, E. e Verhoeven, G., Chapter 9.5: Genetic local search for the traveling salesman problem. In T. Back, D. Fogel, and Z. Michalewicz, editors, Handbook of Evolutionary Computation, pages G9.5:1-7. IOP publishing Ltd and Oxford University Press, 1997.
- Abramovitz, M. e Stegun, I., Handbook of Mathematic Functions with Formulas, Graphs, and Mathematical Tables, Dover Publications, 1972.
- Alander, J. T., "On Optimal Population Size of Genetic Algorithms"n Size for Binary-Coded Genetic Algorithm" in Proceedings of CompEuro 92, IEEE Computer Society Press, pp 65-70, 1992.
- Bäck, T., Fogel, D. B., Michalewicz, Z., Handbook of Evolutionary Computation, Institute of Physics Publishing, Bristol, Oxford University Press, Oxford, 1997.
- Banzhaf, W; Nordin, P.; Keller, R. E. and Francone, F. D. Genetic Programming: an introduction. ISBN 155860510X. Morgan Kaufmann, 1998.
- Barreto, J. Inteligência Artificial. No Limiar do Século XXI. ISBN 859003822X. rrr Edições, 1997.
- Beni, G. e Wang, J. "Swarm intelligence in cellular robotic systems," Proc. and Biological Systems, Il Ciocco Tuscany, Italy, June, 1989
- Cockshott, A. R. e Hartman, B.E. "Improving the fermentation medium for Echinocandin B production part II: Particle swarm optimization". Process Biochemistry, Volume 36, Number 7, February 2001, pp. 661-669(9), Elsevier Science, 2001.

- Costa Jr., E.F., Lage, P.L.C., Biscaia Jr., E.C. On the numerical solution and optimization of styrene polymerization in tubular reactors. *Computers and Chemical Engineering* 27, 2591-1604, 2003.
- Darwin, C. *On the Origin of Species by Means of Natural Selection, or, the Preservation of Favoured Races in the Struggle for Life*. London: J. Murray, 1859
- Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
- Dawkins, R., *The Selfish Gene*, Clarendon Press, Oxford, 1976.
- De Jong, K. A., "An Analysis of the Behavior of a Class of Genetic adaptive Systems", Doctoral dissertation, University of Michigan, 1975.
- Dorigo M., Gambardella L. M. Ant Colonies for the Traveling Salesman Problem. *Biosystems*, v 43, pp. 73-81, 1997.
- Dumitrescu, D., Lazzerini, B., Jain, L. C. and Dumitrescu, A., *Evolutionary Computation*. CRC Press International Series on Computational Intelligence, 2000.
- Eberhart, R. C. e R. W Dobbins. *Neural Network PC Tools: A Practical Guide*. Academic Press, San Diego, CA, 1990.
- Feller, W., *An Introduction to Probability Theory and its Applications* (2nd ed.), Vol. 1, Wiley, 1966.
- Futuyma, D.J. 1986. *Evolutionary Biology*. Sunderland, Mass: Sinauer Associates, Inc.
- Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison- Wesley 1989.
- Goldberg, D., Deb, K. e Clark, J., "Genetic Algorithms, Noise, and the Sizing of Populations". *Complex Systems*, Vol. 6, pp 333-362, 1992.
- Goldberg, D. E. e Rudnick, M., "Genetic Algorithms and the Variance of Fitness", in *Complex Systems*, Vol 5, pp 265-278, 1991.

- Goldberg, D. E., Sastry, K. e Latoza, T., "On the Supply of Building Blocks", in Proceedings of GECCO-2002, pp 333-362, 2002.
- Grefenstette, J. J., "Optimization of Control Parameters for Genetic Algorithms", in IEEE-SMC-16, pp122-128, 1986.
- Harik, G., Cantu-Paz, E., Goldberg, D. e Miller, B., "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations" in Proceedings of the IEEE International Conference on Evolutionary Computation, pp 7-12, 1997.
- Heppner, F. e U. Grenander. A stochastic nonlinear model for coordinated bird flocks. In S. Krasner, Ed., The Ubiquity of Chaos. AAAS Publications, Washington, DC, 1990
- Holland, J. H., Outline for a logical theory of adaptive systems, J. ACM, 3, 297-314, 1962.
- Holland, J., "Genetic Algorithms and the Optimal Allocations of Trails" in SIAM Journal of Computing, Vol 2(2, pp 88-105), 1973.
- Holland, J. H. Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, MI, 1975.
- Kennedy, J. e Eberhart, R. C. Swarm Intelligence. Morgan Kaufmann Publishers, 2001.
- Kennedy, J. e Eberhart, R. C. Particle Swarm Optimization. In Proceedings of the IEEE International Conference in Neural Networks, pp 1942-1948, November, 1995.
- Koza, J. R., Hierarchical genetic algorithms operating on populations of computer programs. Proceedings of the 11th International Joint Conference on Artificial Intelligence, N. S. Sridharan (Ed.), Morgan Kaufmann, San Mateo, CA, 768-774, 1989.
- Koza, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. ISBN 0262111705. MIT Press, 1992.

- Merz, P e Freisleben, B, Genetic Algorithms for Binary Quadratic Programming. Proceedings of the Genetic and Evolutionary Computation Conference, 1999.
- Michalewicz, Z., Genetic Algorithms + Data Structures=Evolution Programs, Third Revised and Extended Edition, Springer-Verlag Berlin Heidelberg, New York, 1992.
- Miller, B. L., "Noise, Sampling, and efficient genetic algorithms", Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1997.
- Millonas, M. M. Swarms, phase transitions, and collective intelligence. In C. G. Langton, Ed., Artificial Life III. Addison Wesley, Reading, MA, 1994.
- Mitchell, M., An introduction to genetic algorithms. Cambridge: MIT Press. 1997. 207 p.
- Moscato, P., Memetic Algorithms: A Short Introduction, D. Corne, M. Dorigo, F. Glover (eds.), New Ideas in Optimization, pp. 219-234, McGraw Hill, London, 1999.
- Moscato, P e Norman, M. G., "A "Memetic" Approach for the Travelling Salesman Problem - Implementation of a Computational Ecology for Combinatorial Optimisation on Message-Passing System". In Proceedings of the International Conference on Parallel Computing and Transputer Applications. IOS Press, Amsterdam, 1992.
- Moscato, P e Cotta, C. A Gentle Introduction to Memetic Algorithms, F. Glover, G. Kochenberger (eds.), Handbook of Metaheuristics, pp. 105-144, Kluwer Academic Publishers, Boston MA, 2003.
- Ourique, C.O., Biscaia Jr., E.C., Pinto, J.C. The use of particle swarm optimization for dynamical analysis in chemical processes. Computers and Chemical Engineering 26, 1783-1793, 2002.
- Papadimitriou, C. H. e Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall Inc., 1982.

- Parsopoulos, K.E., Vrahatis, M.N. Particle swarm optimization method in multiobjective problems. Proceedings ACM Symposium on Applied Computing (SAC), 603- 607, 2002.
- Pomeroy, P., "An Introduction to Particle Swarm Optimization", <http://www.adaptiveview.com>, [15 Setembro de 2003].
- Radcliffe, N.J. e Surry, P.D., Formal Memetic Algorithms. Lecture Notes in Computer Science, 865, Evolutionary Computing, Springer Verlag, Berlin, 1-16, 1994.
- Reeves, C. R., "Using Genetic Algorithms with Small Populations" in Proceedings of the Fifth International Conference On Genetic Algorithms, pp 92-99, 1993.
- Reynolds, C. W. Flocks, herds and schools: a distributed behavioral model. Computer Graphics, 21 (4): 25-34, 1987.
- Tomassini, M., "A Survey of Genetic Algorithm", II Anal Reviews of Computational Physics, World Scientific, 1998.
- Venter, G. and Sobieszczanski-Sobieski, J., "Particle Swarm Optimization", In Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO, Vol. AIAA-2002-1235, April 22-25 2002.
- Wilson, E.O. Sociobiology: The new synthesis. Cambridge, MA: Belknap Press, 1975

Capítulo 3

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS

Introdução

Este capítulo dedica-se à síntese de circuitos combinatórios com portas lógicas através de algoritmos genéticos. O capítulo está organizado em cinco secções. A primeira secção contém um breve resumo sobre o estado da arte da electrónica evolutiva. As secções 2 e 3 apresentam, respectivamente, a formulação do problema e a implementação dos circuitos. A secção 2 serve também de base para os capítulos seguintes uma vez que descreve o problema que se pretende resolver. De seguida, nas secções 4 e 5, são abordados dois temas, o problema de escala na síntese de circuitos lógicos e a questão da parametrização do tamanho da população no AG com repercussão directa no tempo de processamento do algoritmo. As conclusões são apresentadas na secção 6.

1. Electrónica evolutiva

Na última década os Algoritmos Evolutivos (AEs) começaram a ser aplicados no desenho e síntese de circuitos electrónicos, o que levou ao aparecimento de uma nova área de investigação designada por Electrónica Evolutiva (EE) ou *Hardware Evolutivo* (EH - *Evolvable Hardware*) (Zebulum *et al.*, 2001). A EE considera o conceito da síntese automática de sistemas electrónicos em substituição dos modelos, abstrações e técnicas concebidas pelo ser humano. A EE emprega algoritmos de pesquisa para o desenvolvimento de projectos electrónicos (Thompson e Layzell, 1999).

Louis e Rawlins (Louis e Rawlins, 1991) aplicaram os Algoritmos Genéticos (AGs) ao problema da síntese de circuitos combinatórios. Combinaram sistemas baseados em conhecimento com AGs e definiram um novo operador genético designado por cruzamento mascarado. Este esquema permitiu a geração de outros tipos de descendentes que não poderiam ser obtidos através dos operadores de cruzamento clássicos.

John Koza (Koza, 1992) adoptou a Programação Genética (PG) no projecto de circuitos combinatórios, tendo como objectivo a síntese de circuitos funcionais compostos pelas portas lógicas E, OU e NÃO. Na sequência deste trabalho, Coello, Christiansen e Aguirre (Coello *et al.*, 1996) apresentaram um programa de computador capaz de gerar automaticamente circuitos de alta qualidade. Usaram cinco tipos de portas lógicas, E, NÃO, OU, XOR e WIRE¹), com o objectivo de obterem projectos de circuitos funcionais, minimizando o uso de portas lógicas (*i. e.* com o número máximo de WIRES).

¹ Ligação directa, não utilizando portas lógicas

Miller, Thompson e Fogarty (Miller *et al.*, 1997) aplicaram os Algoritmos Evolutivos (AEs) à síntese de circuitos aritméticos. Esta técnica baseia-se na evolução da funcionalidade e da conectividade de uma matriz de células lógicas, com o modelo de recursos disponível no dispositivo FPGA² Xilinx 6216.

Kalganova, Miller e Lipnitskaya (Kalganova *et al.*, 1998) propuseram uma técnica nova na síntese de circuitos de múltiplas entradas. A EH é facilmente adaptada aos vários tipos de portas lógicas de múltiplas entradas, associada a operações de diferentes tipos de álgebra, e pode incluir outro tipo de expressões lógicas. Esta abordagem é uma extensão do método de EH proposto em (Miller *et al.*, 1997).

Com o objectivo de resolver sistemas complexos, Torresen (Torresen, 1998) propôs um método da evolução com a capacidade de se adaptar à medida que a complexidade aumenta. A ideia é a de os circuitos irem evoluindo gradualmente, *i. e.* num método do tipo “dividir para reinar”. A evolução começa individualmente num largo número de células simples e as funções obtidas irão ser os blocos básicos das células seguintes e mais complexas.

Mais recentemente, Hollingworth, Smith and Tyrrell (Hollingworth *et al.*, 2000) descreveram a primeira tentativa de evolução dos circuitos em dispositivos da família Virtex³. Implementaram um somador de dois *bits*, onde

² *Field Programmable Gate Array*. É um dispositivo semiconductor largamente utilizado no processamento de informações digitais. Foi criado pela *Xilinx Inc.* e teve o seu lançamento no ano de 1985 como um dispositivo que poderia ser programado de acordo com as aplicações do utilizador (programador).

³ As FPGAs da família Virtex da Xilinx são baseadas numa arquitectura dividida nas seguintes partes: uma matriz de blocos lógicos configuráveis, designados por CLBs (*Configurable Logic Blocks*); um conjunto de blocos de entrada/saída (E/S), designados por IOBs (*Input/Output Blocks*), que serve de *interface* entre os CLBs, e os pinos do componente; um conjunto de blocos de memória RAM (*Random Access Memory*) dedicados de 4096 *bits* cada; um conjunto de recursos de interligação local, designado *VersaBlock*; um conjunto de recursos genéricos de interligação, designado GPR (*General Purpose Routing*); um conjunto de recursos de interligação periféricos, designados *VersaRing*; quatro DLLs (*Delay-Locked Loops*) dedicadas ao controlo do relógio a aplicar à lógica configurada; uma memória estática de configuração.

as entradas do circuito são os dois números de 2 *bits* e o resultado esperado a soma destes dois números.

Um dos maiores problemas na síntese evolutiva de circuitos electrónicos é o problema de escala. Este problema refere-se ao crescimento acentuado do número de portas lógicas usadas no circuito alvo com o aumento do número de entradas da função lógica que se pretende implementar. Isto resulta num aumento enorme do espaço de pesquisa dificultando a exploração, mesmo utilizando técnicas evolutivas. Outro obstáculo associado é o tempo necessário para o cálculo da função de aptidão dos circuitos (Vassilev e Miller, 2000 e Gordon e Bentley, 2002). Um método possível para solucionar este problema é a utilização de blocos básicos em substituição das portas lógicas simples (Murakawa *et. al.*, 1996). No entanto, esta técnica dá origem a outra dificuldade que é a de saber como definir os blocos básicos necessários e suficientes ao desenho do circuito.

2. Formulação do problema

Os circuitos combinatórios são especificados pelas tabelas de verdade. Estes circuitos podem ter múltiplas entradas e múltiplas saídas e o objectivo é o de implementar circuitos funcionais com a menor complexidade possível. Para esse propósito definiram-se vários conjuntos de portas lógicas sendo os circuitos combinatórios gerados com os componentes de cada um desses conjuntos (Reis e Machado, 2003a, 2003b).

A tabela 3.1 apresenta quatro dos conjuntos que forem especificados neste trabalho. O conjunto 2 (*Gset 2*) é o conjunto de portas lógicas mais simples, os

conjuntos 3 e 4 (*Gsets* 3 e 4) são de média complexidade e por último, o conjunto *Gset* 6 é o mais complexo.

Para cada um dos conjuntos de portas lógicas o AG procura o espaço das soluções com base na teoria da sobrevivência dos mais aptos. Genericamente, os melhores indivíduos de uma população têm tendência a reproduzirem-se e a sobreviver, melhorando desta forma as gerações seguintes. No entanto, indivíduos menos aptos podem também sobreviver e reproduzirem-se (Goldberg, 1989).

Os sistemas EH desenvolvem cromossomas que codificam a descrição funcional dos circuitos electrónicos. Tal como na maioria das aplicações dos AGs, o circuito resultante é o fenótipo dado que contém várias células lógicas pequenas ou genótipos. A terminologia adoptada reflecte a similaridade conceptual entre a EH, a evolução natural e a genética (Hounsell e Arslan, 2000).

Tabela 3.1 - Conjuntos de Portas Lógicas: *Gsets* 2, 3, 4 e 6

Conjunto	Portas Lógicas
<i>Gset</i> 2	{AND,XOR,WIRE}
<i>Gset</i> 3	{AND,OR,XOR,WIRE}
<i>Gset</i> 4	{AND,OR,XOR,NOT,WIRE}
<i>Gset</i> 6	{AND,OR,XOR,NOT,NAND,NOR,WIRE}

2.1. Codificação dos circuitos

No esquema do AG, os circuitos são codificados como uma matriz rectangular ($linha \times coluna = l \times c$) de células lógicas, tal como está representado na figura 3.1.

Cada célula é representada por três genes: $\langle entrada1 \rangle \langle entrada2 \rangle \langle tipo\ de\ porta \rangle$, onde $entrada1$ e $entrada2$ são uma das entradas do circuito, se são entradas da primeira coluna da matriz, ou uma das saídas das colunas anteriores, se estão noutra coluna da matriz. O gene $tipo\ de\ porta$ é um dos elementos do conjunto de portas lógicas adoptado. O cromossoma é constituído pelos conjuntos de três genes, de acordo com o tamanho da matriz. Por exemplo, o cromossoma que representa uma matriz 3×3 está ilustrado na figura 3.2.

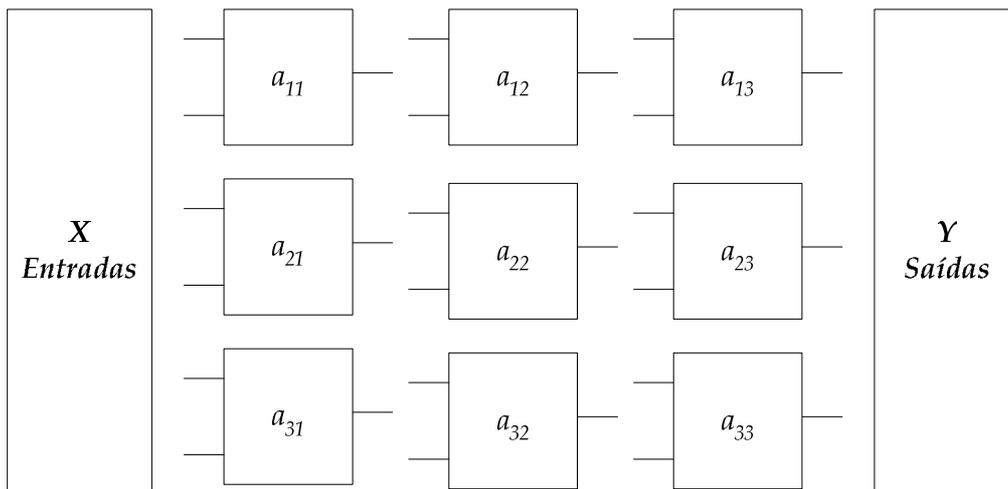


Figura 3. 1- Matriz 3×3 que representa um circuito.

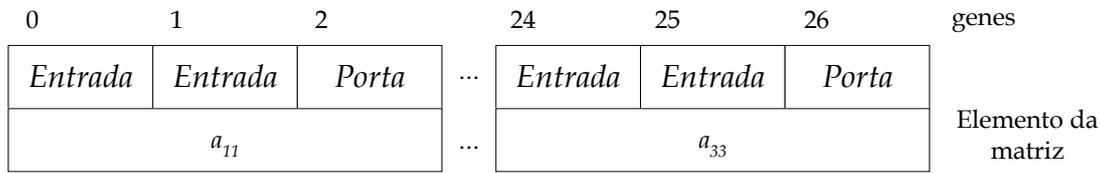


Figura 3.2 - Cromossoma da matriz da figura 3.1

2.2. Operadores genéticos

A população inicial de circuitos (vectores) é gerada aleatoriamente, sendo a pesquisa efectuada nesta população (Holland, 1975). Os três operadores distintos que são implementados no AG são a reprodução, o cruzamento e a mutação, os quais são descritos a seguir.

No que diz respeito à reprodução, as gerações sucessivas de novos vectores são reproduzidos com base na função de aptidão. A selecção é feita pelo método de torneio, onde são seleccionados três vectores (indivíduos) que vão competir entre si, vencendo o indivíduo com melhor aptidão.

Em relação ao operador de cruzamento, os indivíduos vão sendo agrupados aos pares de uma forma aleatória e, em seguida, é efectuada o cruzamento num ponto. Este ponto de cruzamento só é permitido entre células de forma a manter a integridade do cromossoma.

O operador de mutação altera as características totais de uma dada célula da matriz. Desta forma modifica o tipo de porta lógica e as duas entradas, o que significa que pode surgir uma célula completamente nova na matriz.

Além dos operadores anteriores descritos é também implementada a estratégia elitista. Conseqüentemente as melhores soluções são sempre mantidas para a próxima geração.

Para executar o AG é necessário definir o número de indivíduos para se proceder à criação da população inicial P (Goldberg, 1985). Esta população mantém o mesmo tamanho ao longo das gerações.

A taxa de cruzamento TC representa a percentagem da população P que se vai reproduzir em cada geração. Similarmente, TM (taxa de mutação) é a percentagem da população P que vai sofrer uma mutação em cada geração. Normalmente, para que a população evolua, TC é elevado (na ordem dos 80% a 95%) e, para prevenir que a população se diversifique em demasia TM é baixo (na ordem dos 1% a 5%). Neste caso concreto, de evolução de circuitos digitais, adoptou-se $P = 3000$ indivíduos, $TC = 95\%$ e $TM = 5\%$.

2.3. Função de aptidão

O cálculo da função de aptidão F é dividido em duas partes, f_1 e f_2 que medem respectivamente a funcionalidade e a simplicidade. Em primeiro lugar, compara-se o resultado produzido (*i. e.*, o circuito gerado) pelo AG com os valores esperados, de acordo com a tabela de verdade correndo-a *bit a bit* (f_1). Assim que o circuito seja funcional, isto é quando for atingido f_{10} , o AG começa a tentar gerar circuitos com o menor número de portas lógicas. O índice f_2 , que avalia a simplicidade, é incrementado de 1 por cada porta lógica do tipo *wire* existente no circuito, de acordo com:

$$f_{10} = 2^{n_i} \times n_o \quad (3.1)$$

$$f_1 = f_1 + 1 \text{ se } \{\text{bit } i \text{ de } \mathbf{Y}\} = \{\text{bit } i \text{ de } \mathbf{Y}_R\}, i = 1, \dots, f_{10} \quad (3.2)$$

$$f_2 = f_2 + 1 \text{ se tipo de porta} = \text{wire} \quad (3.3)$$

$$F = \begin{cases} f_1, & F < f_{10} \\ f_1 + f_2, & F \geq f_{10} \end{cases} \quad (3.4)$$

onde n_i e n_o representam o número de entradas e de saídas do circuito, respectivamente, \mathbf{Y} é a saída produzida pelo AG e \mathbf{Y}_R é o valor de saída pretendido de acordo com a tabela de verdade previamente definida.

3. Implementação dos circuitos

Esta secção apresenta a implementação de quatro circuitos lógicos combinatórios distintos, a saber, o multiplexador 2 para 1, o somador completo de um *bit*, o circuito de teste de paridade de quatro *bits* e o multiplicador de dois *bits*.

O primeiro caso, o multiplexador 2 para 1 (M21) apresenta uma tabela de verdade de três entradas $\{S_0, I_1, I_0\}$ e uma saída $\{O\}$ o que origina uma matriz de dimensões $l \times c = 3 \times 3$ e o comprimento do vector que representa cada um dos circuitos (*i. e.*, o tamanho do cromossoma) é $TC = 27$.

Em virtude da natureza dos AGs ser estocástica, para cada conjunto de portas lógicas desenvolveram-se várias simulações. O conjunto de portas lógicas que apresenta melhor desempenho é aquele que atinge a solução no menor número de gerações N com a função de aptidão F mais elevada.

Na figura 3.3 pode-se ver a função de aptidão F versus o número de gerações N para alcançar a solução para o circuito M21. Como este circuito tem $n_i = 3$ e $n_o = 1$ resulta $f_{10} = 8$ e $F \geq 12$.

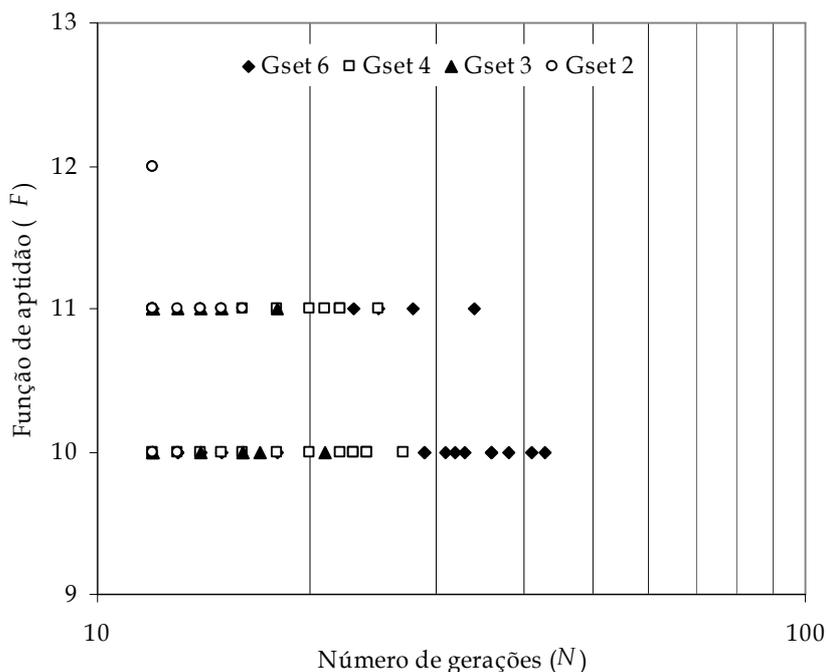


Figura 3.3 - Função de aptidão F versus número de gerações N para obter a solução para o circuito M21, utilizando o AG.

Verifica-se que, neste caso, o melhor conjunto de portas lógicas é o *Gset 2*, dado que chega à solução com a menor média do número de gerações $\mu(N)$ do AG.

Os melhores circuitos resultantes apresentam uma função de aptidão final $F = 12$ tal como o que se exhibe na figura 3.4.

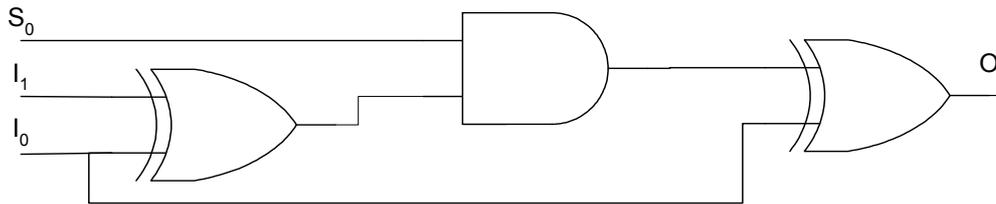


Figura 3. 4 - Circuito Multiplexador 2 para 1 gerado pelo AG.

O segundo caso, o somador completo de um *bit* (SOM1), apresenta uma tabela de verdade de três entradas ($\{A, B, C_{in}\}$ e duas saídas $\{S, C_{out}\}$ o que origina uma matriz de dimensões $l \times c = 3 \times 3$. O comprimento do vector que representa cada um dos circuitos (*i. e.*, o tamanho do cromossoma) é $TC = 27$.

A figura 3.5 ilustra a função de aptidão F versus o número de gerações N necessário para alcançar a solução.

O conjunto de portas lógicas que apresenta melhor desempenho é aquele que atinge a solução no menor número de gerações N com a função de aptidão F mais elevada. Dado que o circuito SOM1 tem $n_i = 3$ e $n_o = 2$, tem-se $f_{10} = 16$ e $F \geq 20$.

Analisando os resultados apresentados na figura 3.5, constata-se que os dois melhores conjuntos de portas lógicas são os conjuntos *Gset 3* e *Gset 2*, pois estes resultam numa menor média do número de gerações para obter a solução $\mu(N)$ acompanhada da melhor média final da função de aptidão $\mu(F)$. Os melhores circuitos SOM1 obtidos têm uma função de aptidão final $F = 19$, sendo o esquemático ilustrado na figura 3.6 um exemplo de um destes circuitos gerado com o *Gset 2*.

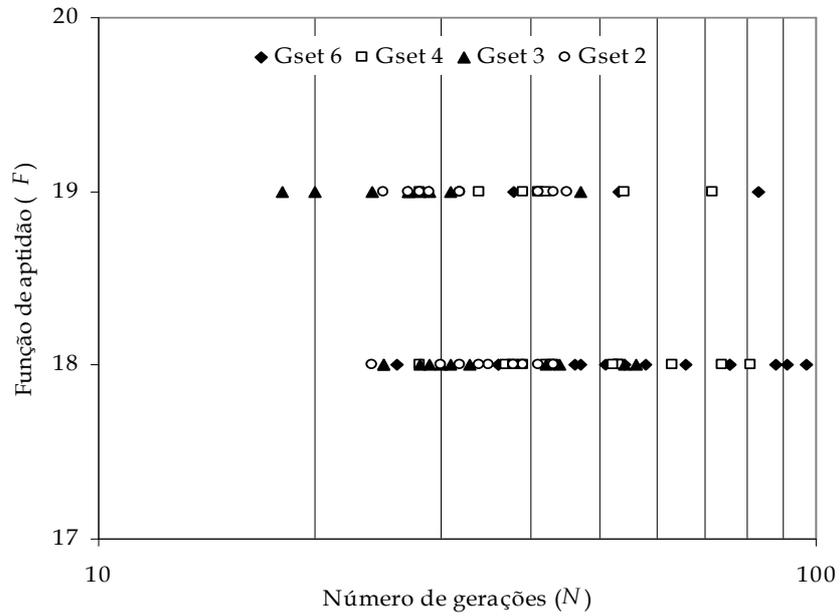


Figura 3.5 - Função de aptidão F versus número de gerações N para obter a solução para o circuito SOM1, utilizando o AG.

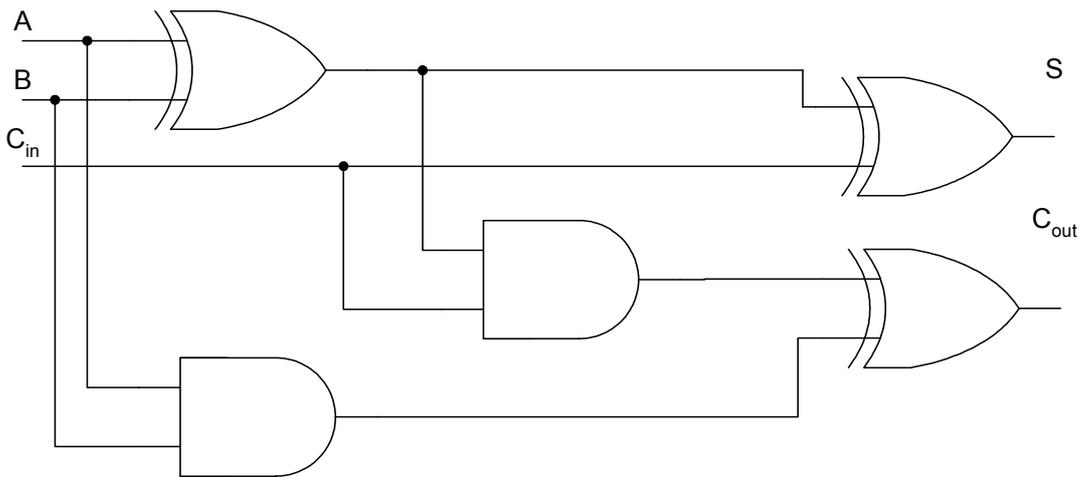


Figura 3.6 - Circuito Somador de um bit gerado pelo AG.

O terceiro caso estudado é o circuito de teste de paridade (par⁴) de quatro bits (TP4). Este circuito exhibe uma tabela de verdade de quatro entradas $\{A_3, A_2, A_1, A_0\}$ e uma saída $\{P\}$ dando origem a uma matriz de dimensões $l \times c = 4 \times 4$, com comprimento do vector $TC = 48$.

A figura 3.7 mostra os resultados da simulação em termos do número de gerações necessário para obter a solução N e a função de aptidão final F , para cada um dos conjuntos de portas lógicas definido.

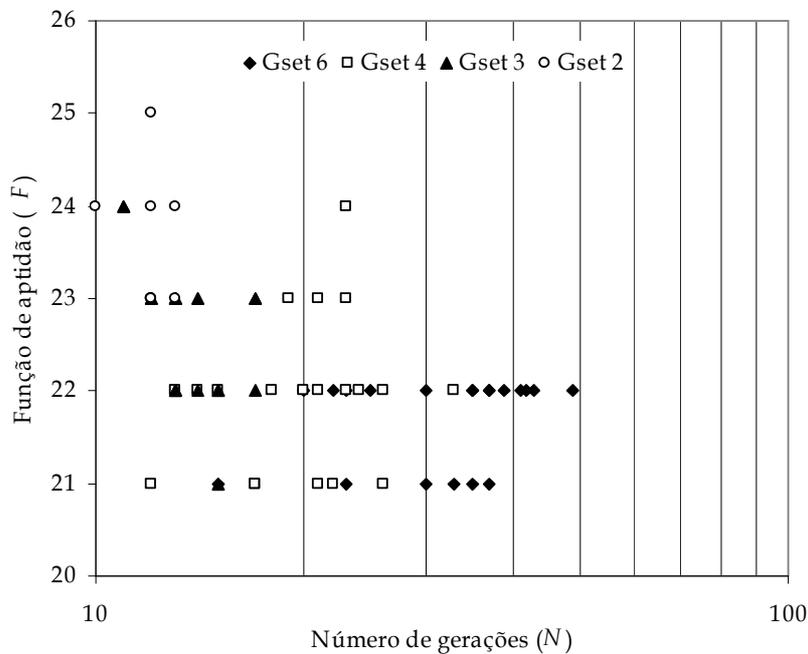


Figura 3.7 - Função de aptidão F versus número de gerações N para obter a solução para o circuito TP4, utilizando o AG.

⁴ Durante a transmissão de uma informação podem ocorrer ruídos que alteram bits de 0 para 1, ou vice-versa. Para detectar, ou até corrigi-los automaticamente, são utilizados códigos detectores, ou correctores, de erros. O código detector de erro mais popular é o código com bit de paridade. O bit de paridade é um bit adicional a uma mensagem binária de forma que o número total de 1's seja par (paridade par) ou ímpar (paridade ímpar), como exemplifica a seguinte tabela:

Mensagem de 4 bits	Bit de paridade par	Bit de paridade ímpar
0000	0	1
0110	0	1
1000	1	0
1111	0	1

Agora tem-se $n_i = 4$ e $n_o = 1$, resultando em $f_{10} = 16$ e $F \geq 24$. Os melhores resultados foram conseguidos com o *Gset 2* e o esquemático de um dos circuitos com mais alta função de aptidão ($F = 25$) é apresentado na figura 3.8.

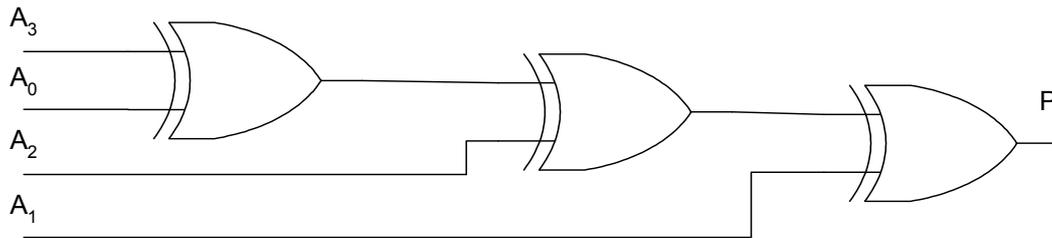


Figura 3. 8 - Circuito Teste de Paridade de quatro *bits* gerado pelo AG.

O quarto caso é o circuito multiplicador de dois *bits* (MUL2) com tabela de verdade de quatro entradas $\{A_1, A_0, B_1, B_0\}$ e quatro saídas $\{C_3, C_2, C_1, C_0\}$. A matriz correspondente é de dimensão $l \times c = 4 \times 4$ e o cromossoma de tamanho $TC = 48$.

Para o circuito MUL2 tem-se $n_i = 4$ e $n_o = 4$, dando $f_{10} = 64$ e $F \geq 72$. Mais uma vez conclui-se que o *Gset 2* é o conjunto com melhor desempenho na síntese destes circuitos combinatórios (figura 3.9). Na figura 3.10 pode-se ver o esquemático do circuito obtido com a função de aptidão mais elevada, *i. e.* $F = 72$.

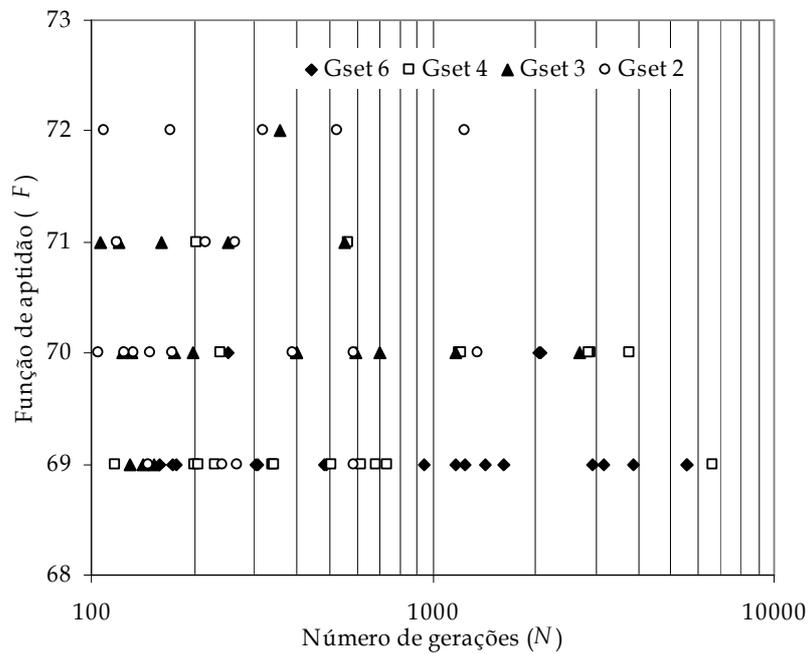


Figura 3.9 - Função de aptidão F versus número de gerações N para obter a solução para o MUL2, utilizando o AG.

A tabela 3.2 mostra a média do número de gerações para alcançar a solução $\mu(N)$ e a média da função de aptidão $\mu(F)$ após a realização de vinte experiências para cada conjunto de portas lógicas.

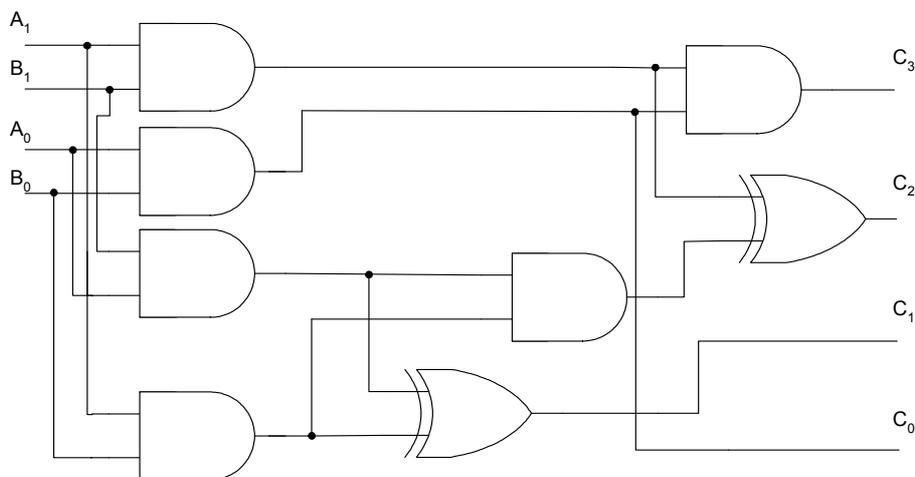


Figura 3.10 - Circuito Multiplicador de dois bits gerado pelo AG.

Tabela 3. 2- Resultados para os circuitos M21, SOM1, TP4 e MUL, utilizando o AG

Conjunto	Circuito							
	M21		SOM1		TP4		MUL2	
	$\mu(N)$	$\mu(F)$	$\mu(N)$	$\mu(F)$	$\mu(N)$	$\mu(F)$	$\mu(N)$	$\mu(F)$
<i>Gset 6</i>	27,15	10,25	72,45	18,15	32,55	21,70	1699,00	69,15
<i>Gset 4</i>	19,75	10,35	53,65	18,35	20,40	21,95	1183,05	69,50
<i>Gset 3</i>	13,55	10,5	32,40	18,45	13,754	22,65	432,40	70,25
<i>Gset 2</i>	12,05	11,5	34,86	18,57	7,95	23,95	362,35	70,45

É possível constatar a superioridade dos *Gsets* 2 e 3, quer pela média do número de gerações para obter a solução quer pela média da função de aptidão obtida.

As figuras 3.11 e 3.12 ilustram estes resultados.

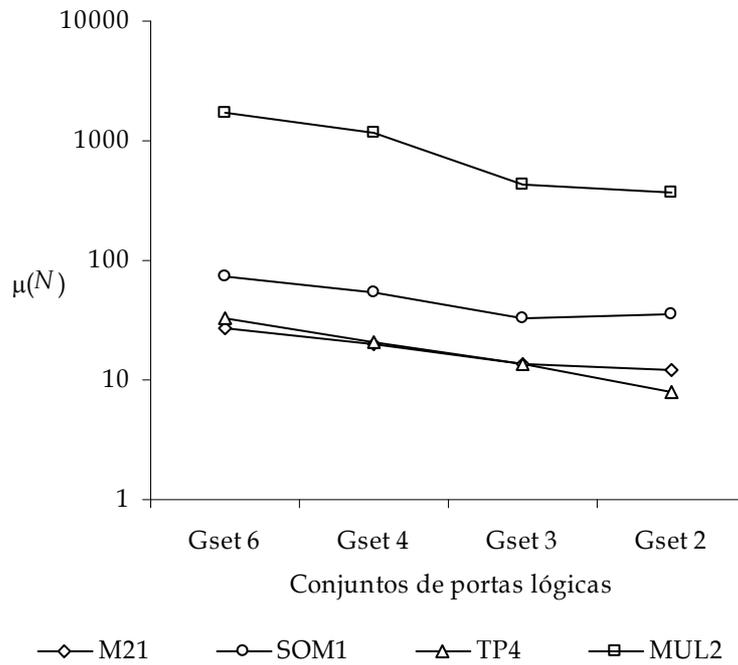


Figura 3.11 - Média do número de gerações para alcançar a solução $\mu(N)$ para os conjuntos de portas lógicas em avaliação, utilizando o AG.

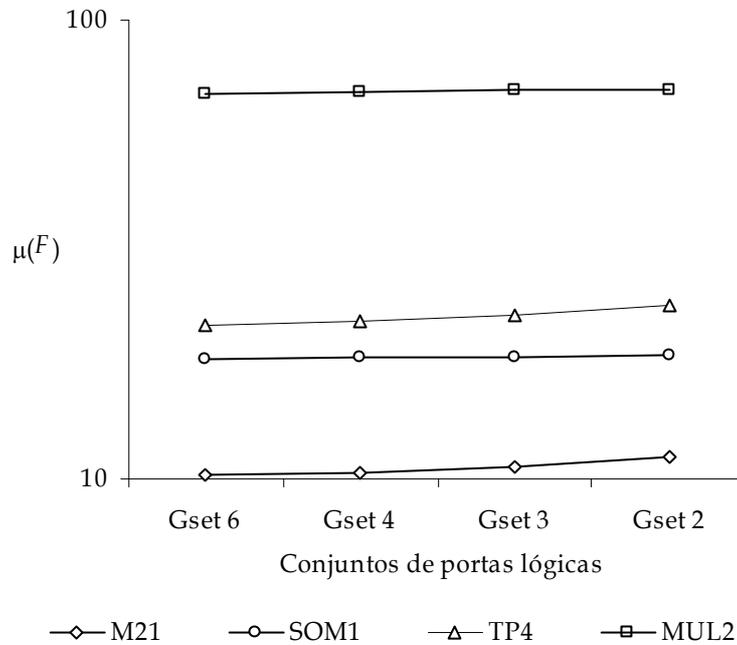


Figura 3.12 - Média da função de aptidão dos circuitos obtidos $\mu(F)$ para os conjuntos de portas lógicas em avaliação, utilizando o AG.

A figura 3.13 ilustra a média da função de aptidão $\mu(F)$ versus a média do número de gerações $\mu(N)$ para obter a solução, para todos os conjuntos de portas lógicas (i. e. Gsets 2, 3, 4 e 6) e para os circuitos M21, SOM1, TP4 e MUL2.

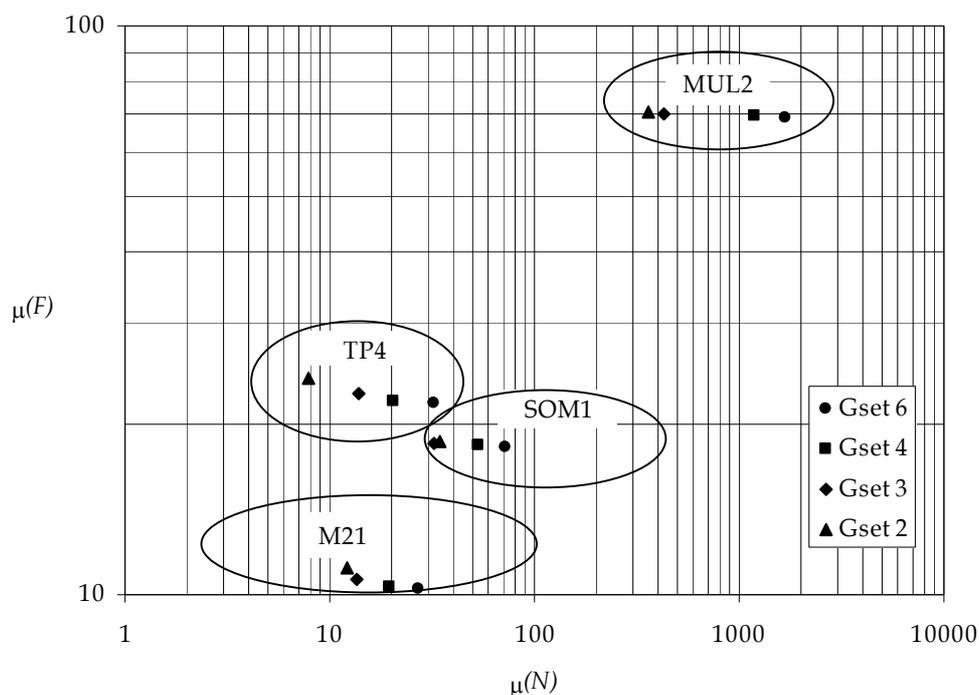


Figura 3. 13 – Média da função de aptidão $\mu(F)$ versus a média do número de gerações para obter a solução $\mu(N)$, utilizando o AG.

Comparando os quatro casos estudados com base na média do número de gerações necessárias para a obtenção das soluções $\mu(N)$ e na média resultante da função de aptidão $\mu(F)$ conclui-se que, independentemente da complexidade do circuito, os melhores resultados surgem com os conjuntos reduzidos de portas lógicas. Esta conclusão apresenta similaridades com o dilema RISC⁵ versus CISC⁶ discutido no âmbito dos processadores.

⁵ *Reduced Instruction Set Computer (RISC)* ou Computador com um Conjunto Reduzido de Instruções, é uma linha de arquitectura de computadores que favorece um conjunto simples e pequeno de instruções que levam aproximadamente a mesma quantidade de tempo para serem executadas.

⁶ *Complex Instruction Set Computer (CISC)* ou Computador com um Conjunto Complexo de Instruções, é um processador capaz de executar centenas de instruções complexas diferentes, sendo assim extremamente versátil.

4. Problema de escala

Outro assunto que emerge com o aumento do número de entradas e de saídas nos circuitos lógicos é o problema de escala. Como as tabelas de verdade crescem exponencialmente com o número de entradas dos circuitos, a carga computacional do AG para chegar à solução aumenta dramaticamente (Reis, Machado e Cunha, 2004b).

As figuras 3.14 a 3.19 mostram a evolução dos índices $\mu(N)$, $\sigma(N)$ e $\mu(F)$ para os circuitos combinatórios de teste de paridade e somador completo, à medida que o número de *bits* aumenta, sendo $\sigma(N)$ o desvio padrão do número de gerações para obter a solução.

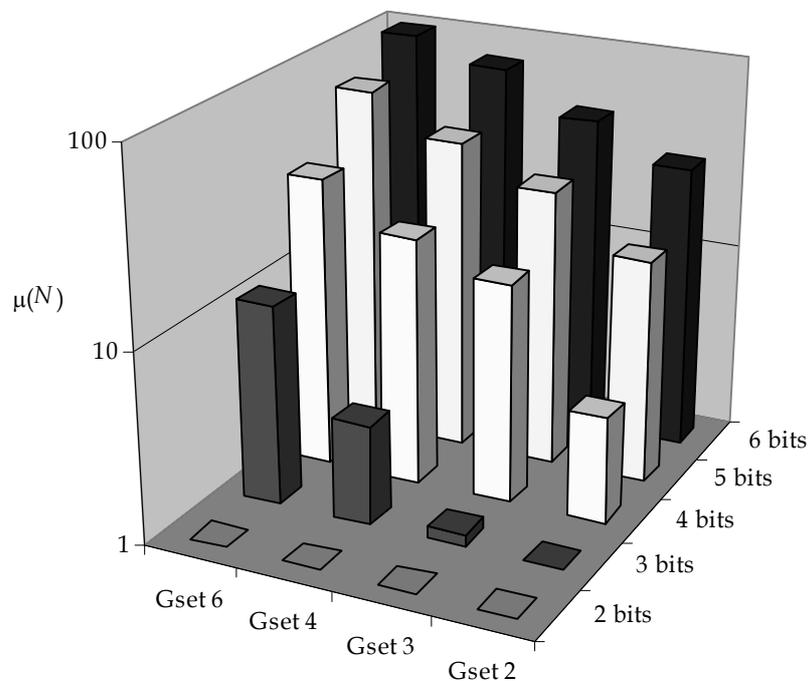


Figura 3.14 - Média do número de gerações para obter a solução $\mu(N)$ para os circuitos de teste de paridade de 2, 3, 4, 5 e 6 bits para os conjuntos de portas lógicas em avaliação.

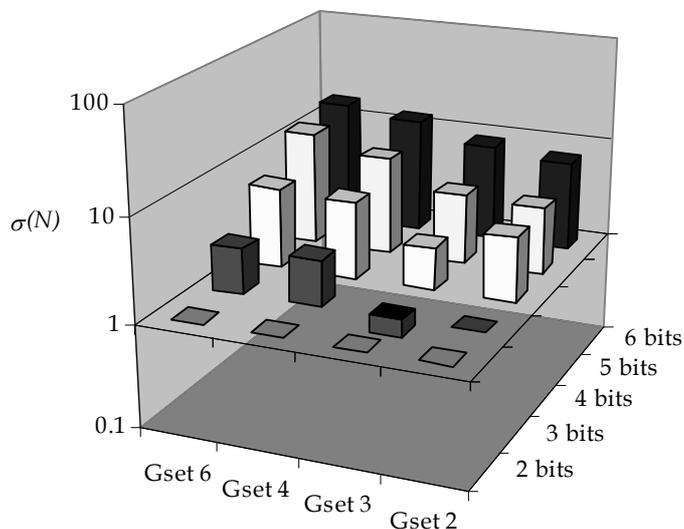


Figura 3.15 - Desvio padrão do número de gerações para obter a solução $\sigma(N)$ para os circuitos de teste de paridade de 2, 3, 4, 5 e 6 bits para os conjuntos de portas lógicas em avaliação.

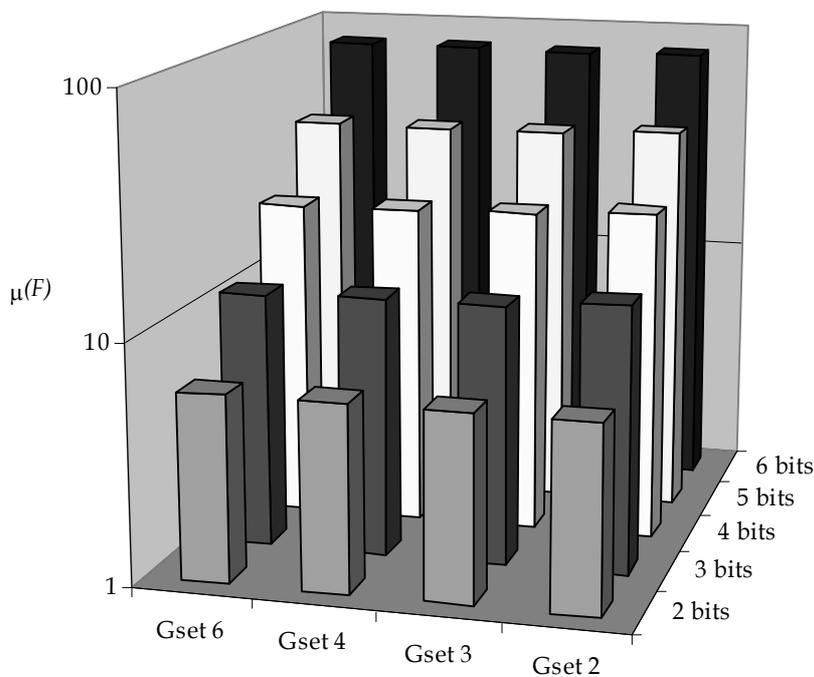


Figura 3.16 - Média da função de aptidão final $\mu(F)$ para os circuitos de teste de paridade de 2, 3, 4, 5 e 6 bits para os conjuntos de portas lógicas em avaliação.

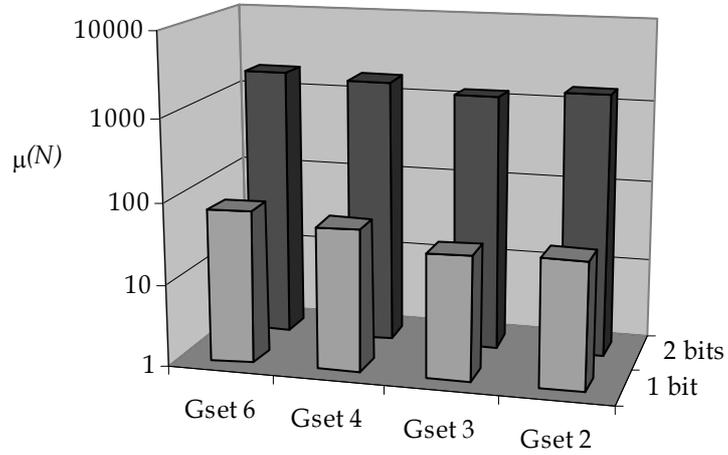


Figura 3. 17 - Média do número de gerações para obter a solução $\mu(N)$ para os circuitos somadores completos de 1 e 2 bits para os conjuntos de portas lógicas em avaliação.

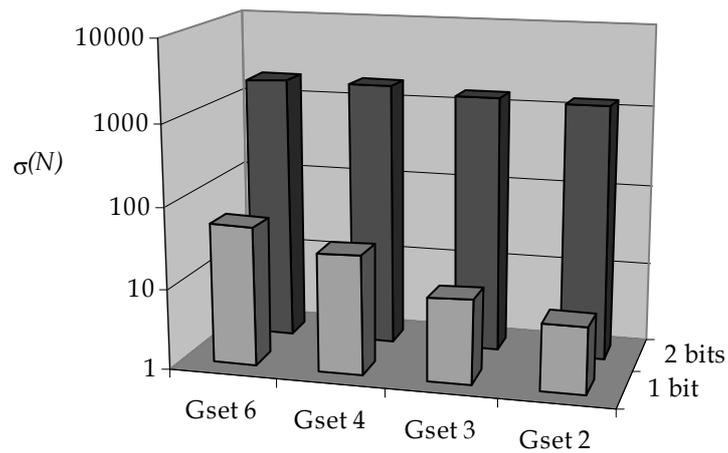


Figura 3. 18 - Desvio padrão do número de gerações para obter a solução $\sigma(N)$ para os circuitos somadores completos de 1 e 2 bits para os conjuntos de portas lógicas em avaliação.

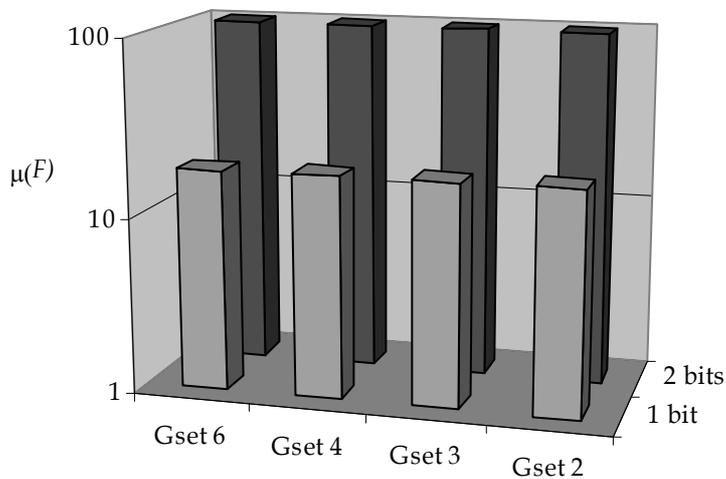


Figura 3.19 - Média da função de aptidão final $\mu(F)$ para os circuitos somadores completos de 1 e 2 bits para os conjuntos de portas lógicas em avaliação.

Pela análise dos gráficos das figuras 3.14 a 3.19 comprova-se o aumento significativo da dificuldade de convergência do AG à medida que o número de bits dos circuitos aumenta.

A figura 3.20 mostra a evolução de $\mu(N)$ e de $\mu(F)$, para os circuitos das famílias de teste de paridade e somador completo, com o aumento do número de bits dos circuitos. A família de circuitos de teste de paridade é de {2, 3, 4, 5 e 6} bit e a família do circuito somador completo é {1 e 2} bit.

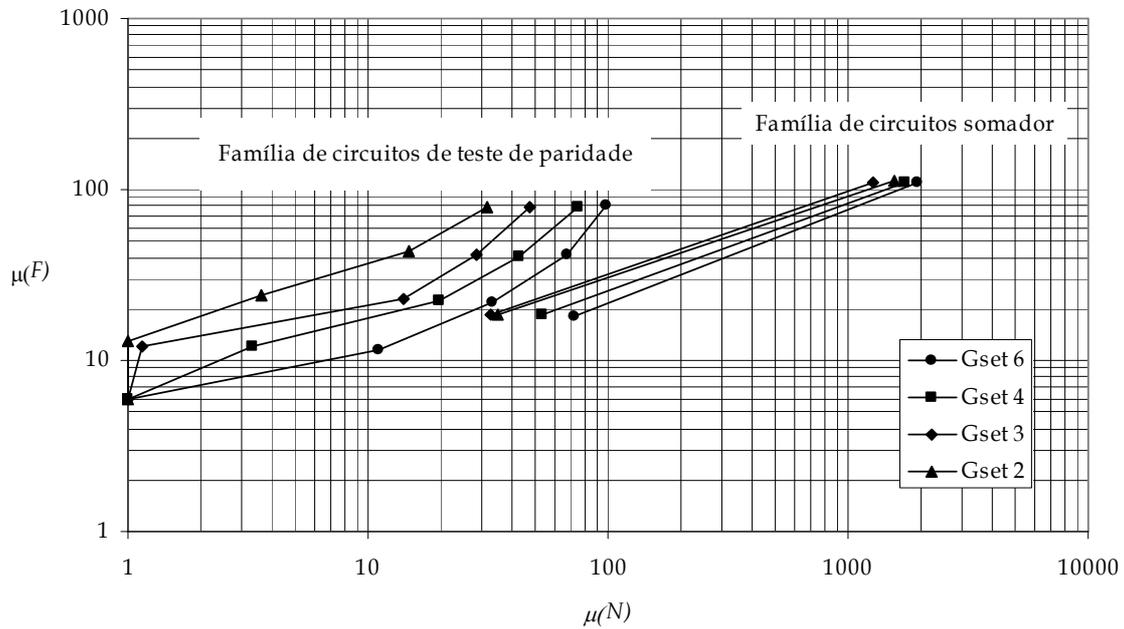


Figura 3. 20 - $\mu(F)$ versus $\mu(N)$ para as famílias dos circuitos de teste de paridade e somador completo, para os conjuntos de portas lógicas 2, 3, 4 e 6.

Analisando o gráfico da família de circuitos de teste de paridade da figura 3.20, verifica-se que, depois de decorrer o transitório inicial, tem-se uma lei exponencial dada aproximadamente por:

$$\mu(F) = ae^{b\mu(N)} \quad a, b \in \mathfrak{R} \quad (3.5)$$

Os coeficientes (a , b) obtidos para os quatro conjuntos de portas lógicas encontram-se na tabela 3.3. Em relação ao coeficiente a pode-se dizer que o conjunto de portas lógicas *Gset 2* apresenta o melhor valor, os conjuntos *Gset 3* e *Gset 4* apresentam valores de a muito idênticos e o conjunto *Gset 6* é o que apresenta pior desempenho. Quanto ao coeficiente b , é possível agrupar o conjunto *Gset 6* com o conjunto *Gset 4* e o conjunto *Gset 2* com o conjunto *Gset 3*.

Tabela 3.3 - Coeficientes a e b da equação 3.5

Conjunto	a	b
<i>Gset 6</i>	9.8	0.0214
<i>Gset 4</i>	12.3	0.0257
<i>Gset 3</i>	12.2	0.0408
<i>Gset 2</i>	21.2	0.0433

5. Tamanho da População e Tempo de Processamento

É conhecido que os AGs são amplamente utilizados como técnicas de pesquisa e de optimização em muitas áreas de aplicação. O sucesso dos AGs depende do correcto estabelecimento dos parâmetros envolvidos, como o tamanho da população, e na interacção entre os diferentes operadores, como sejam os operadores de cruzamento e mutação, os quais criam novos indivíduos operando com os indivíduos existentes (Khan, 2002).

Esta secção concentra-se no estudo do tamanho da população necessário para obter o tempo mínimo de processamento do AG na síntese de circuitos lógicos combinatórios.

5.1. Tamanho da população

O aumento da complexidade na optimização leva a uma necessidade de maiores tamanhos da população. Por esse motivo é importante entender o papel da população no domínio dos AGs.

Em 1992, Goldberg, Deb e Clark (Goldberg *et. al.*, 1992) propuseram modelos de definição de tamanhos de população para diferentes esquemas de selecção. O modelo é baseado na decisão correcta entre o melhor e o segundo melhor bloco de construção ou bloco construtor (BC) da mesma partição. Incorporaram também no modelo ruído vindo de outras partições. No entanto, consideraram que se fossem escolhidos na primeira geração BCs “maus” ou errados, então o AG não conseguiria recuperar desse erro.

Mais tarde, Harik, Cantu-Paz, Goldberg e Miller (Harik *et. al.*, 1997) aperfeiçoaram o modelo anterior com a integração de efeitos cumulativos de tomada de decisão ao longo das gerações sucessivas, ao invés de esta ocorrer só na primeira geração. Estes autores modelizaram a tomada de decisão, entre o melhor e o segundo melhor BC numa partição, ao qual chamaram “problema da ruína do jogador” (*gambler's ruin problem*). Este esquema baseia-se na suposição de que o processo de selecção consiste na selecção por torneio sem substituição.

Miller (Miller, 1997) expandiu este modelo para a previsão do tamanho da população na presença de ruído externo.

Neste encadeamento, observa-se que a maior parte do trabalho desenvolvido na área da definição do tamanho da população nos AGs se divide em dois grupos principais, a saber, a definição do tamanho da população com

base no fornecimento inicial de BCs e a definição do tamanho da população com base na tomada de decisão entre BCs que estão a competir (Khan, 2002). Ambas as versões são combinadas com o modelo da ruína do apostador.

5.2. Experiências desenvolvidas

Tal como já foi referido uma análise fiável de AGs requer um largo número de simulações para garantir com segurança que os efeitos estocásticos foram considerados apropriadamente (Morrison, 2003). Por esta razão, utilizou-se $n = 40$ simulações para cada caso analisado.

As experiências consistem em executar o AG para gerar um circuito lógico combinatório típico, o multiplexador 2 para 1. Na síntese deste circuito usam-se os conjuntos de portas lógicas apresentados na tabela 3.1, complementados com mais três conjuntos novos definidos na tabela 3.4.

O AG foi configurado com os seguintes parâmetros: $TC = 95\%$, $5\% \leq TM \leq 90\%$ e $6 \leq P \leq 10^4$.

Tabela 3.4 - Conjuntos de Portas Lógicas: *Gsets 1a, 1b e 5*

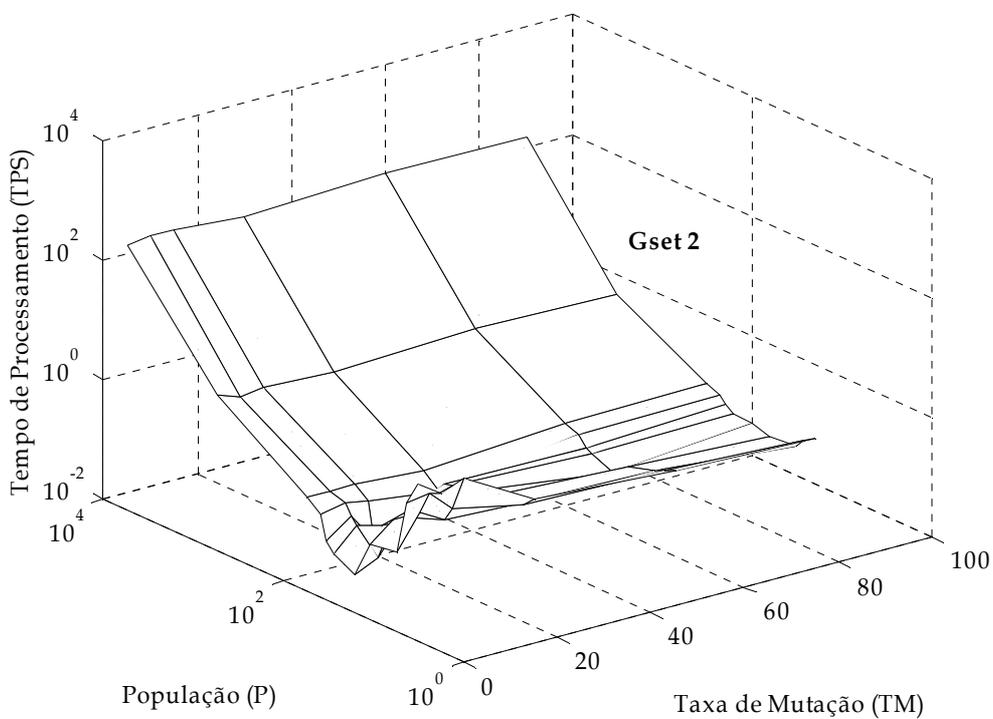
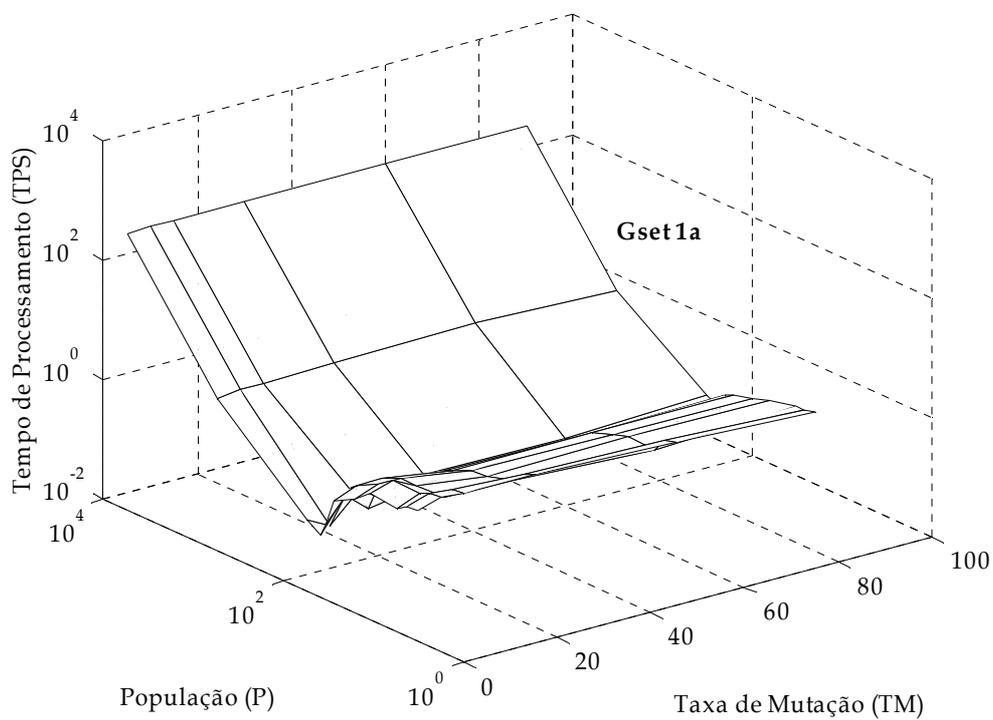
Conjunto	Portas Lógicas
<i>Gset 1a</i>	{NAND,WIRE}
<i>Gset 1b</i>	{XOR,WIRE}
<i>Gset 5</i>	{AND,OR,XOR,NOT,NAND,WIRE}

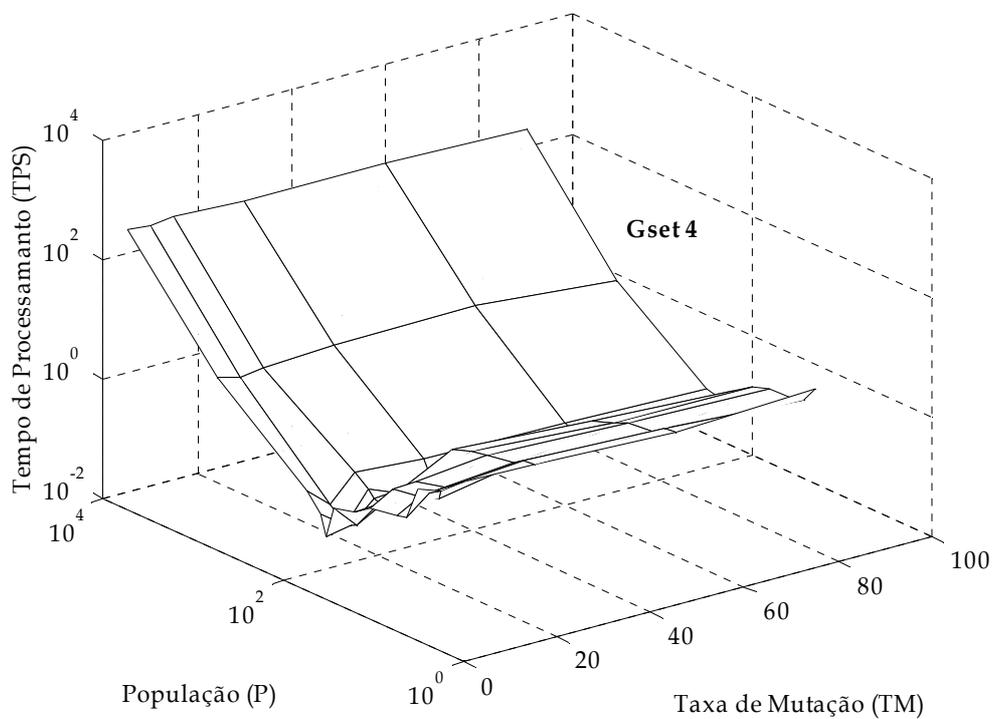
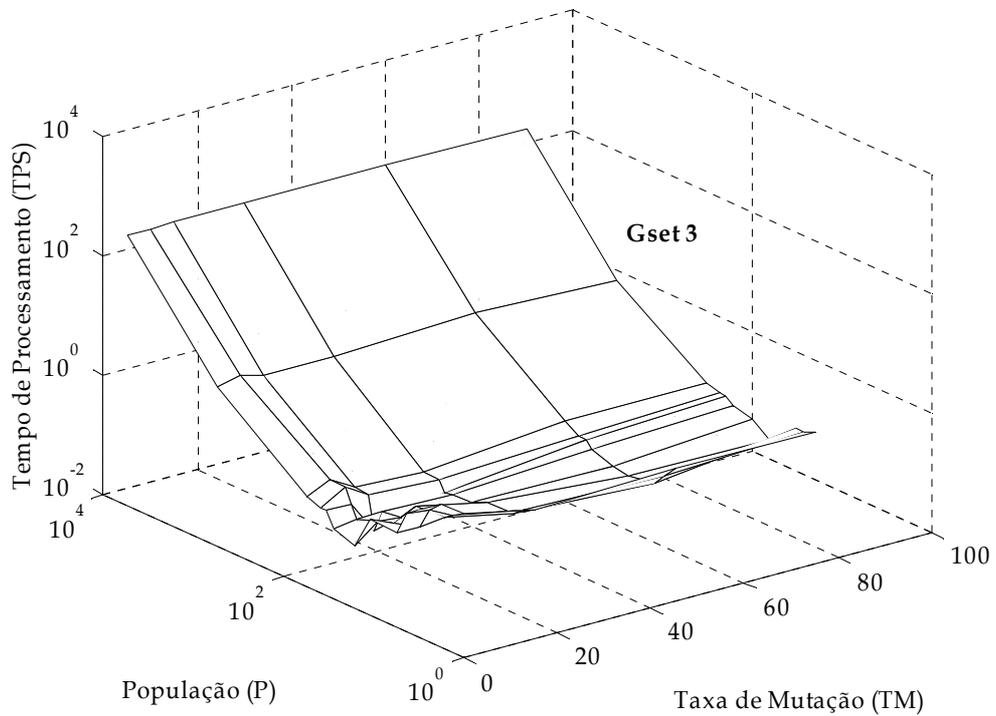
Para se obter o tempo de processamento para alcançar a solução (TPS), foram implementados os passos a seguir apresentados (Reis, Machado e Cunha, 2004a):

- i) Calculou-se a mediana $m_e(N_s)$ do número de gerações para obter a solução N_s ;
- ii) Estimou-se o tempo de processamento de uma geração TP_1 com base na média de execução relativas a cinco simulações.
- iii) O tempo de processamento para alcançar a solução é dado por:

$$TPS = m_e(N_s) \times TP_1 \quad (3.6)$$

A figura 3.21 ilustra o tempo de processamento respectivamente para os conjuntos 1a, 2, 3, 4, 5 e 6. O conjunto 1b não é apresentado porque o AG não encontra solução para este circuito.





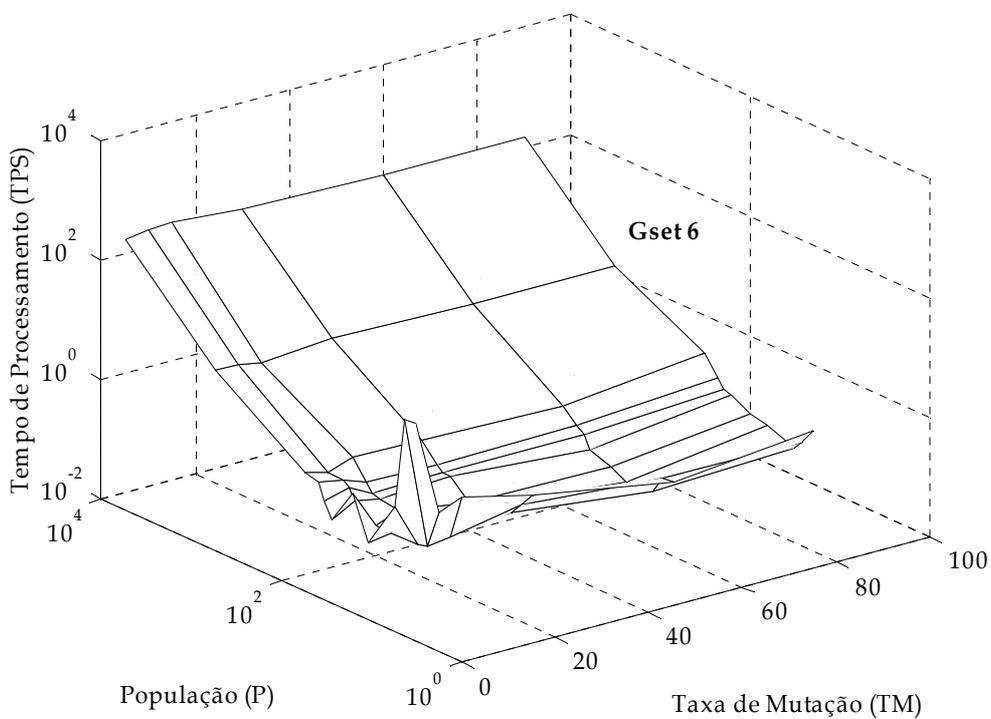
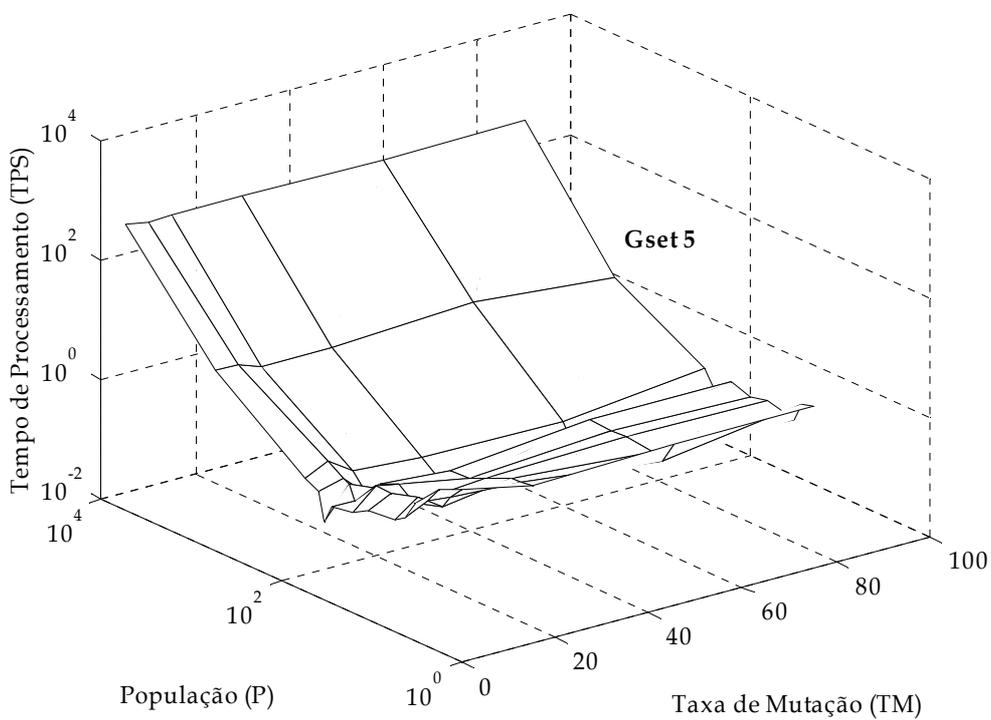


Figura 3. 21 - TPS versus (P, TM) com TC = 95% usando desde o Gset 1a até ao Gset 6 para o circuito M21.

Analisando a figura 3.21 torna-se claro que o *TPS* diminui com o tamanho da população *P*. Todavia, para tamanhos de populações muito pequenos, o AG tem extrema dificuldade em obter a solução devido à falta de BCs básicos, o que leva a um aumento de *TPS*.

Em 5% das simulações, para $P < 50$, o AG termina sem chegar à solução. Conclui-se que, para o circuito multiplexador, o tamanho da população deve ser $50 \leq P \leq 100$. A tabela 3.5 mostra para que parâmetros se obteve o melhor *TPS* para cada um dos conjuntos de portas lógicas.

O melhor *TPS* obteve-se com o conjunto *Gset 1a* ($TPS = 57,42$ ms) com $P = 70$ e $TM = 5\%$.

Tabela 3. 5 - Melhor *TPS*, utilizando o AG

Conjunto	<i>P</i>	<i>TM</i>	<i>TPS</i> (segundos)
<i>Gset 1a</i>	70	5%	0,05742
<i>Gset 2</i>	70	10%	0,11656
<i>Gset 3</i>	60	5%	0,11109
<i>Gset 4</i>	70	10%	0,13533
<i>Gset 5</i>	70	10%	0,19287
<i>Gset 6</i>	50	5%	0,13400

No que diz respeito ao parâmetro *TM*, é possível afirmar que a sua influência no tempo de processamento tem uma importância diminuta. No entanto, os melhores resultados ocorreram para $5\% \leq TM \leq 10\%$ com exceção do conjunto *Gset 6*, tal como pode ser confirmado na figura 3.22. A figura 3.22

apresenta o tempo de processamento para obter a solução TPS versus TM , para todos os conjuntos de portas lógicas.

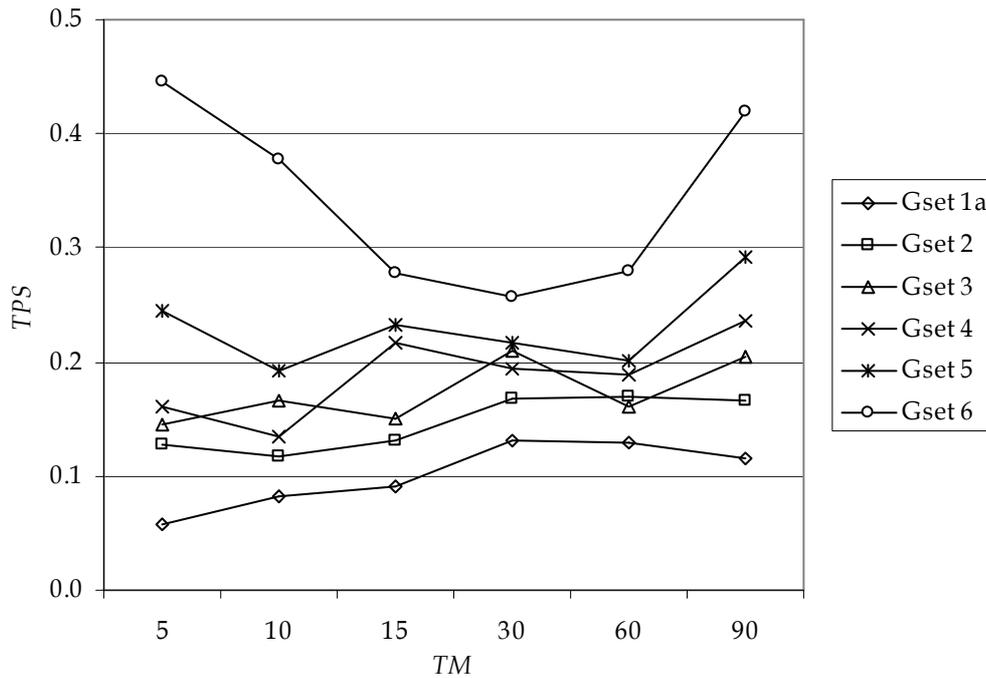


Figura 3.22 - TPS para alcançar a solução versus TM , com $TC = 95\%$ e $P = 70$, para o circuito M21.

Verifica-se também que TP_1 aumenta significativamente com P , mas que TP_1 é praticamente independente de TM . As figuras 3.23 e 3.24 ilustram TP_1 versus P e TM , respectivamente, para $TC = 95\%$ e $TM = 5\%$.

Estudando em pormenor a figura 3.23 constata-se que a representação gráfica de TP_1 segue uma lei potência do tipo:

$$TP_1 = cP^d \quad c, d \in \mathfrak{R} \quad (3.7)$$

onde $c = 7,57 \cdot 10^{-6}$ e $d = 1,54$.

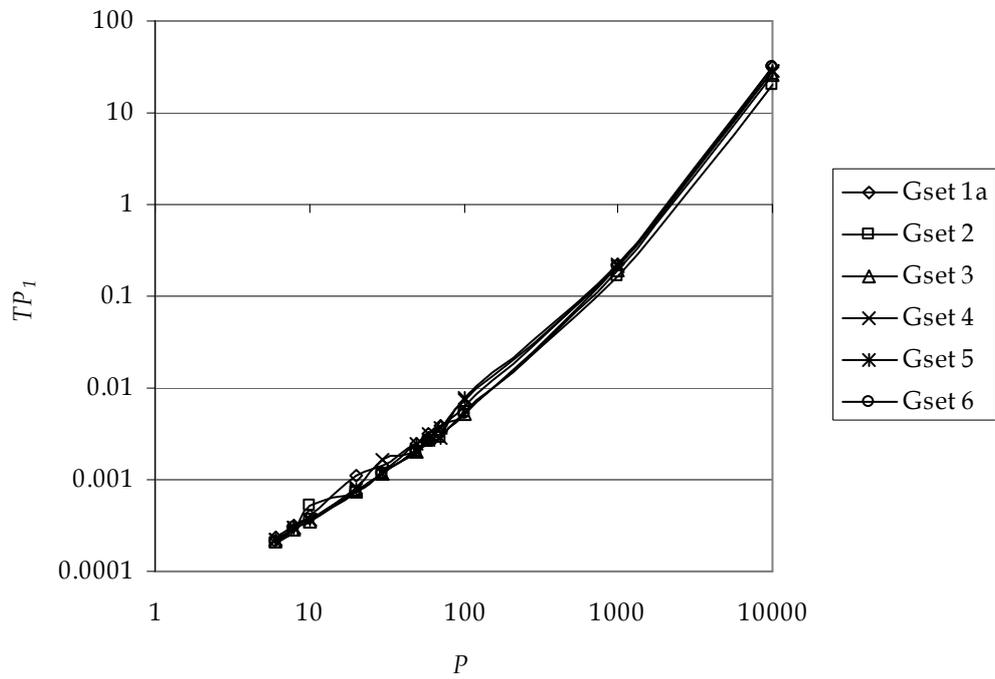


Figura 3. 23 - TP_1 versus P com $TC = 95\%$ e $TM=5\%$ para o circuito M21.

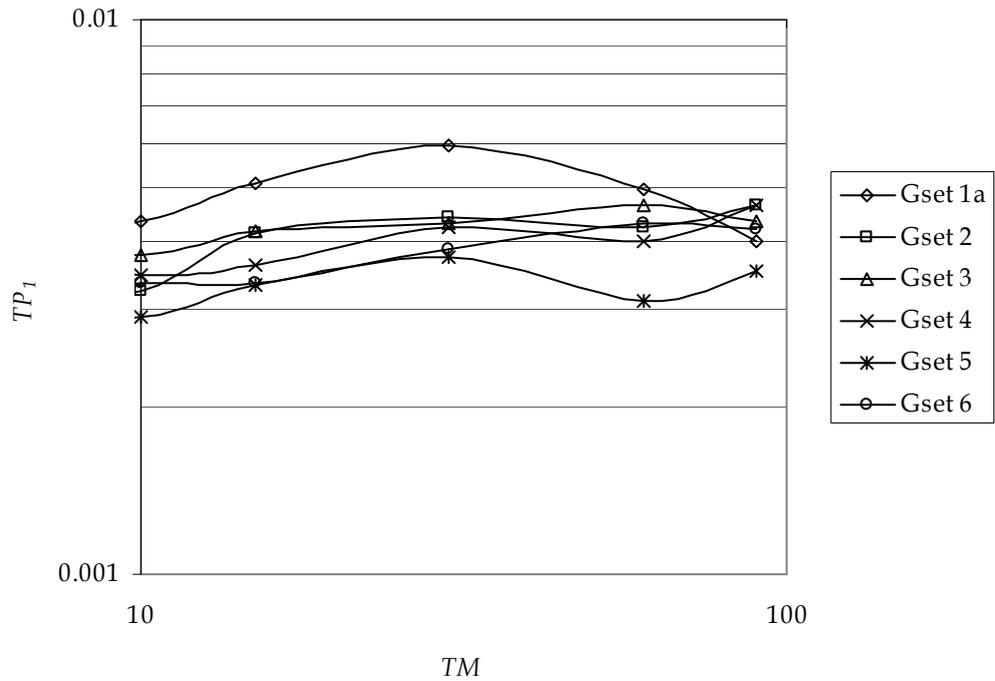


Figura 3. 24 - TP_1 versus TM com $CR = 95\%$ e $P = 70$ para o circuito M21.

6. Resumo do capítulo

Este capítulo descreve a base do AG implementado para o projecto de circuitos combinatórios em termos da codificação, operadores genéticos e função de aptidão, servindo de suporte aos capítulos seguintes. O capítulo inicia também o estudo do problema de escala na síntese de circuitos digitais e faz um estudo do tamanho da população a adoptar aquando da definição dos parâmetros do AG.

Com base nos resultados obtidos é possível delinear as seguintes conclusões:

- Os melhores resultados, no que diz respeito aos índices $\mu(N)$, $\sigma(N)$ e $\mu(F)$, foram obtidos com os conjuntos de portas lógicas de menor complexidade - *Gsets 2 e 3*;
- O AG apresenta maior dificuldade de convergência à medida que o número de *bits* dos circuitos lógicos combinatórios aumenta, o que confirma o problema de escala existente na síntese destes circuitos;
- O tempo de processamento, *TPS*, para obter a solução diminui com o tamanho da população. No entanto há um limite inferior a partir do qual o *TPS* tem um comportamento inverso.

Referências

- Coello, C. A., Christiansen, A. D. e Aguirre, A. H., "Using Genetic Algorithms to Design Combinational Logic Circuits", *Intelligent Engineering through Artificial Neural Networks*. Vol. 6, pp. 391-396, 1996.
- Goldberg, D. E., "Optimal Initial Population Size for Binary-Coded Genetic Algorithm" in *TCGA Report No 85001*, Tuscaloosa: University of Alabama, 1985.
- Goldberg, D. E., *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, 1989.
- Goldberg, D. E., *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, 2002.
- Goldberg, D., Deb, K. e Clark, J., "Genetic Algorithms, Noise, and the Sizing of Populations". *Complex Systems*, Vol. 6, pp 333-362, 1992.
- Gordon, T. G. e Bentley, P., "Towards Development in Evolvable Hardware," in *Proc. of the 2002 NASA/DOD Conference on Evolvable Hardware*, pp. 241-250, 2002.
- Harik, G., Cantu-Paz, E., Goldberg, D. e Miller, B., "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations" in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp 7-12, 1997.
- Holland, J., "Adaptation in Natural and Artificial Systems" Ann Arbor, MI:University of Michigan Press, 1975.

- Hollingworth, G. S., Smith, S. L. e Tyrrell, A. M., "The Intrinsic Evolution of Virtex Devices Through Internet Reconfigurable Logic," in Proceedings of the Third International Conference on Evolvable Systems. Vol. 1801, pp. 72-79, 2000.
- Hounsell, B. e Arslan, T., "A Novel Evolvable Hardware Framework for the Evolution of High Performance Digital Circuits," in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 525-532, 2000.
- Kalganova, T., Miller, J. F. e Lipnitskaya, N., "Multiple_Valued Combinational Circuits Synthesised using Evolvable Hardware," in Proceedings of the 7th Workshop on Post-Binary Ultra Large Scale Integration Systems, 1998.
- Khan, N., "Population Sizing in Genetic and Evolutionary Algorithms" in Proceedings of the Genetic and Evolutionary Computation Conference, 2002.
- Koza, J. R., Genetic Programming. On the Programming of Computers by means of Natural Selection, MIT Press, 1992.
- Louis, S. J. e Rawlins, G. J., "Designer Genetic Algorithms: Genetic Algorithms in Structure Design," in Proc. of the Fourth Int. Conference on Genetic Algorithms, 1991.
- Miller, B. L., "Noise, Sampling, and efficient genetic algorithms", Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1997.
- Miller, J. F., Thompson, P. e Fogarty, T, Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications. Chapter 6, Wiley, 1997.

- Morrison, R. W., "Dispersion-Based Population Initialization" in Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2003, pp 1210-1221, 2003.
- Murakawa, M., Yoshizawa, S., Kajitani, I., Furuya, T., Iwata, M., Higuchi, T. Hardware Evolution at Function Level. In Proceedings of The 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, September 22-26, 1996.
- Reis, C. e Tenreiro Machado, J. A. Synthesis of Combinational Logic Circuits using Genetic Algorithms, 8CLEEE - 8º Congresso Luso-Espanhol de Engenharia Electrotécnica, Vilamoura, Portugal, pp. 1.191-1.196, Julho, 2003a.
- Reis, C. e Machado, J. A. T., "An Evolutionary Approach to the Synthesis of Combinational Circuits", in Proceedings of the IEEE International Conference on Computational Cybernetics, 2003b.
- Reis, C. Tenreiro Machado, J. e Boaventura Cunha, J., Population Size and Processing Time in a Genetic Algorithm, in Proceedings of the IEEE International Conference on Computational Cybernetics, Vienna University of Technology, Austria, 2004a.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Evolutionary Design of Combinational Logic Circuits, Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Vol. 8, No. 5, pp. 507-513, September, 2004b.
- Thompson, A. e Layzell, P. "Analysis of unconventional evolved electronics," Communications of the ACM, Vol. 42, pp. 71-79, 1999.
- Torresen, J., "A Divide-and-Conquer Approach to Evolvable Hardware," in Proceedings of the Second International Conference on Evolvable Hardware, Vol. 1478, pp. 57-65, 1998.

Vassilev, V. K. e Miller, J. F., "Scalability Problems of Digital Circuit Evolution," in Proc. of the Second NASA/DOD Workshop on Evolvable Hardware, pp. 55-64, 2000.

Zebulum, R. S., Pacheco, M. A. e Vellasco, M. M., *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*, CRC Press, 2001.

Capítulo 4

SÍNTESE DE CIRCUITOS COM ALGORITMOS GENÉTICOS E CÁLCULO FRACCIONÁRIO

Introdução

Neste capítulo são propostos dois algoritmos genéticos para a síntese de circuitos lógicos combinatórios. Os algoritmos propostos exploram novas formas de avaliação dos circuitos digitais. O primeiro algoritmo usa uma função de aptidão que, além de testar cada uma das linhas da tabela de verdade do circuito a implementar, avalia também a descontinuidade do erro entre duas linhas consecutivas dessa tabela. Por este motivo, as tabelas de verdade são construídas aplicando o código *Gray*. O segundo algoritmo aplica ferramentas do Cálculo Fraccionário (CF) na avaliação dos circuitos, o que permite uma avaliação avançada, dado que esta não diz respeito somente ao presente, mas também ao histórico da evolução dos circuitos. Esta abordagem dá origem a uma função de avaliação dos circuitos digitais que se designa por função de aptidão dinâmica.

Nesta ordem de ideias o capítulo está organizado em quatro secções. A secção 1 é dedicada ao Cálculo Fraccionário, dando suporte teórico às secções

seguintes; a secção 2 define as funções de aptidão, estática e dinâmica; as experiências desenvolvidas são detalhadas na secção 3 e por último, na secção 4, resumem-se os aspectos principais deste capítulo.

1. Cálculo Fraccionário

A área do CF lida com operadores de integração e diferenciação de ordem arbitrária, por vezes designada como não-inteira teve a sua origem juntamente com o cálculo diferencial clássico (Oldham e Spanier, 1974; Miller e Ross, 1993). A teoria do CF constitui uma ferramenta bem adaptada à modelização de muitos fenómenos físicos, permitindo que a descrição tenha em consideração algumas peculiaridades que os modelos clássicos de ordem inteira simplesmente negligenciam. A aplicação de CF esteve “adormecida” até há bem pouco tempo. No entanto, os avanços na teoria do caos motivaram um novo interesse nesta área. Nas últimas duas décadas desenvolveu-se investigação na viscoelasticidade, caos/fractais, biologia, processamento de sinal, identificação de sistemas, difusão e propagação de ondas, electromagnetismo e controlo automático (Chen e Moore, 2002; Koh e Kelly, 1990; Méhauté, 1991; Oustaloup, 1995; Tenreiro, 1997; Torvik e Bagley, 1984; Westerlund, 2002).

A generalização do conceito de diferenciação $D^\alpha[f(x)]$ para valores de α não inteiros surge logo no início da teoria do cálculo diferencial. De facto, Leibniz, numa troca de correspondência com Bernoulli, L'Hôpital e Wallis, figuram várias notas de cálculos para $\alpha = 1/2$ (Miller e Ross, 1993; Oldham e Spanier, 1974). Não obstante, a adopção do CF em algoritmos de controlo foi somente estudada recentemente, quer no domínio das frequências, quer no domínio do tempo discreto (Koh e Kelly, 1990; Méhauté, 1991; Oustaloup, 1995; Tenreiro, 1997).

A definição matemática de derivada de ordem fraccionária α tem sido tema de abordagens diferentes. Por exemplo, as equações 4.1 e 4.2 representam as definições de derivada fraccionária de ordem α do sinal $x(t)$, respectivamente segundo as definições de Laplace (para condições iniciais iguais a zero) e de Grünwald-Letnikov:

$$D^\alpha [x(t)] = L^{-1} \{ s^{-\alpha} X(s) \} \quad (4.1)$$

$$D^\alpha [x(t)] = \lim_{h \rightarrow 0} \frac{1}{h^\alpha} \sum_{k=0}^{\infty} \frac{(-1)^k \Gamma(\alpha + 1)}{k! \Gamma(\alpha - k + 1)} x(t - kh) \quad (4.2)$$

onde Γ é a função gama e h o incremento de tempo.

Esta formulação (Tenreiro, 1997) inspirou o desenvolvimento de um algoritmo de cálculo discreto no domínio dos tempos, baseado na aproximação do incremento de tempo h através do período de amostragem T e numa série truncada de r -termos, tal como é apresentado na equação seguinte:

$$D^\alpha [x(t)] \approx \frac{1}{T^\alpha} \sum_{k=0}^r \frac{(-1)^k \Gamma(\alpha + 1)}{k! \Gamma(\alpha - k + 1)} x(t - kT) \quad (4.3)$$

2. Funções de aptidão estática e dinâmica

O objectivo deste estudo é o de encontrar novas técnicas de avaliação dos indivíduos de uma população de forma a obter AGs com melhores desempenhos. Para esse efeito, são propostos dois conceitos novos para melhorar a função de aptidão estática padrão F_e com:

- Introdução da medida da descontinuidade do erro;
- Recurso a uma função de avaliação dinâmica.

A introdução da medida da descontinuidade do erro na função de aptidão estática vai permitir avaliar não só os *bits* da função de saída da tabela de verdade, mas também o erro entre linhas adjacentes da tabela.

A ideia de usar memória no sentido de se conseguirem melhores desempenhos das funções de aptidão foi introduzida pela primeira vez por Sano e Kita (Sano e Kita, 2000 e 2002). O objectivo é a optimização de sistemas com funções de aptidão com flutuação aleatória. Para isso, desenvolveram um AG com avaliação da função de aptidão baseada em memória (MFEGA - *Memory-based Fitness Evaluation*). Este conceito baseia-se no armazenamento em memória dos valores da função de aptidão anteriores, criando um histórico e, com isso, introduzindo um modelo estocástico simples para estimar os valores de maior interesse da função de aptidão.

No nosso caso, recolher informação para memória dos valores da função de aptidão, tem o objectivo de avaliar a evolução dos indivíduos da população. Desta forma, o valor da função de aptidão de cada indivíduo é baseado não só no presente, mas também no seu historial. Esta função de aptidão designa-se por função de aptidão dinâmica F_d .

O cálculo de F_e na equação 4.8 está dividido em duas partes, f_1 e f_2 , onde f_1 mede a funcionalidade dos circuitos e a descontinuidade do erro e f_2 mede a simplicidade do circuito digital.

Numa primeira fase compara-se a saída \mathbf{Y} , produzida pelo AG ao gerar o circuito, com os valores pretendidos \mathbf{Y}_R , de acordo com a tabela de verdade, *bit*

a *bit*. Por outras palavras, o índice f_{11} que mede o número de *bits* que estão correctos na tabela de verdade, é incrementado de uma unidade por cada *bit* correcto da saída até que f_{11} atinja o valor máximo f_{10} , que ocorre quando se obtém um circuito funcional (equações 4.4 e 4.5).

Para se medir a variabilidade do erro de saída, f_{11} é decrementado de $\delta \in [0, 1]$ para cada descontinuidade do erro $\mathbf{Y}_R - \mathbf{Y}$, onde descontinuidade significa passar de $\mathbf{Y}_R - \mathbf{Y} = 0$ para $\mathbf{Y}_R - \mathbf{Y} = 1$ ou, vice-versa, quando se comparam duas linhas consecutivas da tabela de verdade (equação 4.6).

Assim que o circuito fica funcional, numa segunda fase o AG tenta gerar circuitos com o menor número de portas lógicas. Isto significa que, o circuito resultante deve conter o maior número de genes possível de *<tipo de porta>* \equiv *<wire>*. O índice f_2 , que avalia a simplicidade (o número de operações nulas), é incrementado de *1 unidade* (*0 unidades*) para cada *wire* (*porta lógica*) do circuito gerado (equação 4.7). Desta forma a função de aptidão é calculada de acordo com:

- Primeira fase, funcionalidade do circuito:

$$f_{10} = 2^{ni} \times no \quad (4.4)$$

$$f_{11} = f_{11} + 1 \text{ se } \{\textit{bit } i \text{ de } \mathbf{Y}\} = \{\textit{bit } i \text{ de } \mathbf{Y}_R\}, i = 1, \dots, f_{10} \quad (4.5)$$

$$f_1 = f_{11} - \delta \text{ se } \textit{erro}_i \neq \textit{erro}_{i-1}, i = 1, \dots, f_{10} \quad (4.6)$$

(quando se mede a descontinuidade)

- Segunda fase, simplicidade do circuito:

$$f_2 = f_2 + 1 \text{ se } \textit{tipo de porta} = \textit{wire} \quad (4.7)$$

$$F_e = \begin{cases} f_1, & F_e < f_{10} \\ f_1 + f_2, & F_e \geq f_{10} \end{cases} \quad (4.8)$$

onde n_i e n_o representam o número de entradas e saídas do circuito.

O conceito da função de aptidão dinâmica F_d emerge da analogia entre os AG e os sistemas de controlo. No caso do AG, o objectivo consiste em controlar uma população através da função de aptidão. Nesta linha de orientação, a função de aptidão estática F_e corresponde a um tipo de algoritmo proporcional enquanto a função de aptidão dinâmica F_d pode ser implementada da seguinte forma:

$$F_d = F_e + KD^\alpha [F_e] \quad (4.9)$$

onde $-1 \leq \alpha \leq 1$ representa a ordem fraccionária diferencial (integral) para valores positivos (negativos) de α e K é o “ganho” do termo dinâmico.

A equação 4.9 pode ser apresentada separando as componentes integral e diferencial, como se mostra na equação 4.10:

$$F_d = F_e + K_I I^\lambda [F_e] + K_D D^\mu [F_e] \quad (4.10)$$

onde $0.0 \leq \lambda \leq 1.0$ é o coeficiente integral de ordem fraccionária, $0.0 \leq \mu \leq 1.0$ é o coeficiente diferencial de ordem fraccionária e K_I , K_D são, respectivamente, os “ganhos” integral e diferencial do termo dinâmico.

3. Experiências desenvolvidas

Neste conjunto de experiências desenvolveram-se $n = 1000$ simulações para

cada um dos casos testados. As experiências consistiram na execução do AG para projectar três circuitos lógicos combinatórios típicos, o multiplexador dois para um (M21), o circuito de teste de paridade de 4 bits (TP4) e o circuito somador completo de um *bit* (SOM1), usando as funções de aptidão descritas na secção anterior.

Usaram-se dois conjuntos de portas lógicas, o *Gset 2* e o *Gset 4* definidos na tabela 3.1 do capítulo 3, por serem representativos dos dois tipos de conjuntos de portas lógicas estabelecidos no capítulo anterior, respectivamente, como RISC e CISC.

As taxas de cruzamento, mutação e tamanho da população foram especificados, respectivamente, com os valores $TC = 95\%$, $TM = 20\%$ e $P = 100$.

Ter um AG com desempenho superior significa obter as soluções no menor número de gerações N , com o menor desvio padrão possível, reduzindo conseqüentemente a natureza estocástica do algoritmo.

3.1. Utilizando a função de aptidão estática F_e

Nesta sub-secção analisam-se os efeitos quando se adopta a função de aptidão estática que inclui a medida da descontinuidade do erro δ .

As figuras 4.1 a 4.3 ilustram a média do número de gerações para obtenção da solução $\mu(N)$ e correspondente desvio padrão *versus* factor de descontinuidade $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ usando, respectivamente, os conjuntos *Gset 2* e *Gset 4*, para os circuitos M21, TP4 e SOM1.

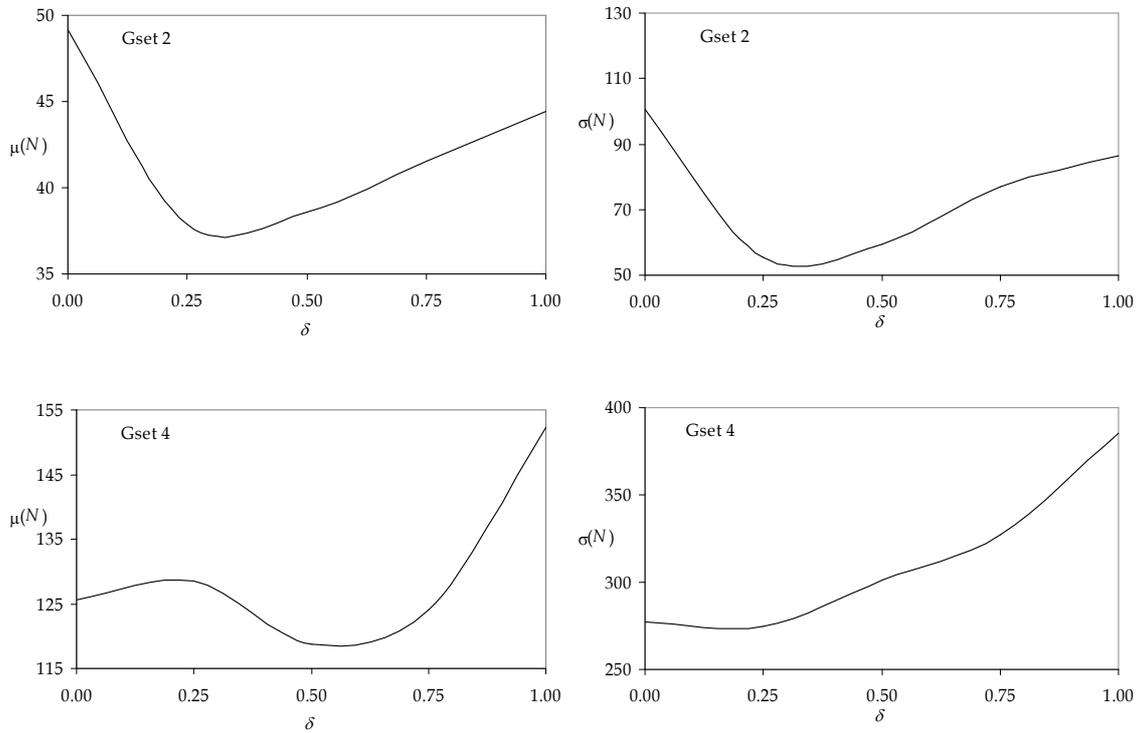


Figura 4. 1- Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $\delta = \{0, 0.25, 0.5, 0.75, 1\}$ com os conjuntos *Gset 2* e *Gset 4*, para o circuito M21.

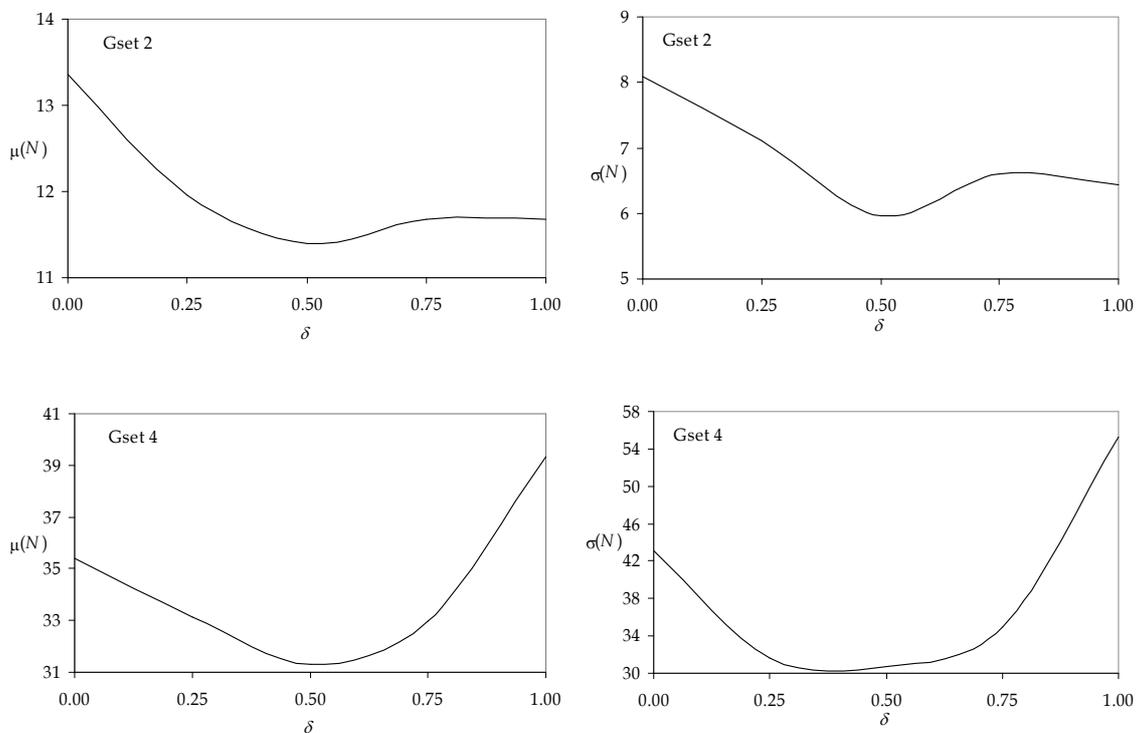


Figura 4. 2 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $\delta = \{0, 0.25, 0.5, 0.75, 1\}$ com os conjuntos *Gset 2* e *Gset 4*, para o circuito TP4.

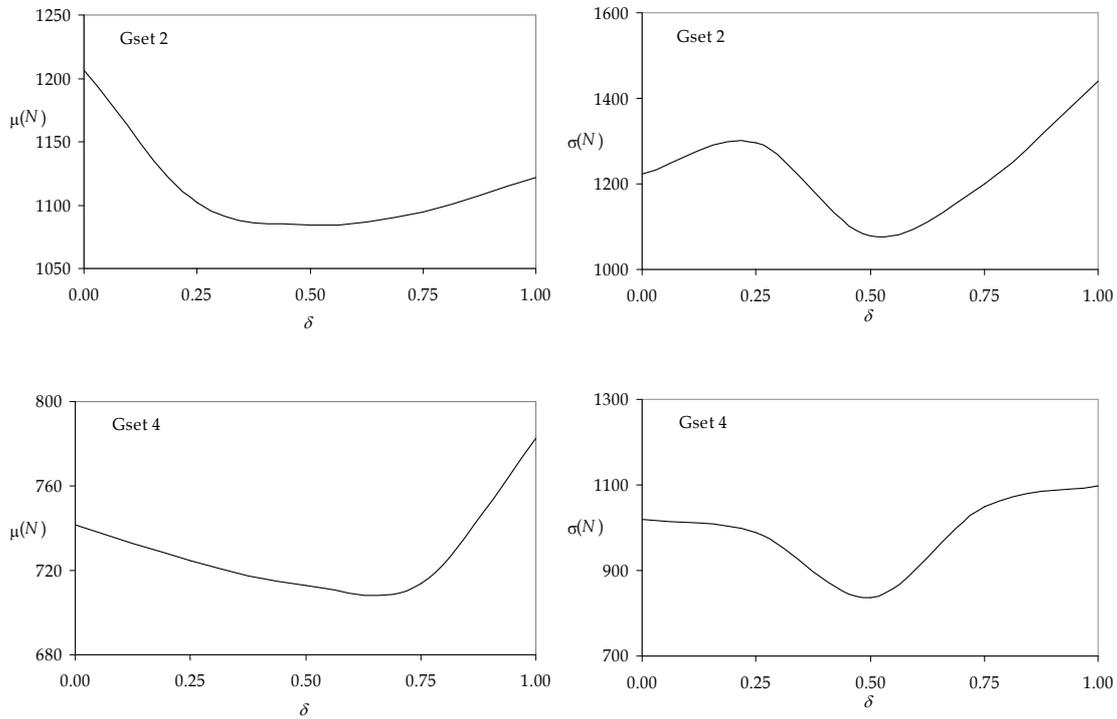


Figura 4.3 – Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $\delta = \{0, 0.25, 0.5, 0.75, 1\}$ com os conjuntos *Gset 2* e *Gset 4*, para o circuito SOM1.

Os resultados obtidos revelam, tal como esperado (Reis, Machado e Cunha, 2004), que o conjunto de portas lógicas do tipo RISC (*Gset 2*) apresenta melhor desempenho em relação ao conjunto do tipo CISC (*Gset 4*) para todos os valores de δ . Por outro lado, examinando a influência de δ , conclui-se que a resposta do AG é superior para valores na região de $\delta = 0.5$ para os dois circuitos e os dois conjuntos de portas lógicas.

3.2. Utilizando a função de aptidão dinâmica F_d

Nesta sub-seção analisa-se o desempenho do AG quando se adopta a função de aptidão dinâmica (equação 4.10).

O primeiro conjunto de simulações investiga, separadamente, o esquema diferencial com $\mu = \{0,0; 0,25; 0,5; 0,75; 1,0\}$ e o esquema integral com $\lambda = \{0,0; 0,25; 0,5; 0,75; 1,0\}$, em F_d para ganhos de $10^{-3} \leq K_D \leq 10^2$ e $10^{-3} \leq K_I \leq 10^2$. A implementação do operador de ordem integral/diferencial fraccionário adota a equação 4.3 com uma série truncada de $r = 50$ termos.

As figuras 4.4 a 4.15 mostram a média $\mu(N)$ e o respectivo desvio padrão $\sigma(N)$ do número de gerações para obter a solução para os esquemas diferencial PD^μ (i.e., $K_I = 0.0$) e integral PI^λ (i.e., $K_D = 0.0$), para os circuitos M21, TP4 e SOM1, com os conjuntos de portas lógicas *Gset 2* e *Gset 4*. Os gráficos incluem os traçados para $\mu = 0.0$ e $\lambda = 0.0$, ou seja sem a função de aptidão dinâmica, para facilitar a comparação dos resultados.

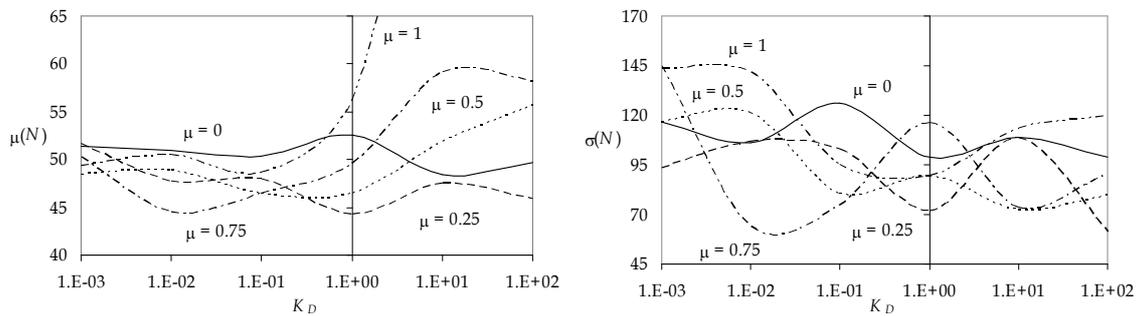


Figura 4. 4 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto *Gset 2*, para o circuito M21.

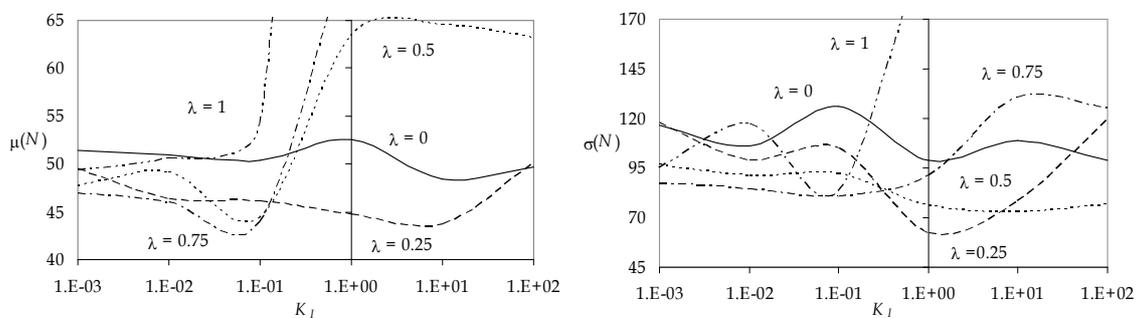


Figura 4. 5 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto *Gset 2*, para o circuito M21.

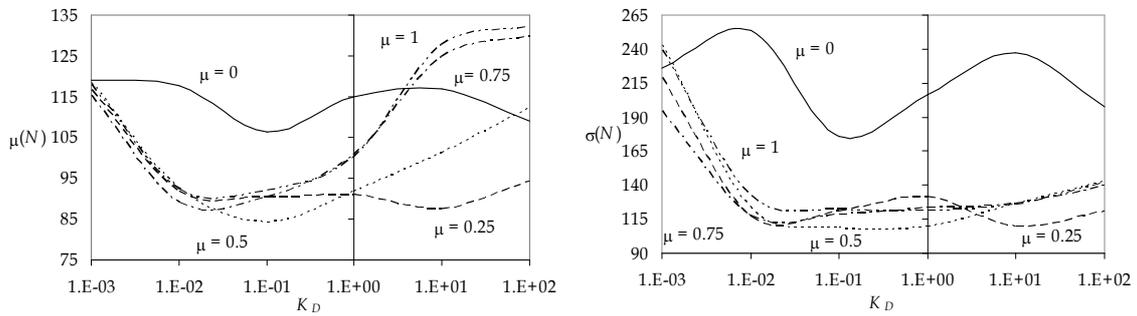


Figura 4. 6 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 4$, para o circuito M21.

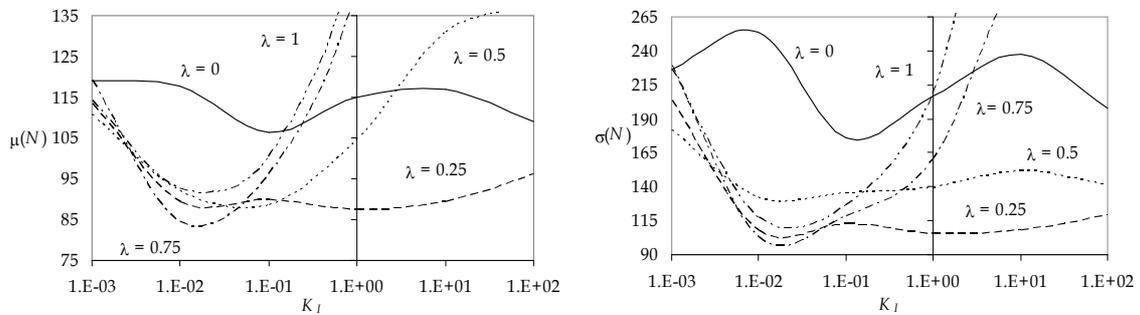


Figura 4. 7 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta = 0$) com o conjunto $Gset 4$, para o circuito M21.

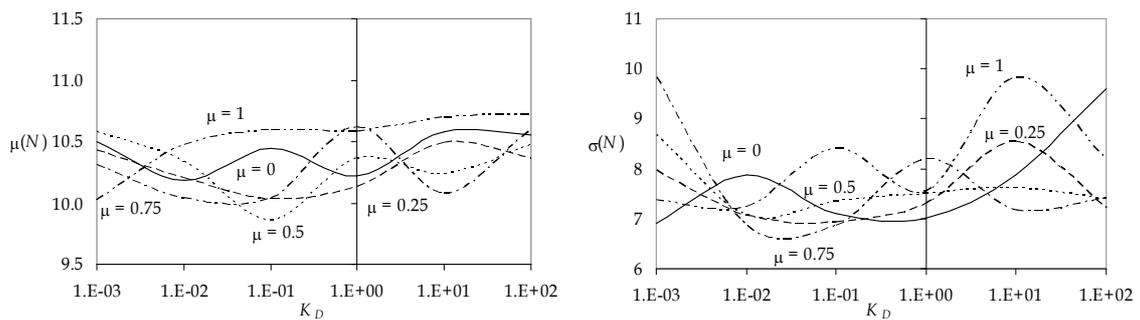


Figura 4. 8 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta = 0$) com o conjunto $Gset 2$, para o circuito TP4.

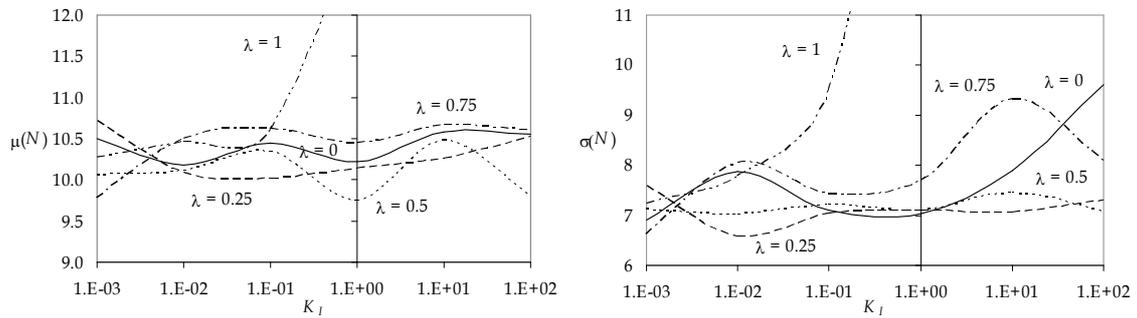


Figura 4. 9 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta=0$) com o conjunto *Gset 2*, para o circuito TP4.

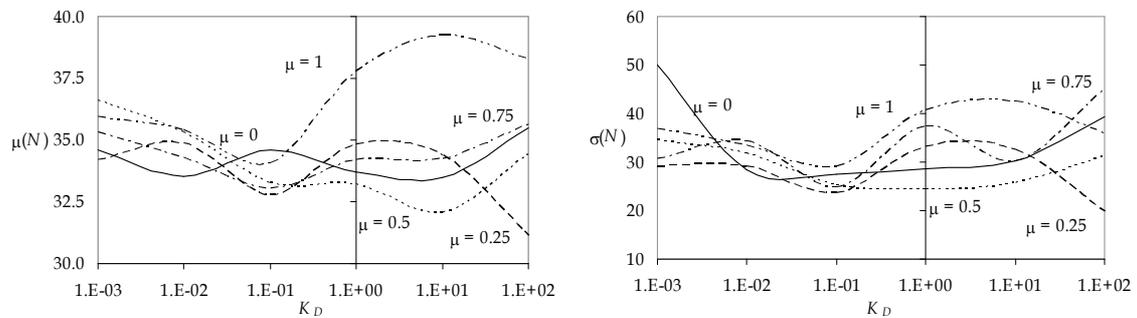


Figura 4. 10 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta=0$) com o conjunto *Gset 4*, para o circuito TP4.

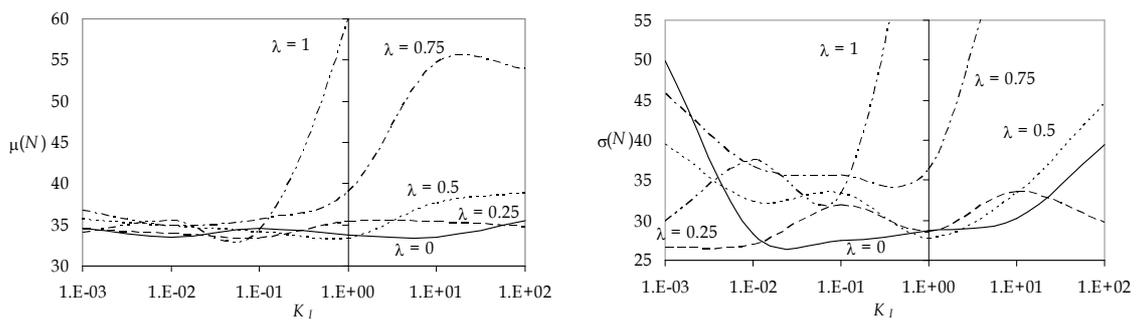


Figura 4. 11 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta=0$) com o conjunto *Gset 4*, para o circuito TP4.

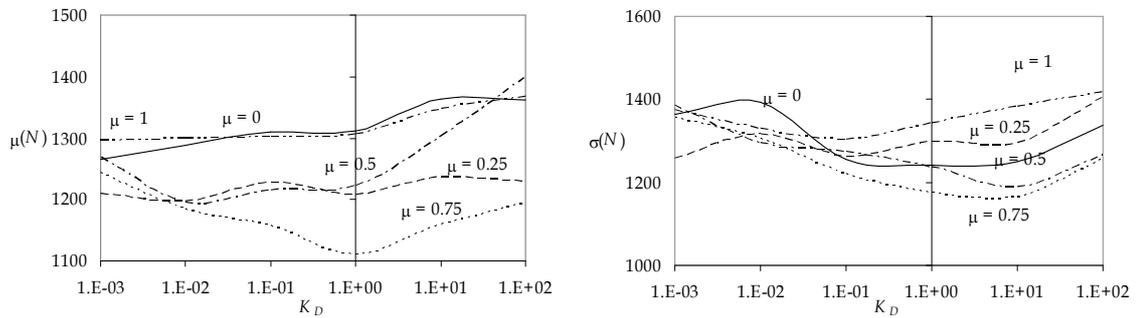


Figura 4. 12 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta=0$) com o conjunto *Gset 2*, para o circuito SOM1.

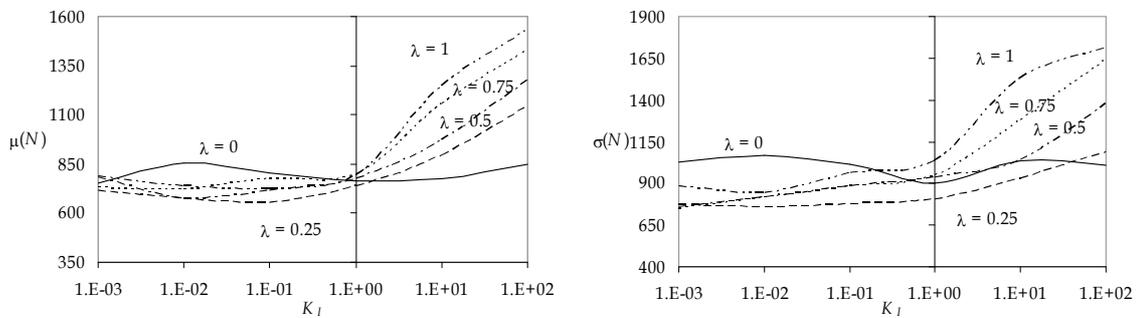


Figura 4. 13 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta=0$) com o conjunto *Gset 2*, para o circuito SOM1.

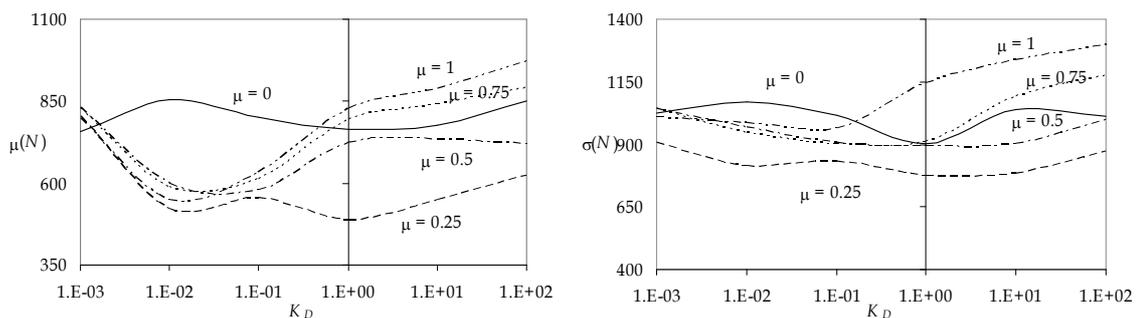


Figura 4. 14 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PD^μ ($\delta=0$) com o conjunto *Gset 4*, para o circuito SOM1.

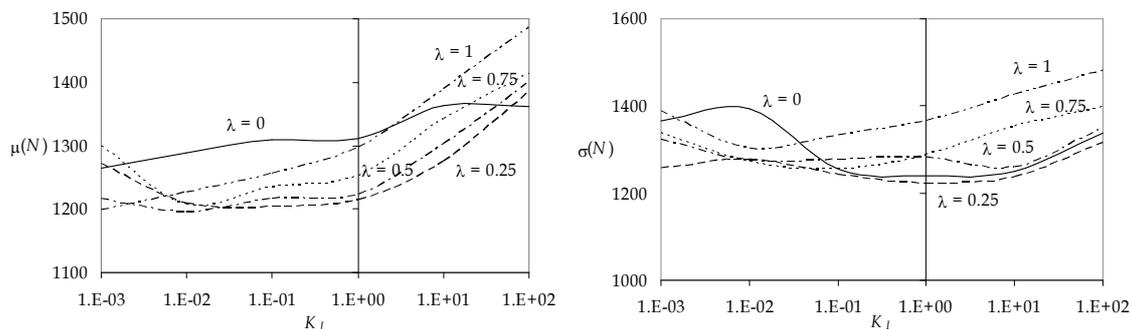


Figura 4. 15 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema PI^λ ($\delta=0$) com o conjunto $Gset 4$, para o circuito SOM1.

Analisando os gráficos das figuras 4.4 a 4.15 constata-se a superioridade de desempenho do AG para valores de k (ganhos) inferiores a 1, em particular para os esquemas diferenciais da função de aptidão dinâmica.

As tabelas 4.1 e 4.2 exibem os pares de parâmetros (μ, K_D) ou (λ, K_I) para cada melhor solução obtida em termos da média do número de gerações $\mu(N)$ e em termos do desvio padrão $\sigma(N)$, respectivamente, para os esquemas integral e diferencial de F_d .

Tabela 4. 1- Parâmetros (μ, K_D) ou (λ, K_I) para cada melhor solução em termos de $\mu(N)$

Circuito	$Gset 2$	$Gset 4$
M21	$(\mu, K_D) = (0.25, 1)$	$(\mu, K_D) = (0.5, 0.1)$
	$(\mu, K_I) = (0.25, 10)$	$(\lambda, K_I) = (0.75, 0.01)$
TP4	$(\mu, K_D) = (0.5, 0.1)$	$(\mu, K_D) = (0.25, 100)$
	$(\mu, K_I) = (0.5, 1)$	$(\lambda, K_I) = (0.25, 0.1)$
SOM1	$(\mu, K_D) = (0.75, 1)$	$(\mu, K_D) = (0.5, 0.1)$
	$(\mu, K_I) = (0.5, 0.01)$	$(\mu, K_I) = (0.25, 0.1)$

Tabela 4. 2 - Parâmetros (μ, K_D) ou (λ, K_I) para cada melhor solução em termos de $\sigma(N)$

Circuito	Gset 2	Gset 4
M21	$(\mu, K_D) = (0.25, 100)$	$(\mu, K_D) = (0.5, 0.1)$
	$(\lambda, K_I) = (0.25, 1)$	$(\lambda, K_I) = (0.75, 0.01)$
TP4	$(\mu, K_D) = (0.75, 0.1)$	$(\mu, K_D) = (0.25, 100)$
	$(\lambda, K_I) = (0.25, 0.01)$	$(\lambda, K_I) = (0.25, 0.001)$
SOM1	$(\mu, K_D) = (0.75, 10)$	$(\mu, K_D) = (0.75, 0.1)$
	$(\mu, K_I) = (0.25, 1)$	$(\mu, K_I) = (0.25, 0.01)$

Genericamente, conclui-se que o conceito da função de aptidão dinâmica F_d produz resultados superiores em particular no esquema diferencial. Além disso e uma vez mais, o conjunto de portas lógicas do tipo RISC apresenta melhor desempenho do que o conjunto do tipo CISC.

Dado que os melhores resultados foram obtidos para $\mu = 0.25$ e $\lambda = 0.25$, decidi investigar-se a combinação destes parâmetros. Consequentemente, o segundo conjunto de simulações avalia o esquema proporcional-integral-derivativo $PI^\lambda D^\mu$. No entanto, devido ao elevado número de combinações possíveis de $\{\lambda, \mu, K_I, K_D\}$ estabeleceu-se $\lambda = \mu = 0.25$ e $10^{-3} \leq K_D = K_I \leq 10^2$.

Os resultados destas simulações podem ser visualizados nas figuras 4.16 a 4.21, que ilustram a média do número de gerações para obter a solução $\mu(N)$ e o desvio padrão $\sigma(N)$, com o esquema $PI^\lambda D^\mu$, para os circuitos M21, TP4 e SOM1, utilizando os conjuntos de portas lógicas Gset 2 e Gset 4 ($\delta = 0$).

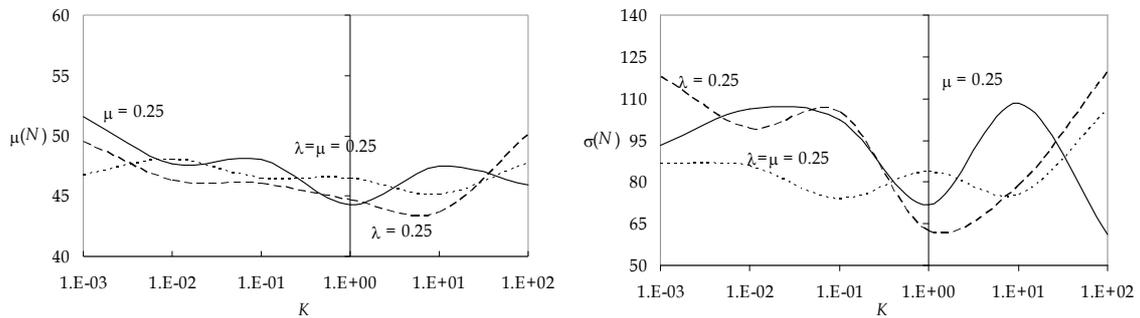


Figura 4. 16 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^{\lambda}D^{\mu}$, para $K = K_D = K_I$ e $(\delta = 0)$ com o conjunto *Gset 2*, para o circuito M21.

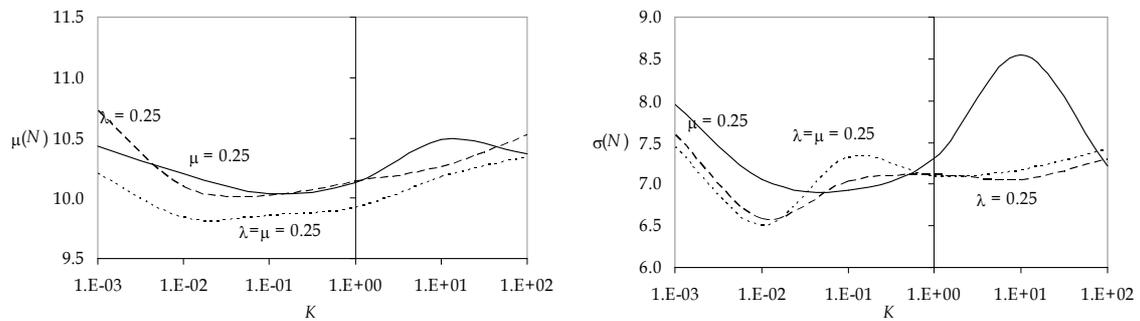


Figura 4. 17 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^{\lambda}D^{\mu}$, para $K = K_D = K_I$ e $(\delta = 0)$ com o conjunto *Gset 2*, para o circuito TP4.

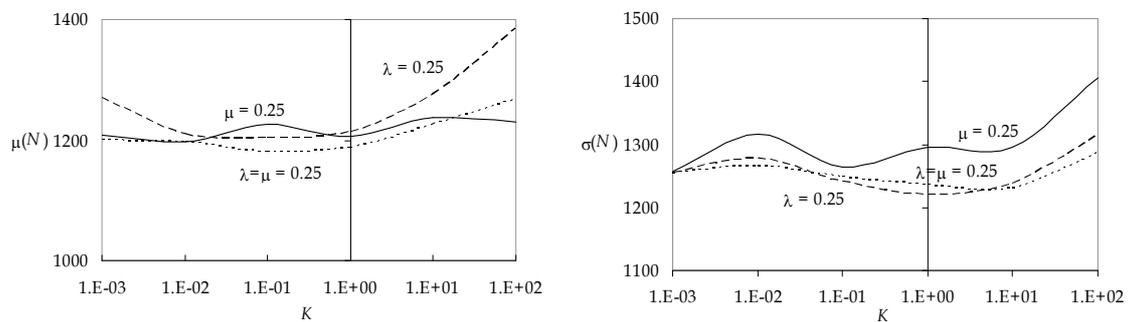


Figura 4. 18 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^{\lambda}D^{\mu}$, para $K = K_D = K_I$ e $(\delta = 0)$ com o conjunto *Gset 2*, para o circuito SOM1.

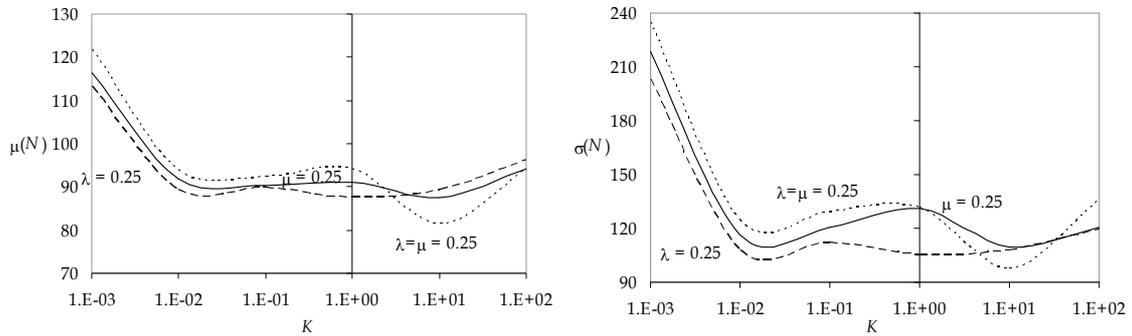


Figura 4. 19 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^{\lambda}D^{\mu}$, para $K = K_D = K_I$ e $(\delta = 0)$ com o conjunto $Gset 4$, para o circuito M21.

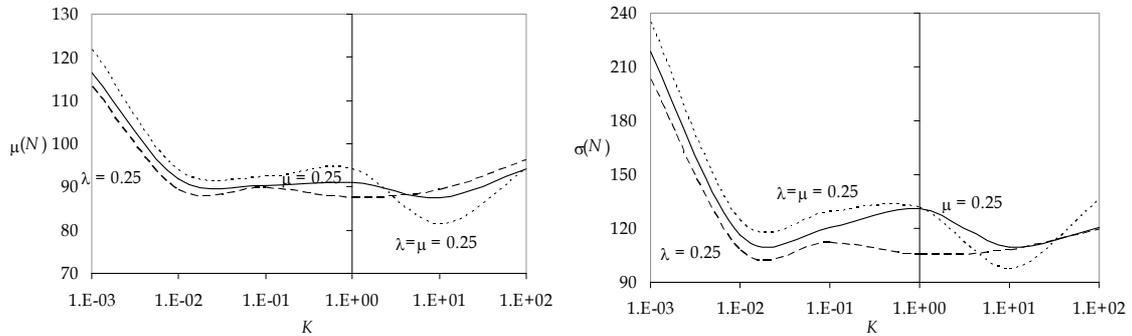


Figura 4. 20 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^{\lambda}D^{\mu}$, para $K = K_D = K_I$ e $(\delta = 0)$ com o conjunto $Gset 4$, para o circuito TP4.

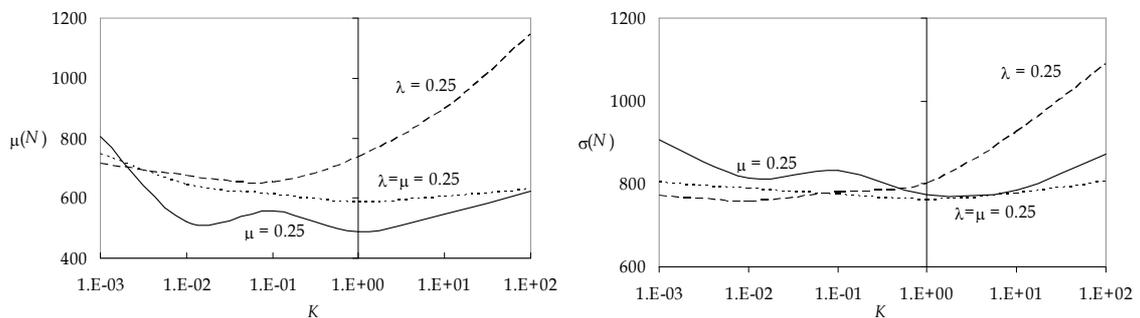


Figura 4. 21 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para o esquema $PI^{\lambda}D^{\mu}$, para $K = K_D = K_I$ e $(\delta = 0)$ com o conjunto $Gset 4$, para o circuito SOM1.

Comparando os esquemas $PD^{1/4}$ e $PI^{1/4}$ com o caso $PI^{1/4}D^{1/4}$ e observando os gráficos das figuras 4.16 a 4.18 que apresentam os resultados obtidos com o *Gset 2* para os circuitos M21, TP4 e SOM1 verifica-se que a inclusão das duas acções, a acção integral e a diferencial, melhora ligeiramente os resultados para valores de k inferiores a 1.

Analisando as figuras 4.19 a 4.21, que apresentam os resultados obtidos com o *Gset 4* para os mesmos circuitos, constata-se que a adopção do esquema $PI^{1/4}D^{1/4}$ não altera significativamente os resultados.

No terceiro conjunto de simulações inseriu-se, no esquema $PI^{1/4}D^{1/4}$, a medida da descontinuidade do erro, o que deu origem aos gráficos das figuras 4.22 a 4.27.

Neste caso é notório a melhoria do desempenho do AG, quer em termos da média do número de gerações para obtenção da solução, quer no que diz respeito ao desvio padrão do número de gerações para a obtenção da solução, principalmente nas simulações com o conjunto de portas lógicas *Gset 2*.

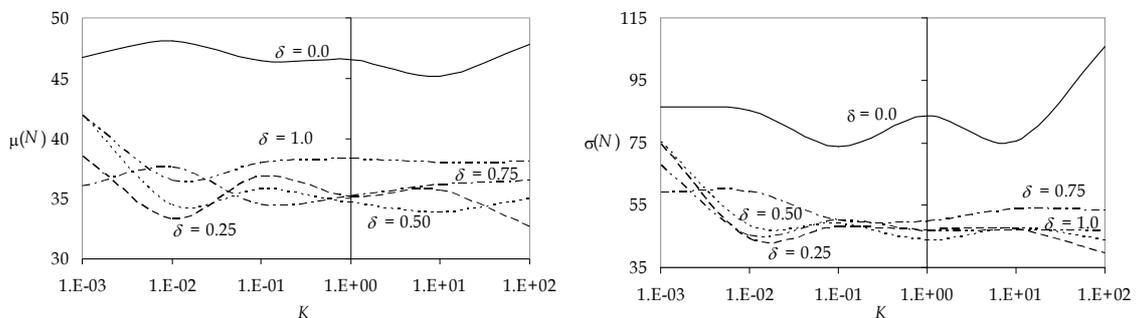


Figura 4. 22 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto *Gset 2*, para o circuito M21.

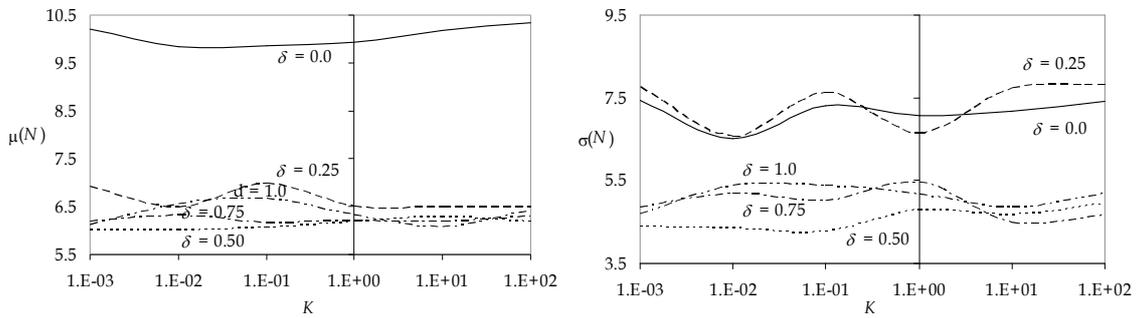


Figura 4. 23 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto *Gset 2*, para o circuito TP4.

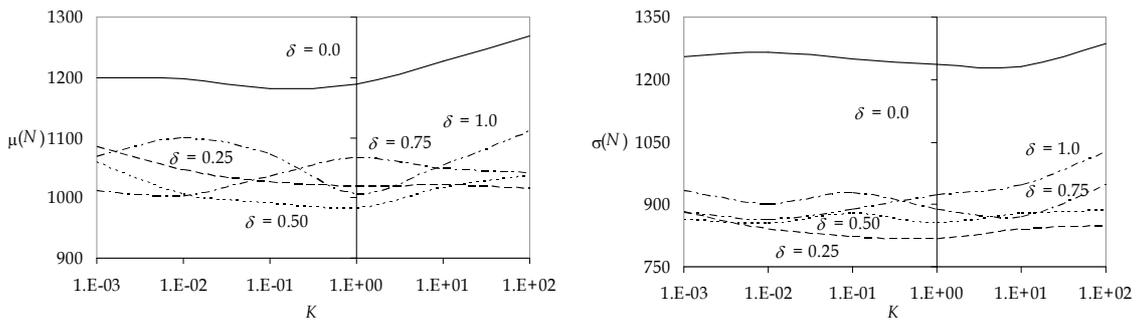


Figura 4. 24 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto *Gset 2*, para o circuito SOM1.

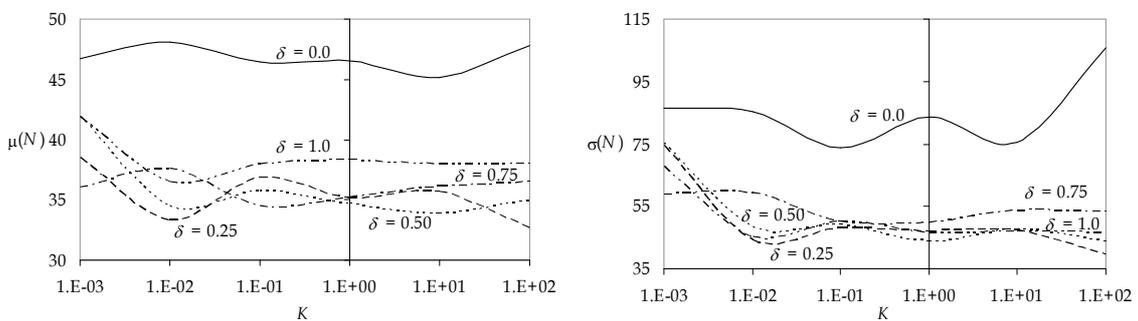


Figura 4. 25 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto *Gset 4*, para o circuito M21.

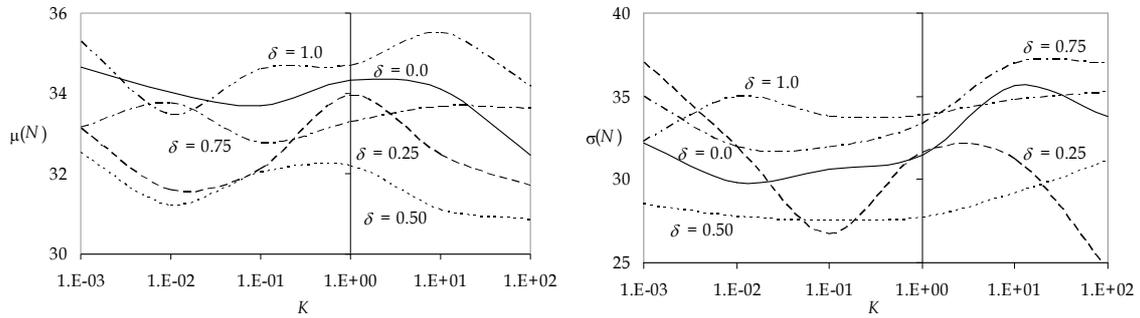


Figura 4. 26 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto *Gset 4*, para o circuito TP4.

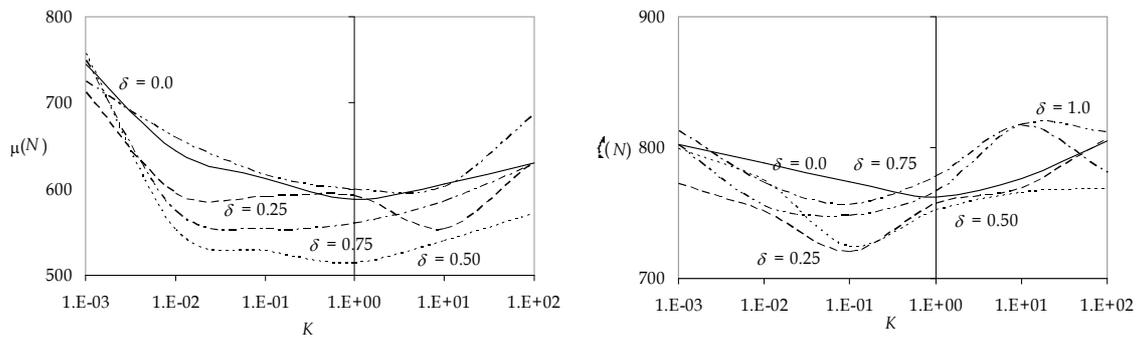


Figura 4. 27 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para $PI^{1/4}D^{1/4}$ versus $K = K_D = K_I$ para $\delta = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ com o conjunto *Gset 4*, para o circuito SOM1.

3.3. Planos de fase

Desenvolveu-se uma análise de desempenho dos AGs baseada no traçado do plano de fase, sendo a derivada da função de aptidão calculada aplicando a fórmula apresentada na equação 4.11.

$$DF \cong \frac{F_k - F_{k-1}}{\Delta t} \quad \text{com } \Delta t = 1 \quad (4.11)$$

Nestes gráficos decidi apresentar-se a derivada da média da função de aptidão *versus* a média da função de aptidão, o que indica a evolução da função de aptidão média dos indivíduos da população ao longo das gerações. No entanto, como a solução que o AG devolve é sempre baseada no melhor indivíduo da população, ilustra-se também um exemplo de traçado de um plano de fase com a derivada da melhor função de aptidão *versus* a melhor função de aptidão para o circuito SOM1, com o conjunto de portas lógicas *Gset 4*, usando F_e . Com esta técnica é possível comparar o grau de facilidade (dificuldade) do AG em atingir a solução.

Dada a natureza estocástica da evolução dos AGs, os planos de fases variam de experiência para experiência, pelo que uma generalização franca não é possível. No entanto, os gráficos apresentados indicam a dinâmica global para cada caso pelo que são indicativos da evolução do processo.

3.3.1. Com a função de aptidão estática (F_e)

As figuras 4.28 a 4.30 mostram os planos de fase da média da função de aptidão estática, com $\delta = 0$, para os conjuntos de portas lógicas *Gset 2* e *Gset 4*, para os circuitos M21, TP4 e SOM1.

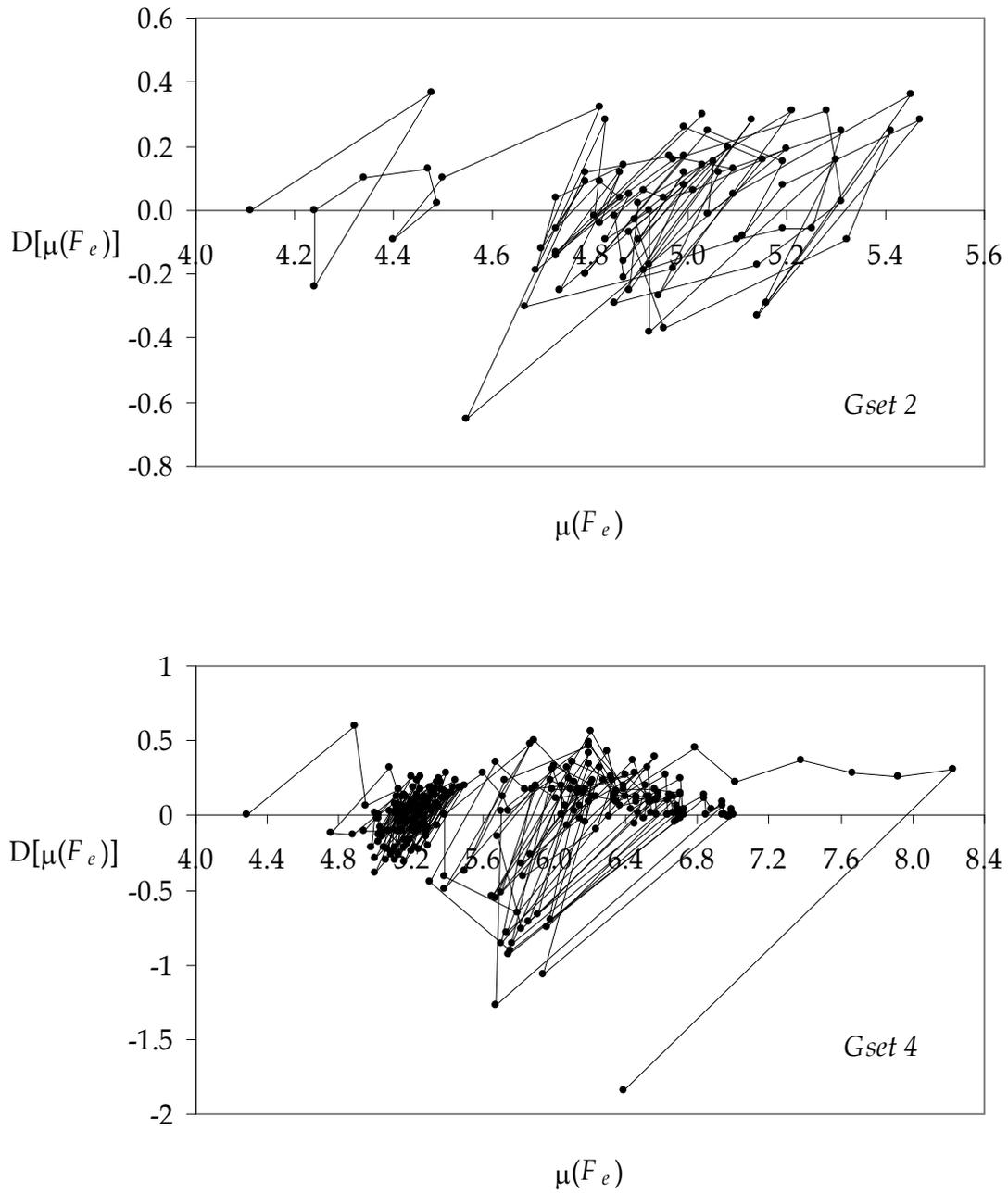


Figura 4. 28 - Plano de fase de uma execução do AG, com os conjuntos *Gset 2* e *Gset 4*, usando F_e , para o circuito M21.

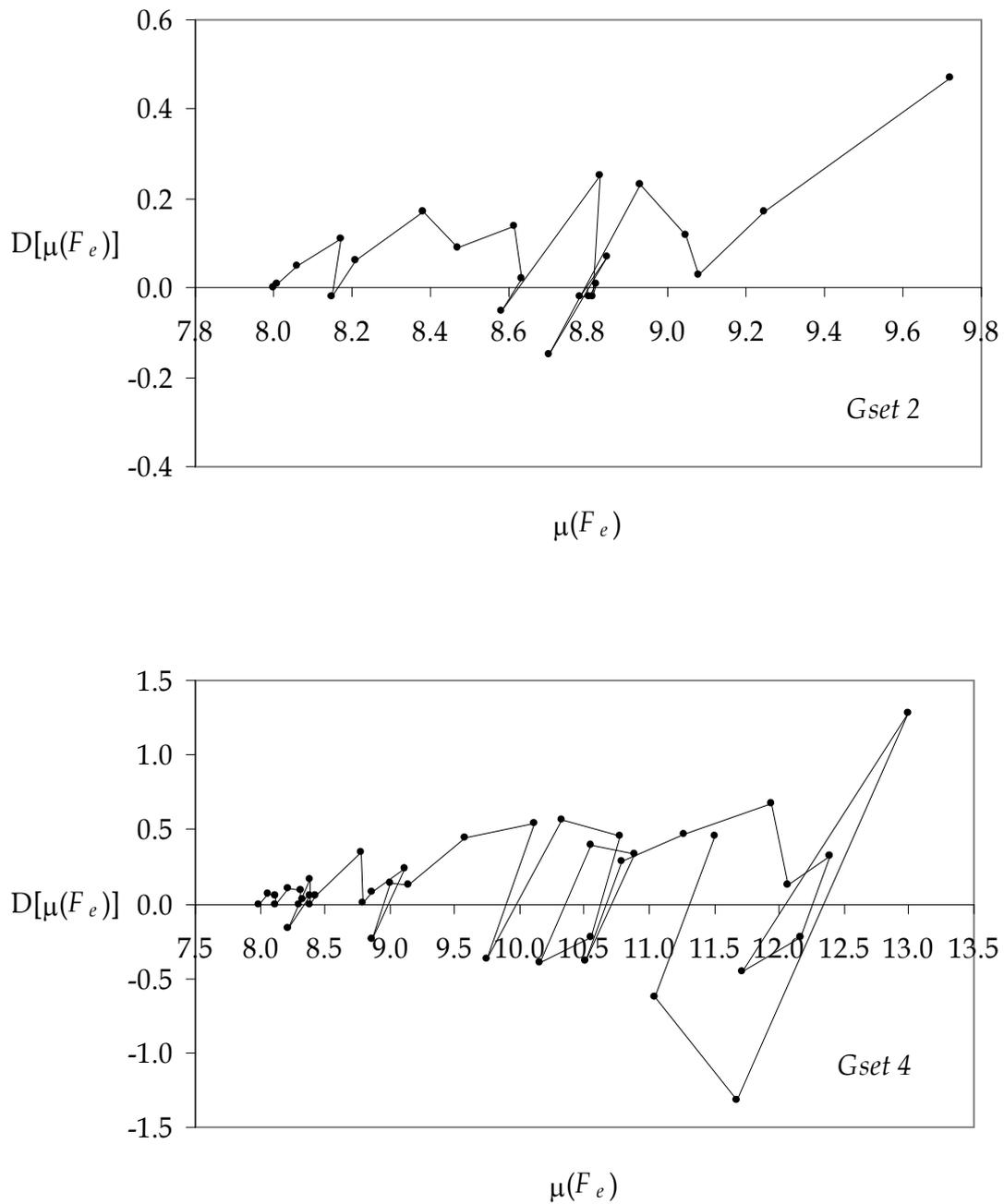


Figura 4. 29 - Plano de fase de uma execução do AG, com os conjuntos *Gset 2* e *Gset 4*, usando F_e , para o circuito TP4.

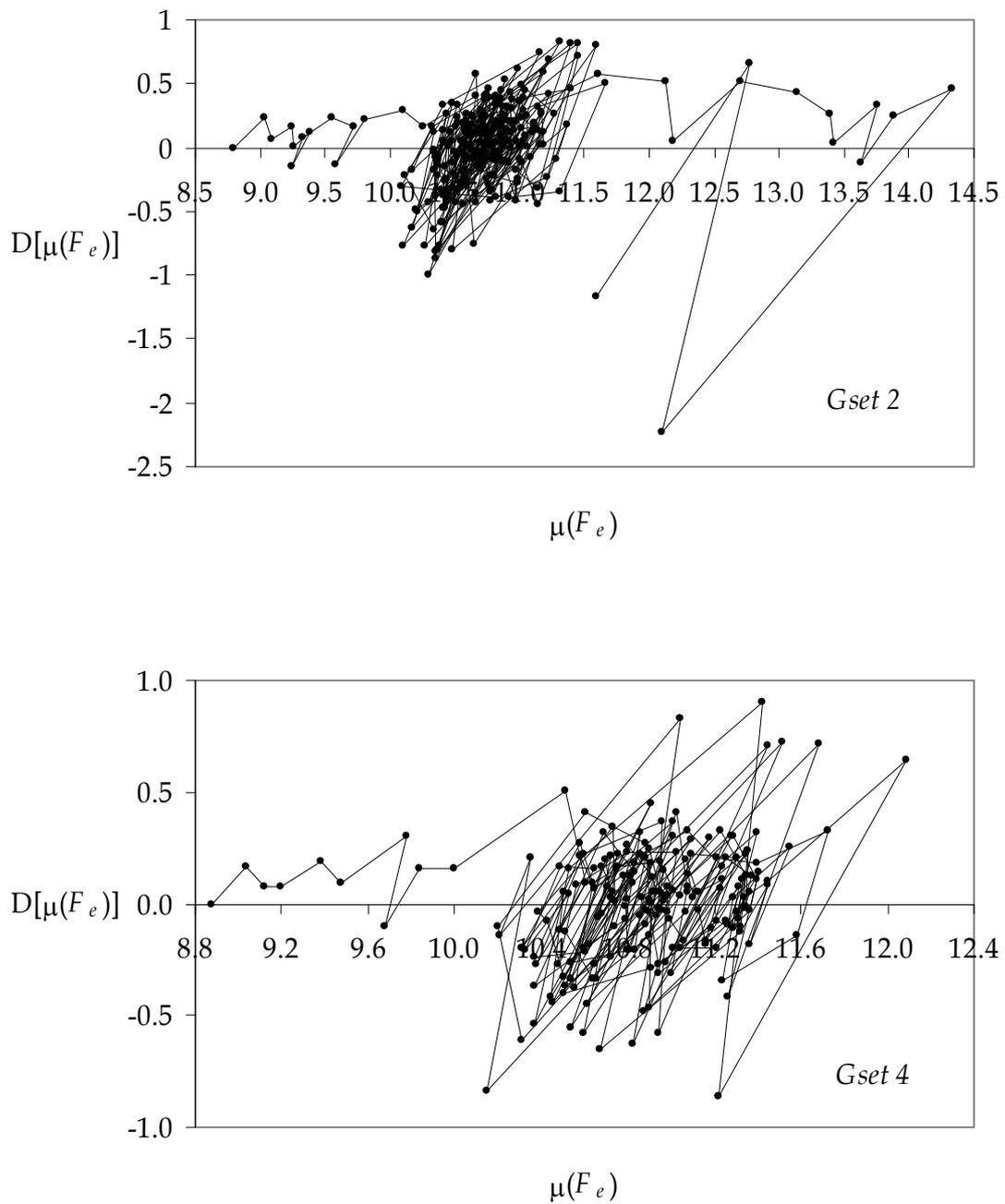


Figura 4. 30 - Plano de fase de uma execução do AG, com os conjuntos *Gset 2* e *Gset 4*, usando F_e , para o circuito SOM1.

Da análise dos gráficos das figuras 4.28 a 4.30 destacam-se as seguintes conclusões:

- O AG apresenta maior dificuldade de convergência na síntese dos circuitos M21, TP4 e SOM1 com o conjunto de portas lógicas $G_{set 4}$;
- O AG apresenta maior facilidade de convergência para o circuito TP4 e maior dificuldade para o circuito SOM1.

Como os traçados dos planos de fase, através da derivada da média da função de aptidão ou através da melhor função de aptidão são bastante distintos, ilustram-se nas figuras 4.31 e 4.32, respectivamente, os planos de fase da média e da melhor função de aptidão estática, com $\delta = 0$, para o conjunto de portas lógicas $G_{set 4}$, para o circuito SOM1, da mesma experiência.

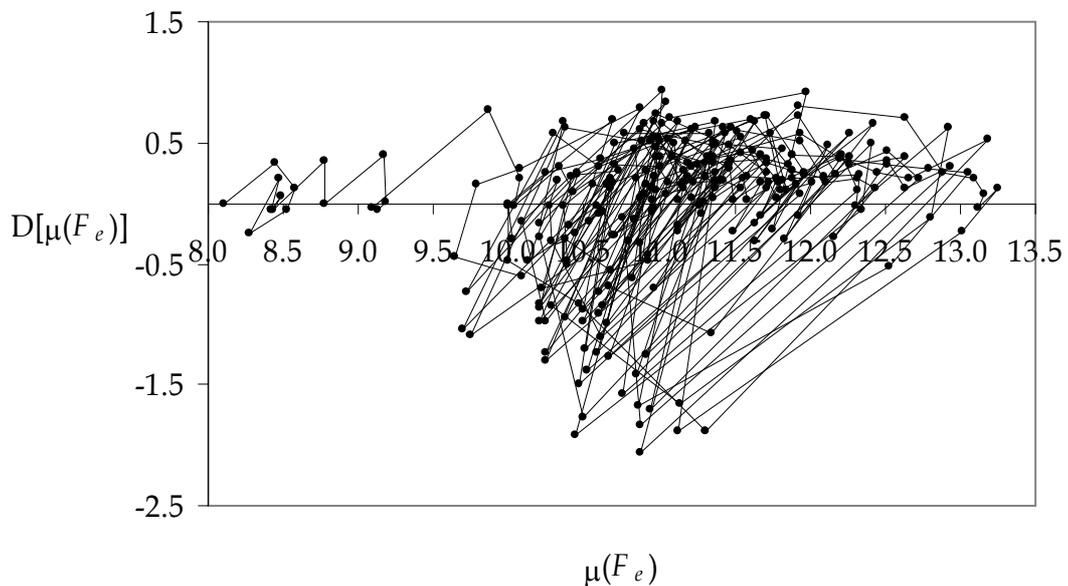


Figura 4. 31 - Plano de fase calculado com a derivada da média da função de aptidão de uma execução do AG, com o conjunto $G_{set 4}$, usando F_e , para o circuito SOM1.

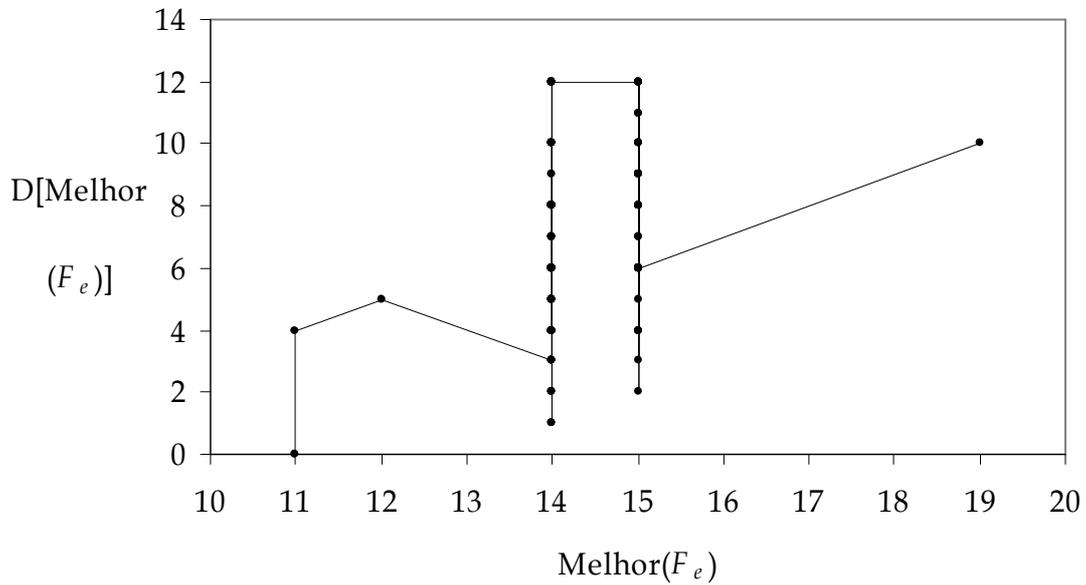


Figura 4. 32 - Plano de fase calculado com a derivada da melhor função de aptidão de uma execução do AG, com o conjunto *Gset 4*, usando F_e , para o circuito SOM1.

Analisando os gráficos das duas figuras anteriores e sabendo que correspondem à mesma experiência, verifica-se que na primeira iteração do AG a média da função de aptidão $\mu(F)$ apresenta um valor pouco superior a 8.0, evoluindo de seguida em dezassete iterações para um valor acima de 10.0, seguindo-se uma evolução mais lenta até atingir a solução. Relativamente à melhor função de aptidão, na primeira iteração apresenta um valor de 11, evoluindo rapidamente para 14, mantendo-se entre 14 e 15 durante várias iterações até atingir a solução com um valor de $F_e = 19$.

3.3.2. Com a função de aptidão dinâmica (F_d)

As figuras 4.33 a 4.35 mostram os planos de fase da média da função de aptidão

dinâmica para o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para os conjuntos de portas lógicas $Gset 2$ e $Gset 4$, para os circuitos M21, TP4 e SOM1.

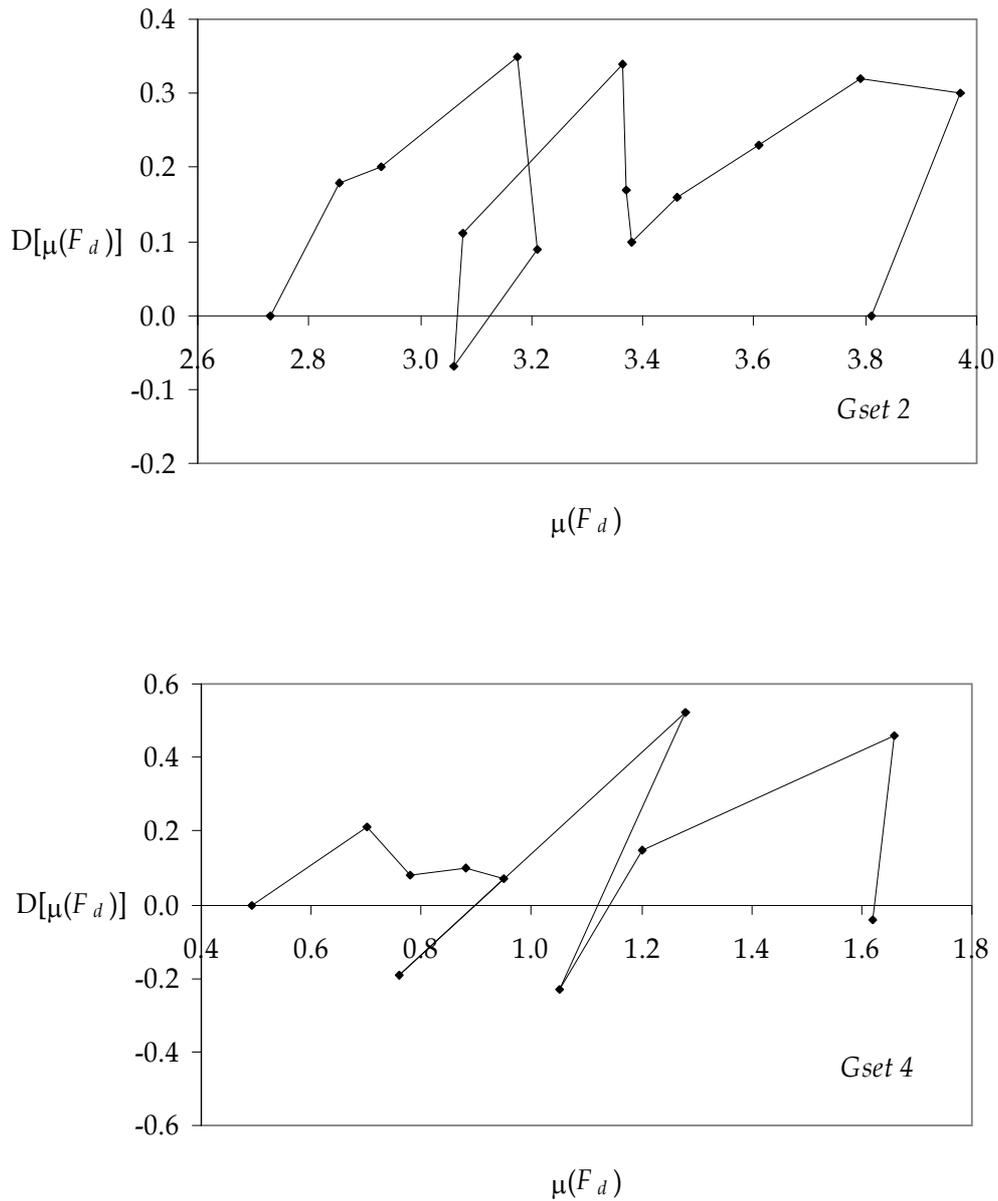


Figura 4. 33 - Plano de fase de uma execução do AG, com os conjuntos $Gset 2$ e $Gset 4$, usando F_d com o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para o circuito M21.

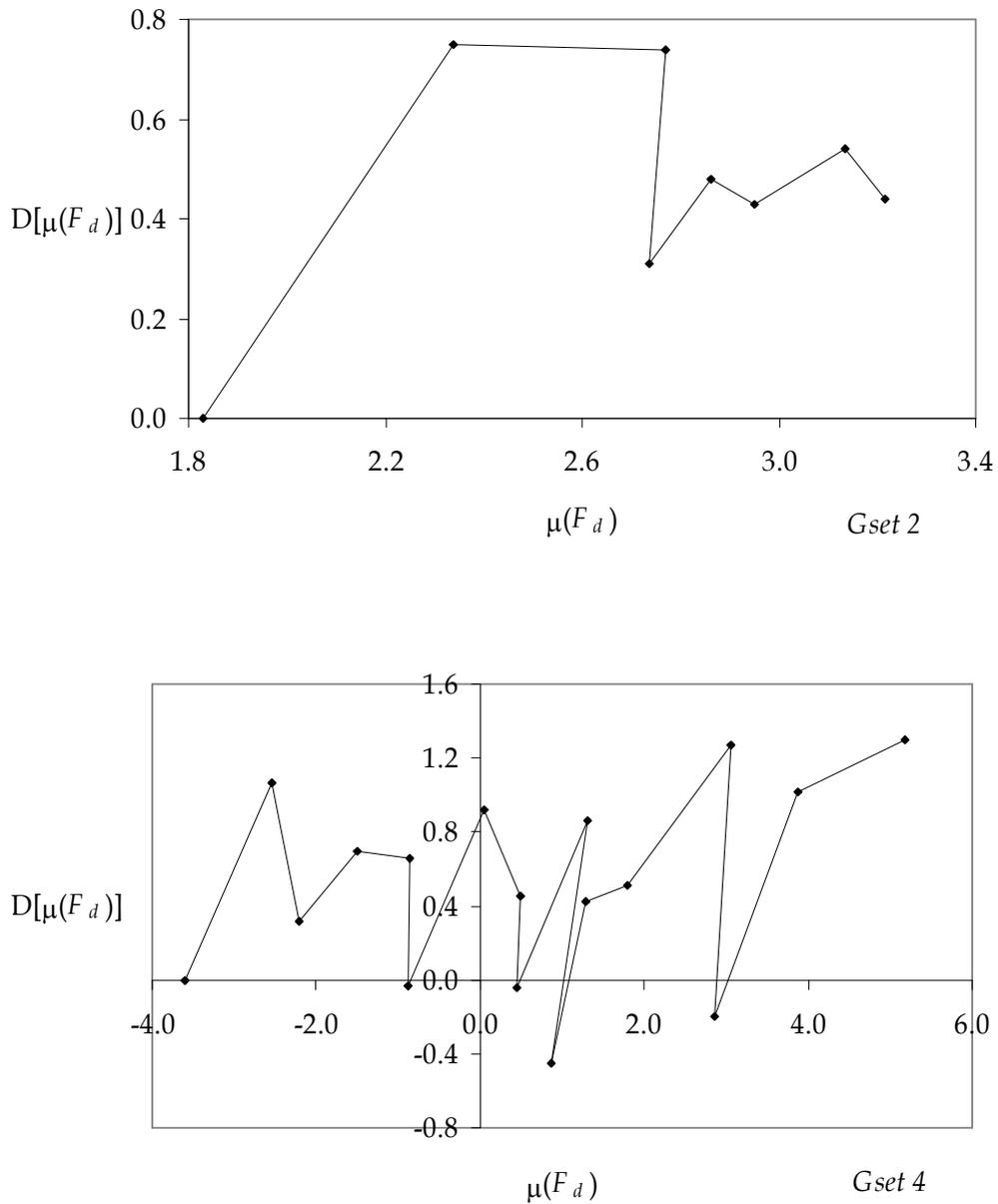


Figura 4. 34 - Plano de fase de uma execução do AG, com os conjuntos *Gset 2* e *Gset 4*, usando F_d com o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para o circuito TP4.

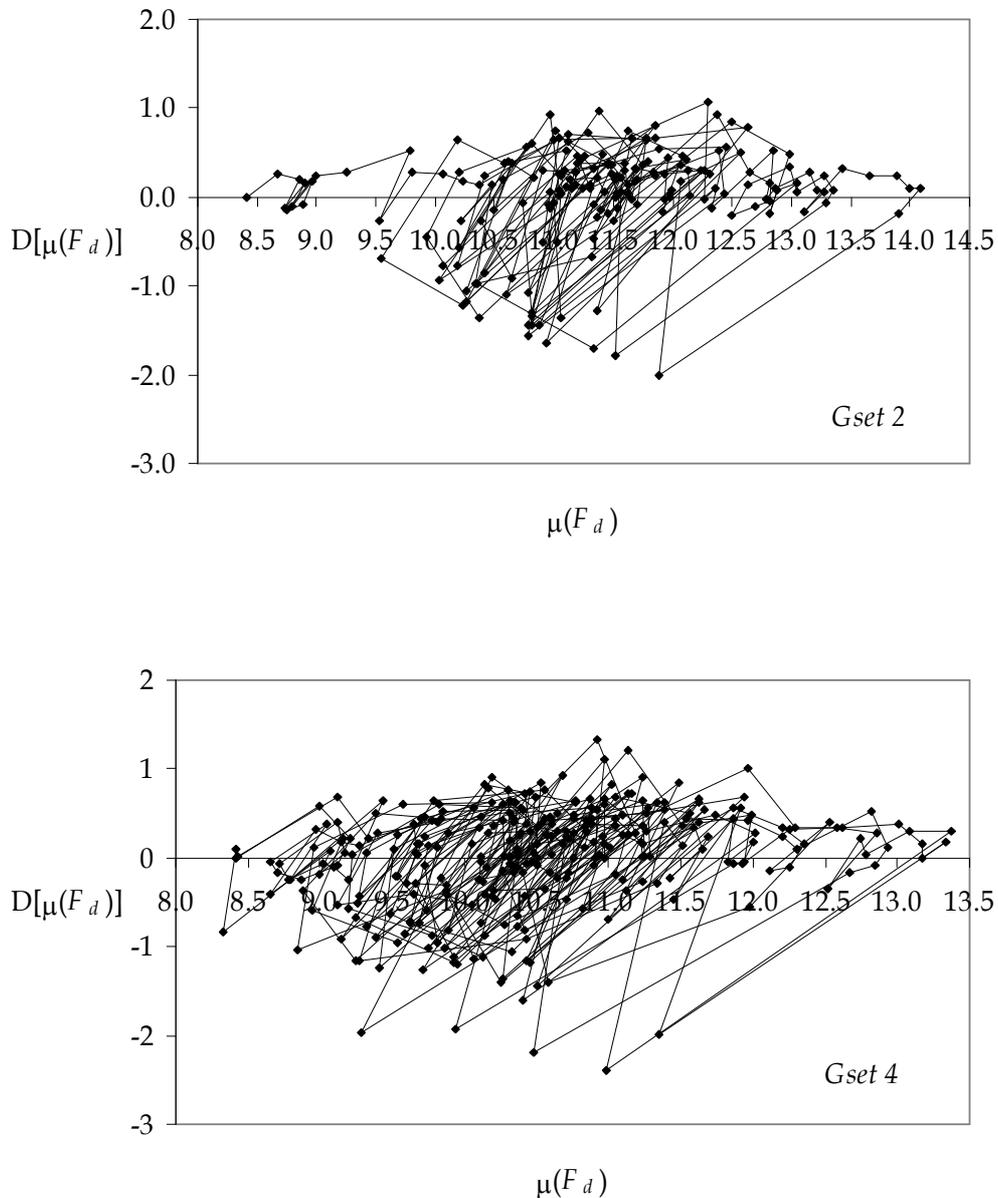


Figura 4. 35 - Plano de fase de uma execução do AG, com os conjuntos *Gset 2* e *Gset 4*, usando F_d com o esquema $PI^{1/4}D^{1/4}$ com $K = K_D = K_I = 1$ e $\delta = 0.50$, para o circuito SOM1.

Comparando os gráficos dos planos de fase desta secção com os gráficos da secção 3.3.1 verifica-se que, com a adopção da função de aptidão dinâmica, o AG apresenta melhor capacidade de convergência especialmente na síntese dos

circuitos M21 e TP4. Constata-se também que o AG converge melhor com o conjunto de portas lógicas *Gset 2*.

3.4. Outros circuitos

Nesta secção são tratados e analisados três novos circuitos, nomeadamente o circuito de teste de paridade de cinco *bits* (TP5), o subtrator completo de um *bit* (SUB1) e o multiplicador de dois *bits* (MUL2), com as seguintes características:

- O circuito TP5, tem cinco entradas $\mathbf{X} = \{A_4, A_3, A_2, A_1, A_0\}$ e uma saída $\mathbf{Y}_R = \{P\}$. O tamanho da matriz é $l \times c = 5 \times 5$ e o comprimento do vector que representa cada um dos circuitos (*i. e.*, o tamanho do cromossoma) é $TC = 75$.
- O circuito SUB1, tem três entradas $\mathbf{X} = \{A, B, B_{in}\}$ e duas saídas $\mathbf{Y}_R = \{S, B_{out}\}$. O tamanho da matriz é $l \times c = 3 \times 3$ e o comprimento do vector que representa cada um dos circuitos (*i. e.*, o tamanho do cromossoma) é $TC = 27$.
- O circuito MUL2, tem quatro entradas $\{A_1, A_0, B_1, B_0\}$ e quatro saídas $\{C_3, C_2, C_1, C_0\}$. A matriz correspondente é de dimensão $l \times c = 4 \times 4$ e o cromossoma de tamanho $TC = 48$.

As experiências foram realizadas com os conjuntos de portas lógicas *Gset 2* e *Gset 4*, com os parâmetros $TC = 95\%$, $TM = 20\%$ e $P = 1000$ circuitos, usando os esquemas de avaliação da função de aptidão descritos anteriormente.

As figuras 4.36 a 4.41 ilustram $\mu(N)$ e $\sigma(N)$ para os dois conjuntos de portas lógicas quando se aplicam os diferentes esquemas da função de aptidão,

$\delta = 0.5$, $\mu = 0.5$ e $K = 0.5$ para os circuitos TP5, SUB1 e MUL2. Uma vez mais, os gráficos dos resultados demonstram um melhoramento notável quando se adopta os novos conceitos propostos.

Os diagramas esquemáticos destes três novos circuitos, podem ser visualizados nas figuras 4.42 a 4.44.

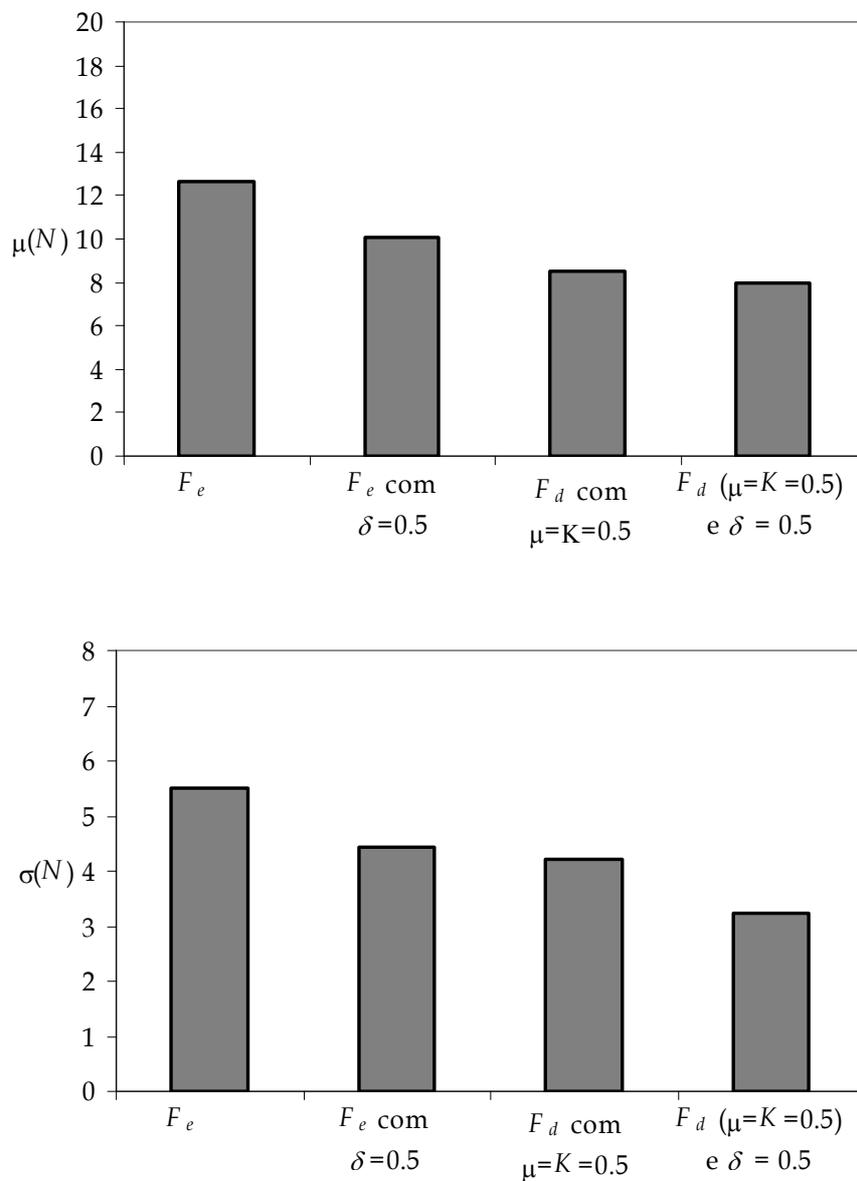


Figura 4. 36 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto *Gset 2*, para o circuito TP5.

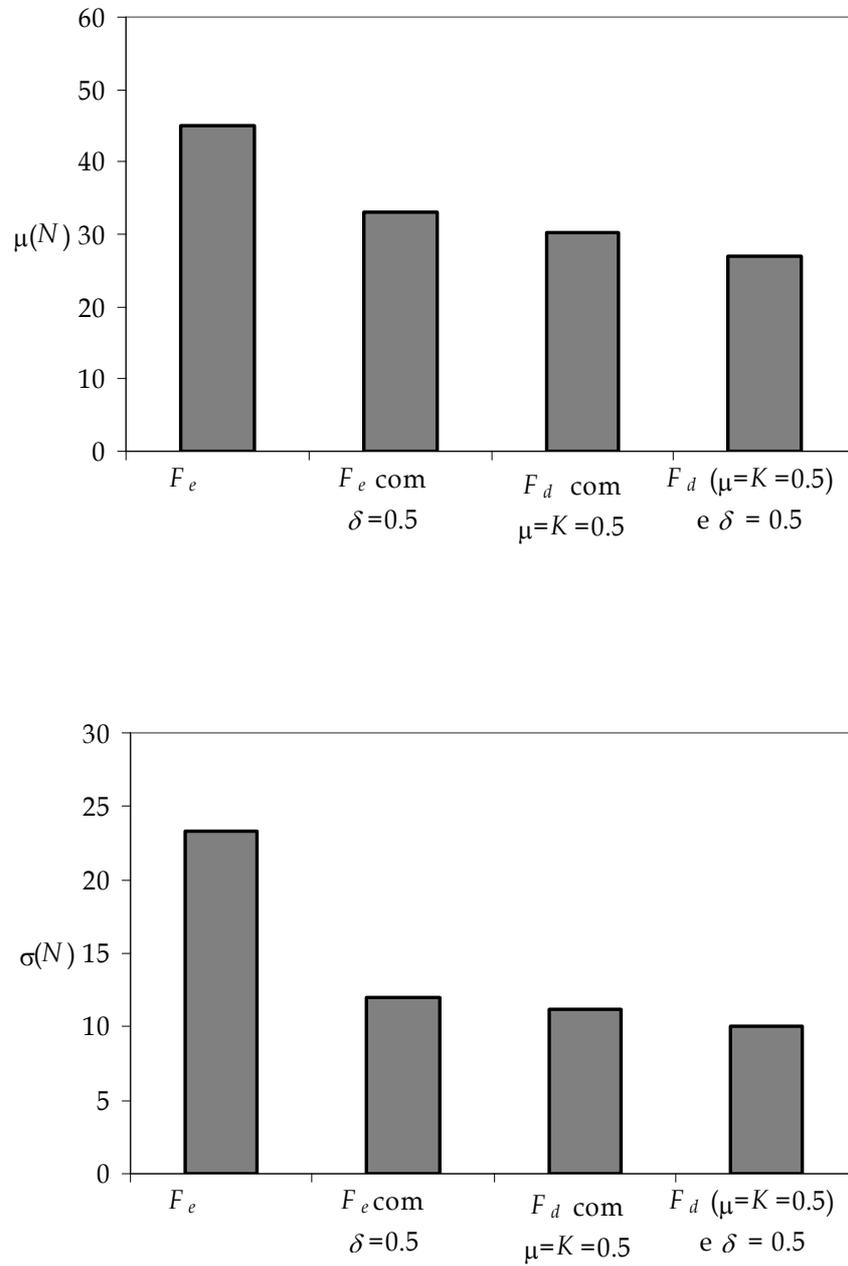


Figura 4. 37 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto *Gset 4*, para o circuito TP5.

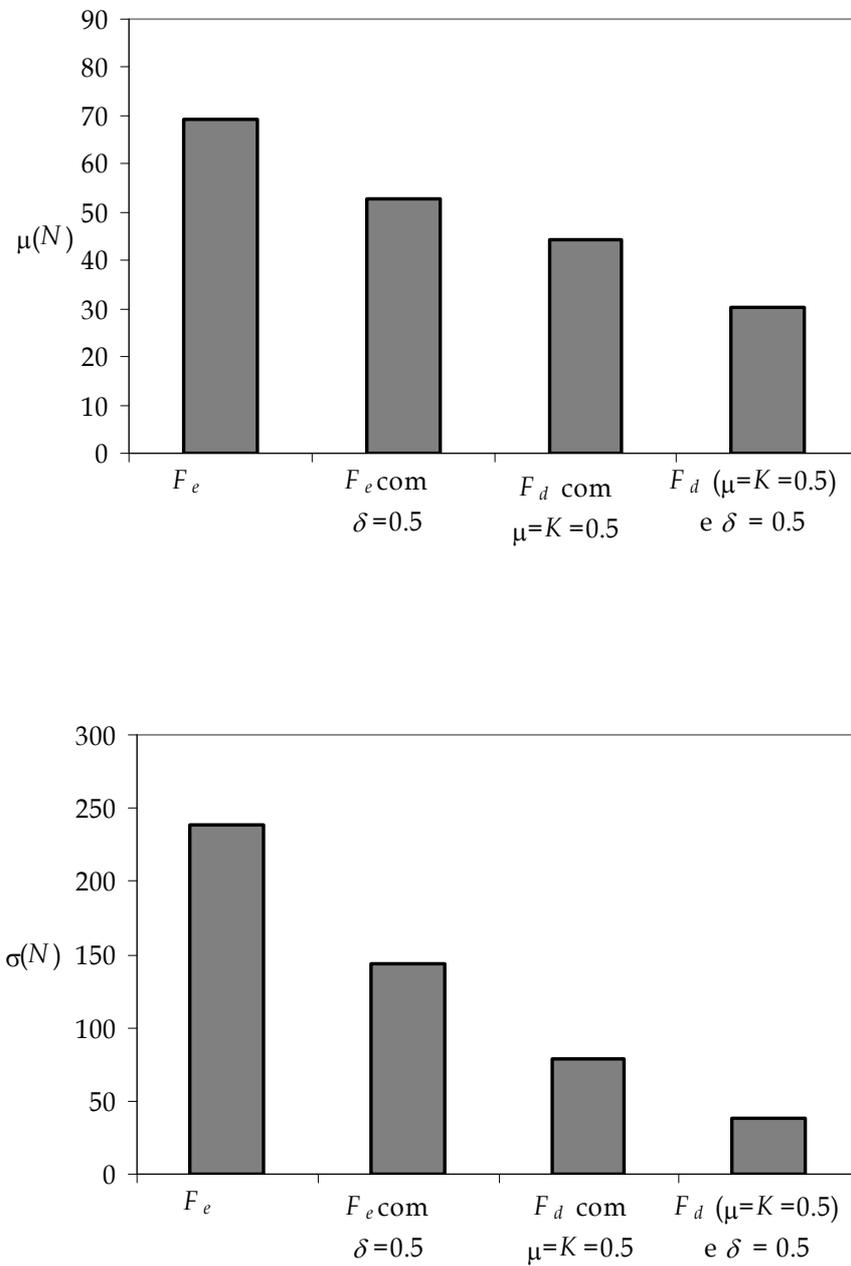


Figura 4. 38 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto *Gset 2*, para o circuito SUB1.

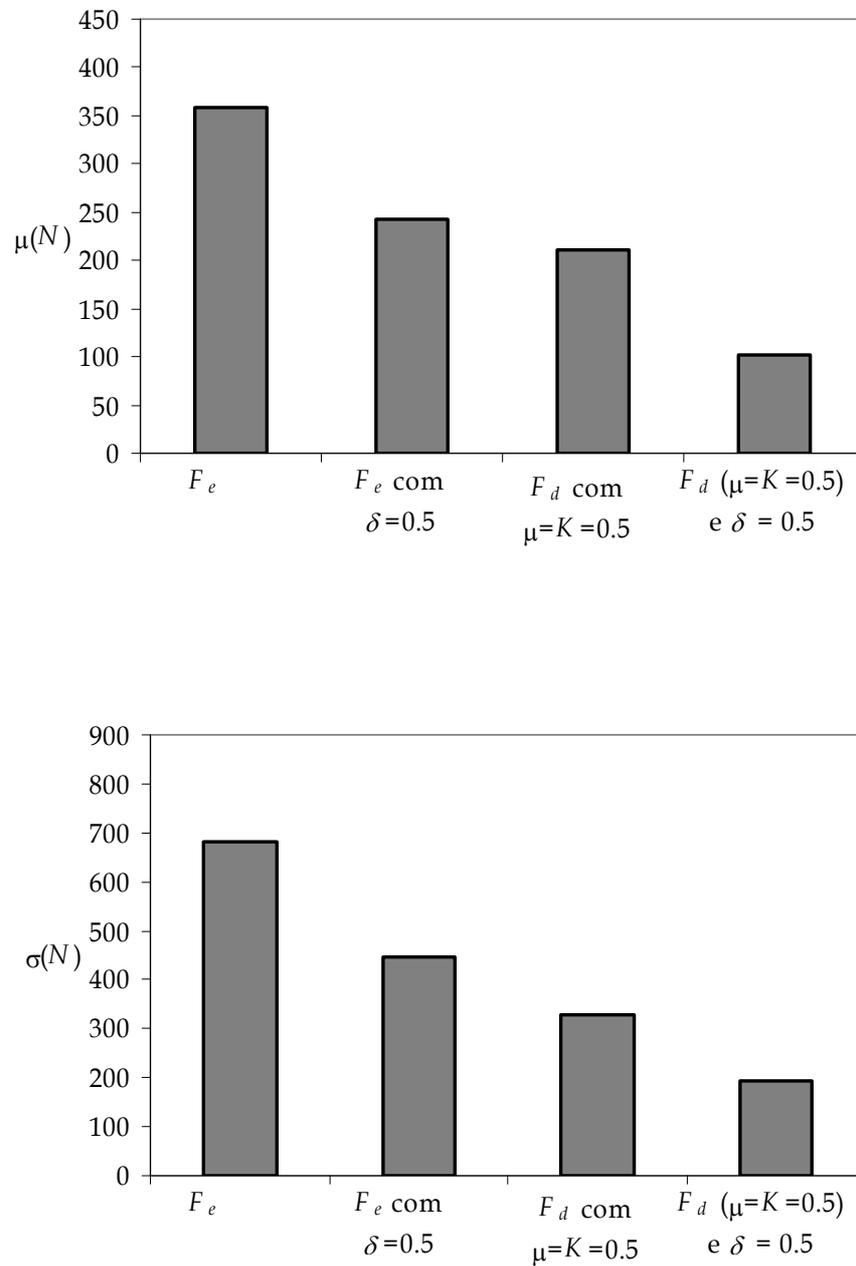


Figura 4. 39 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto *Gset 4*, para o circuito SUB1.

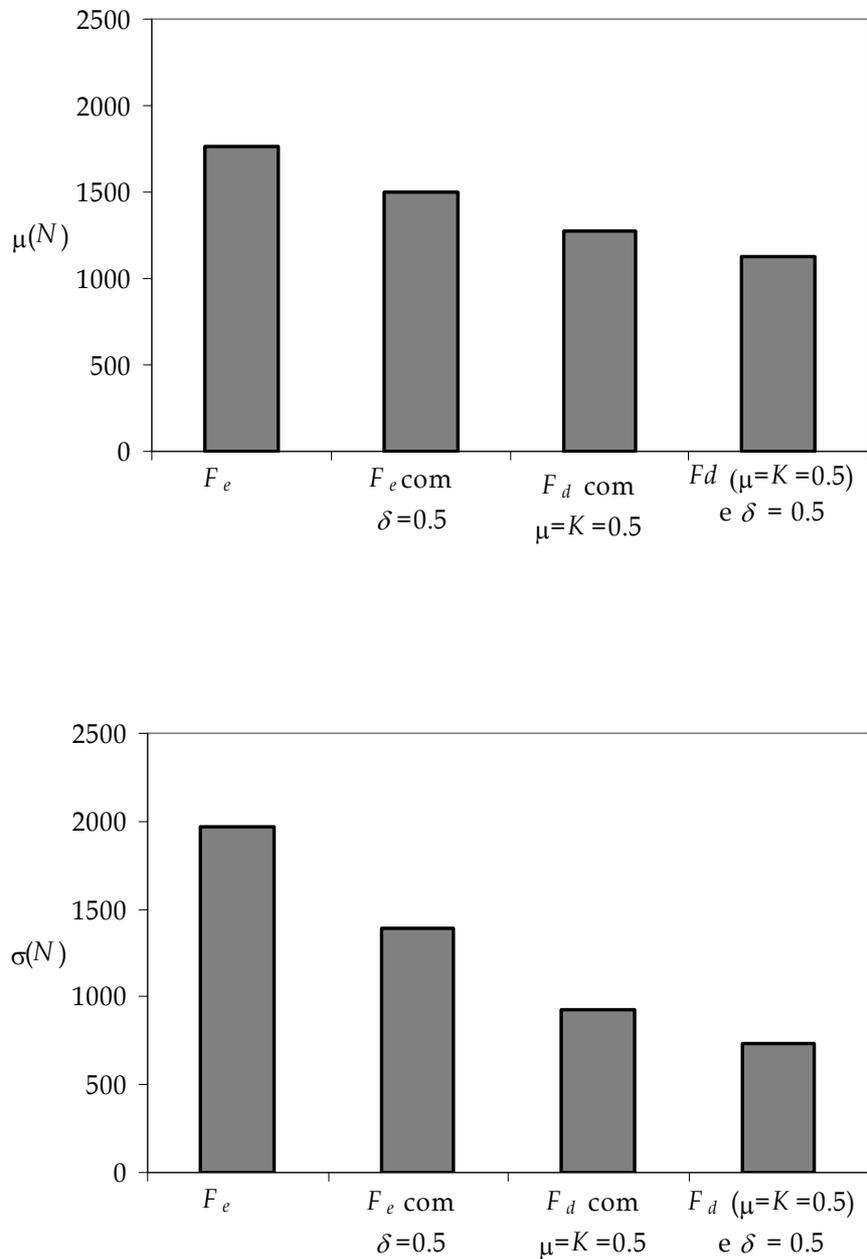


Figura 4. 40 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto *Gset 2*, para o circuito MUL2.

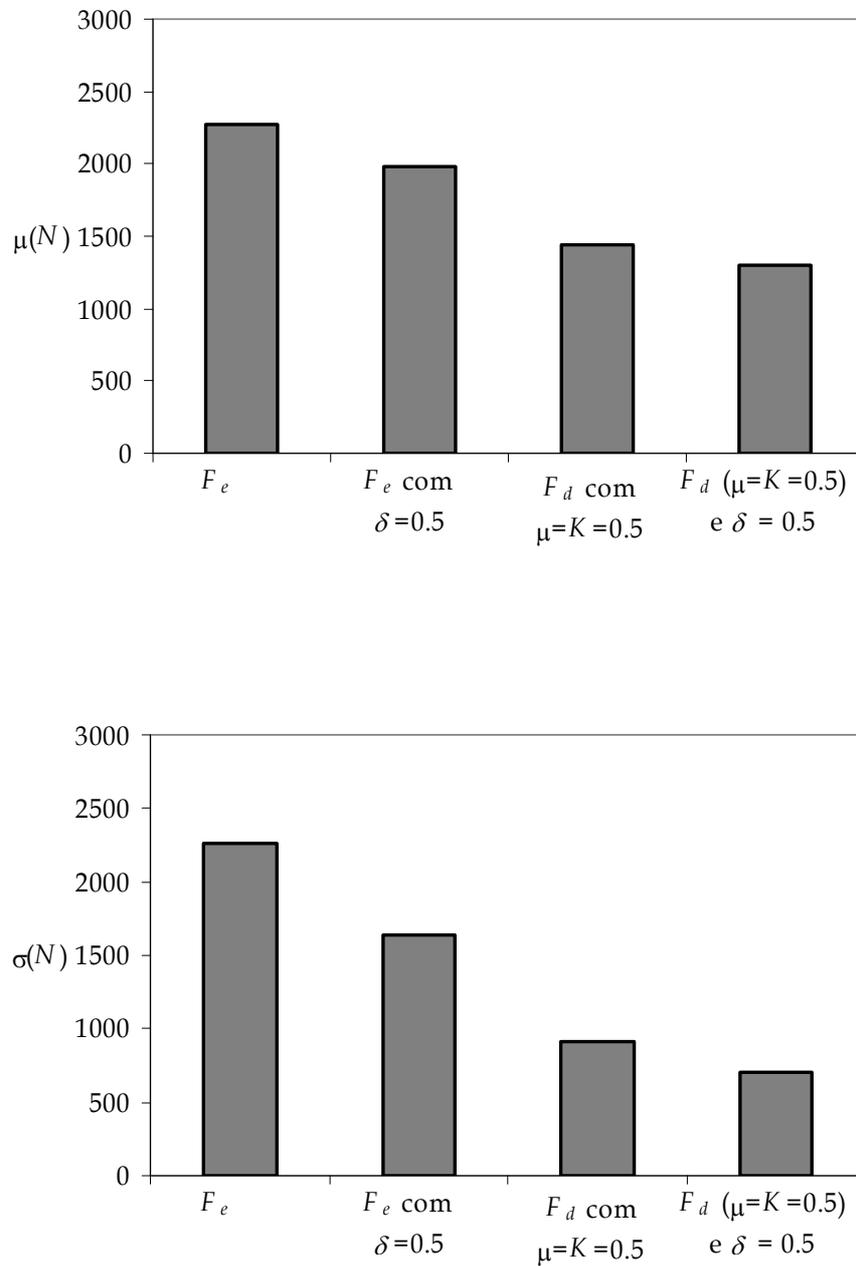


Figura 4. 41 – Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução para com o conjunto *Gset 4*, para o circuito MUL2.

Observando os gráficos das figuras 4.36 a 4.41 é inequívoca a superioridade do AG com a função de aptidão dinâmica no esquema $PD^{1/2}$ com $k = 0,5$ e

medida da descontinuidade do erro com $\delta=0,5$, para os circuitos TP5, SUB1 e MUL2, com os conjuntos de portas lógicas $G_{set 2}$ e $G_{set 4}$.

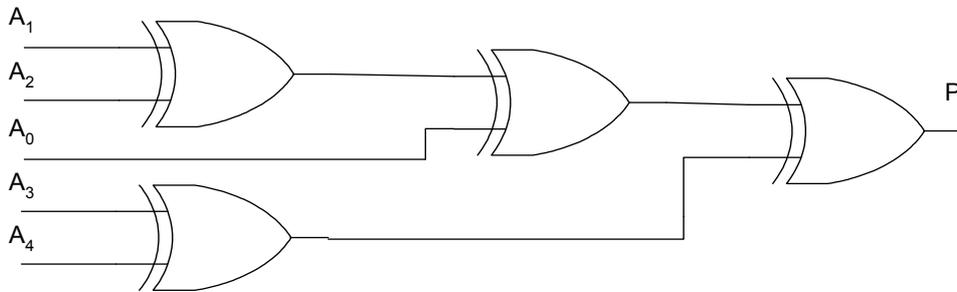


Figura 4. 42 - Circuito TP5 gerado pelo AG.

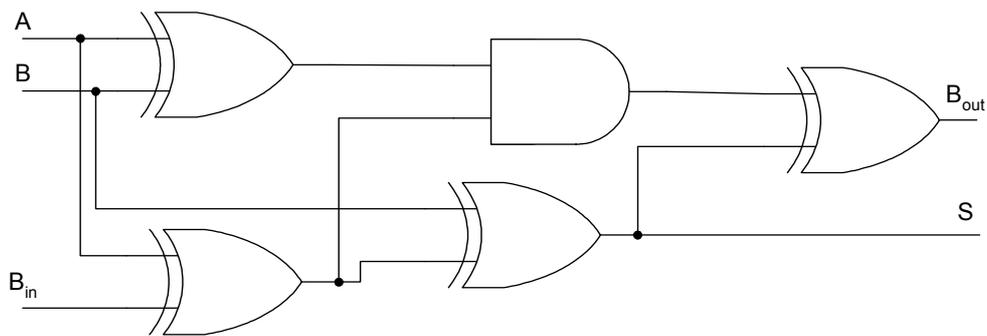


Figura 4. 43 - Circuito SUB1 gerado pelo AG.

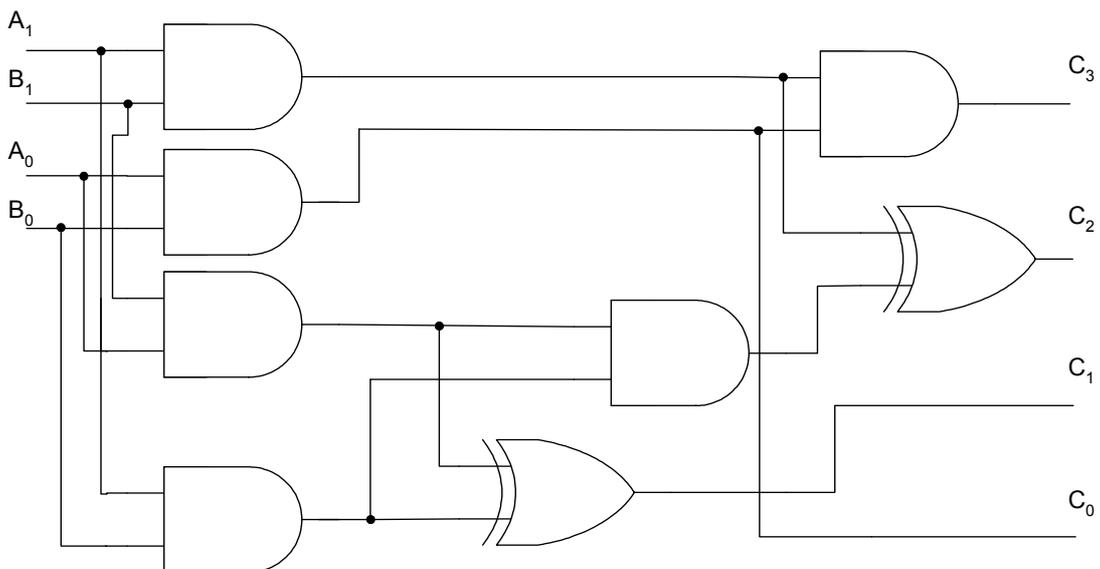


Figura 4. 44 - Circuito MUL 2 gerado pelo AG.

3.5. Calculando F_d usando uma aproximação de Padé

Outra forma de calcular a função de aptidão dinâmica F_d , consiste em utilizar uma aproximação de Padé (em fracção racional) da série (4.3) resultante da fórmula de Euler:

$$\begin{aligned}
 D^\mu(z) &= \left(\frac{1}{T}\right)^\mu \text{Padé}\left\{(1-z^{-1})^\mu\right\}_{m,n} & (4.12) \\
 &= \left(\frac{1}{T}\right)^\mu \frac{P_m(z^{-1})}{Q_n(z^{-1})} \\
 &= \left(\frac{1}{T}\right)^\mu \frac{p_0 + p_1 z^{-1} + \Lambda + p_m z^{-m}}{q_0 + q_1 z^{-1} + \Lambda + q_m z^{-n}}
 \end{aligned}$$

onde $m, n \in \mathbb{N}$ são as ordens dos polinómios P e Q e z^{-1} representa a amostragem de tempo discreta.

Neste conjunto de experiências utilizaram-se $n = 100$ simulações para cada um dos casos testados. As experiências consistiram na síntese de três circuitos lógicos combinatórios, o multiplexador dois para um (M21), o circuito de teste de paridade de 4 bits (TP4) e o circuito somador completo de um bit (SOM1), usando a função de aptidão dinâmica apresentada na equação 4.9. Usaram-se dois conjuntos de portas lógicas, o *Gset 2* e o *Gset 4*, com $CR = 95\%$, $MR = 20\%$ e $P = 100$.

Na implementação dos operadores de ordem fraccionária adoptou-se a equação 4.12 com $m = n = 4$ e $T = 1$ s.

Em virtude de existir um largo número de combinações possíveis dos parâmetros do AG, os resultados que se apresentam a seguir encontram-se limitados a um conjunto restrito de casos. Por este motivo e *a priori*, outros

valores podem dar origem a resultados diferentes. No entanto, desenvolveu-se um número adicional de experiências numéricas que permitiram concluir que os casos tratados são representativos.

Analisou-se o desempenho do AG, com adopção da função de aptidão incluindo o esquema dinâmico. As simulações investigaram o esquema diferencial $\mu = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ de F_d para ganhos da gama $K \in [0, 1]$.

Da figura 4.45 à figura 4.50 pode constatar-se a média do número de gerações necessário para obtenção da solução $\mu(N)$ e o respectivo desvio padrão $\sigma(N)$ versus K , com F_d , para os circuitos M21, TP4 e SOM1, usando os conjuntos de portas lógicas *Gset 2* e *Gset 4*. Para melhor comparação dos resultados, os gráficos incluem o caso $\mu = 0$, que corresponde à função de aptidão estática F_e .

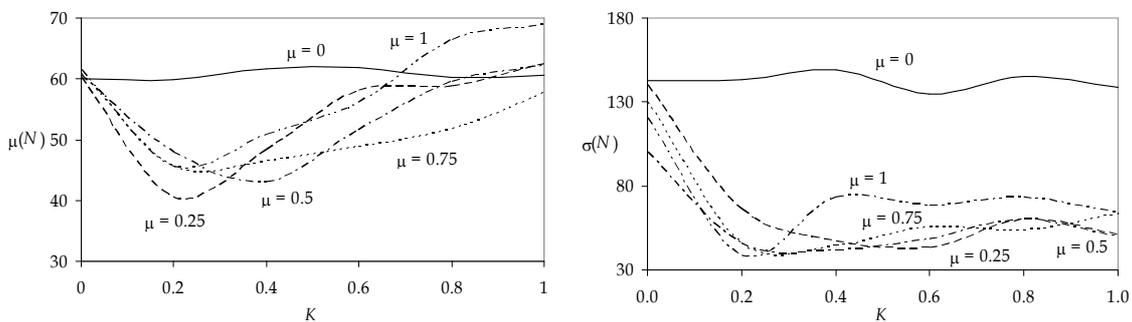


Figura 4. 45 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto *Gset 2*, para o circuito M21.

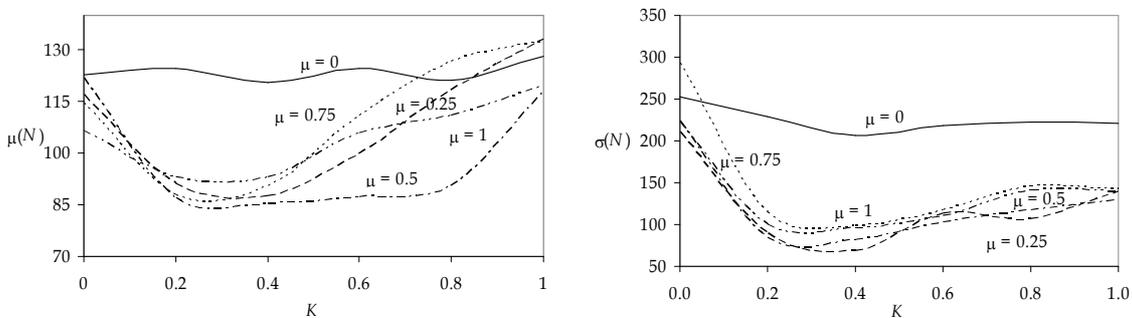


Figura 4. 46 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto *Gset 4*, para o circuito M21.

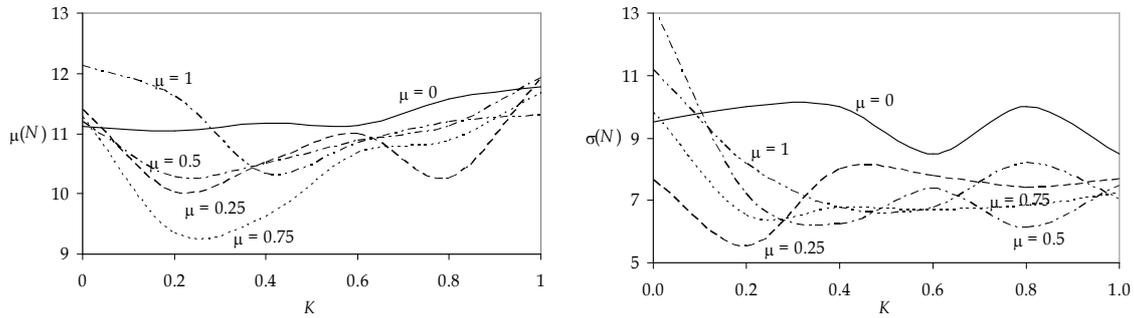


Figura 4. 47 - Média do número de gerações para obter a solução $\mu(N)$ e desvio padrão $\sigma(N)$ com o conjunto *Gset 2*, para o circuito TP4.

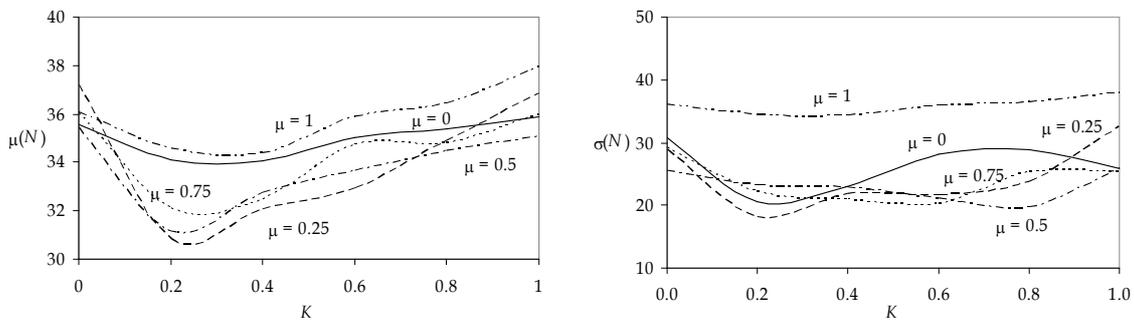


Figura 4. 48 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ p do número de gerações para obter a solução com o conjunto *Gset 4*, para o circuito TP4.

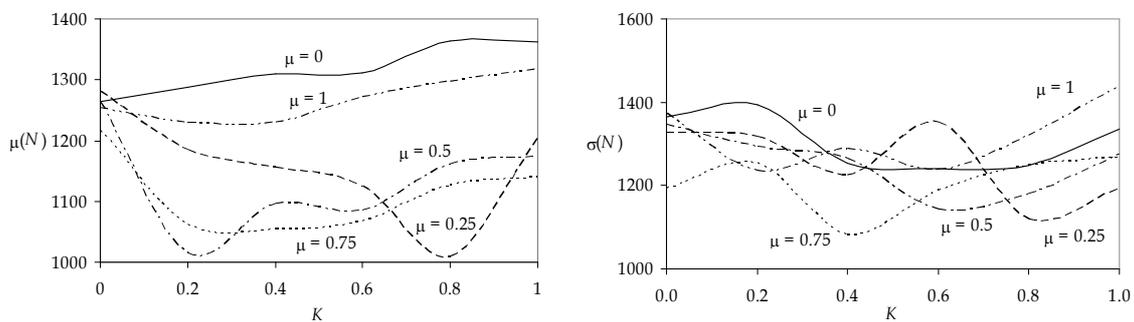


Figura 4. 49 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto *Gset 2*, para o circuito SOM1.

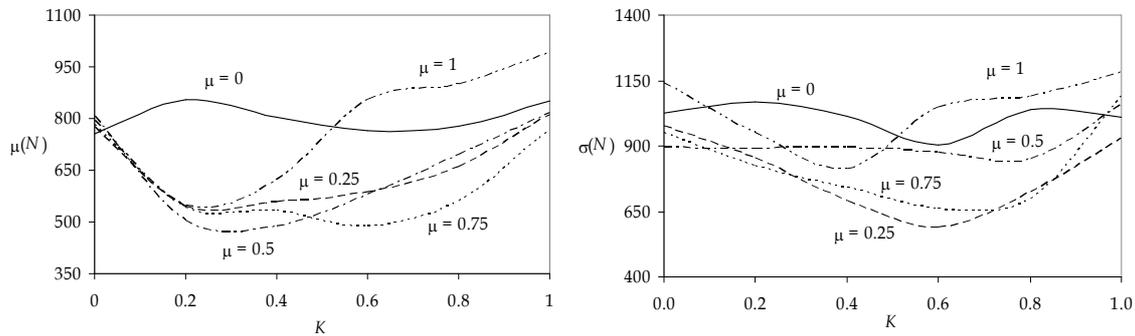


Figura 4. 50 - Média $\mu(N)$ e desvio padrão $\sigma(N)$ do número de gerações para obter a solução com o conjunto *Gset 4*, para o circuito SOM1.

Observando os gráficos anteriores, verifica-se que a modificação do conceito da função de aptidão padrão com a inclusão dos efeitos dinâmicos, melhora significativamente o desempenho do AG (Reis, Machado e Cunha, 2005a, 2005b, 2005c) quando se usa a série de Taylor. Aplicando a aproximação de Padé, obtêm-se resultados similares para $\mu(N)$ e $\sigma(N)$ com um esquema computacional mais eficiente em termos de tempos de execução do AG (Reis *et. al.*, 2006).

4. Resumo do capítulo

O presente capítulo aborda o desenvolvimento e análise de resultados de novos conceitos na implementação da função de aptidão que avalia os circuitos gerados por um AG. São propostos dois novos conceitos, a saber:

- Função de aptidão estática com medida da descontinuidade do erro;

■ Função de aptidão dinâmica.

Ambas as abordagens permitiram melhores resultados face aos obtidos com a função de aptidão estática, nomeadamente no que diz respeito ao número de gerações necessárias para a obtenção da solução.

Foram também estudados os dois conceitos aplicados em conjunto, isto é a função de aptidão dinâmica com a inclusão da medida da descontinuidade do erro na função de aptidão. Com esta combinação obtiveram-se resultados superiores aos obtidos quando se utilizam separadamente.

O traçado dos planos de fase revelou ser uma ferramenta de análise da convergência dos algoritmos genéticos.

Os resultados obtidos neste capítulo incentivam a aplicação destes novos conceitos da função de aptidão nos algoritmos da computação evolutiva.

Referências

- Chen, Y. Q. e Moore, K. L., Discretization schemes for fractional-order differentiators and integrators. *IEEE Trans. On Circuits and Systems*, vol. 49, pp 363-367, March 2002.
- Koh, C. G. e Kelly, J. M. Application of fractional derivatives to seismic analysis of base-isolated models. *Earthquake Engineering and Structural Dynamics*, vol.19, pp.229-241, 1990.
- Méhauté, A. L., *Fractal Geometries: Theory and Applications*. Penton Press, London, 1991.
- Miller, K. S. e Ross, B. *An Introduction to the Fractional Calculus and Fractional Differential Equations*. John Wiley & Sons, New York, 1993.
- Oldham, K. B. e Spanier, J. *The Fractional Calculus: Theory and Application of Differentiation and Integration to Arbitrary Order*. Academic Press, New York, 1974.
- Oustaloup, A. *La Dérivation Non Entier: Théorie, Synthèse et Applications*. Editions Hermes, 1995.
- Reis, Cecília, Tenreiro Machado, J. A. e Boaventura Cunha, J. Evolutionary Design of Combinational Logic Circuits, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Fuji Technology Press, Vol. 8, No. 5, pp. 507-513, September, 2004.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Fractional Dynamic Fitness Functions for GA-based Circuit Design, *Proceedings of GECCO 2005 - Genetic and Evolutionary Computation Conference*, Washington, DC, USA, pp. 1571-1572, June, 2005a.

- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Digital Circuit Design Using Dynamic Fitness Functions, LBP of GECCO 2005 - Genetic and Evolutionary Computation Conference, Washington, DC, USA, June, 2005b.
- Reis, C., J. Machado, e Cunha, J. Evolutionary design of combinational circuits using fractional-order fitness. In Proceedings of the Fifth EUROMECH Nonlinear Dynamics Conference, pages 1312-1321, August, 2005c.
- Reis, C., Tenreiro Machado, J. A., Boaventura Cunha, J. e Figueiredo, L. Fractional-Order Evolutionary Design Of Digital Circuits, FDA'06 - 2nd IFAC Workshop on Fractional Differentiation and its Applications, Porto, Portugal, pp 445-450, July, 2006.
- Sano, Y. e Kita, H.. Optimization of Noisy Fitness Functions by means of Genetic Algorithms using History of Search. In Proceedings of PPSN VI, pp. 571-581, 2000.
- Sano, Y. e Kita, H.. Optimization of Noisy Fitness Functions by means of Genetic Algorithms using History of Search with test of Estimation. In Proceedings of CEC, 2002.
- Tenreiro Machado, J. A. Analysis and Design of Fractional-Order Digital Control Systems. SAMS Journal Systems Analysis, Modelling, Simulation, vol. 27: 107-122, 1997.
- Torvik, P. J. e Bagley, R. L. On the Appearance of the Fractional Derivative in the Behaviour of real materials. ASME Journal of Applied Mechanics, vol. 51, pp. 294-298, June 1984.
- Westerlund, S. Dead Matter Has Memory! Causal Consulting. Sweden: Kalmar, 2002.

Capítulo 5

SÍNTESE DE CIRCUITOS COM ALGORITMOS MEMÉTICOS

Introdução

O presente capítulo propõe um algoritmo genético híbrido, conhecido também por Algoritmo Memético (AM) (Moscato, 1989), aplicado ao projecto de circuitos lógicos combinatórios. Sabendo que os algoritmos híbridos são bastante eficientes na resolução de problemas difíceis de optimização combinatoria, sugere-se um AM que combina um AG para síntese de circuitos digitais, com um algoritmo de pesquisa local do tipo de porta (APLTP). A conjugação de estratégias que incluem pesquisas globais e locais são adoptadas nas recentes abordagens de optimização híbridas. A ideia fundamental é a de aplicar um refinamento local a algoritmos evolutivos, conseguindo melhorar as funções de aptidão dos indivíduos da população.

Nesta ordem de ideias, a secção 1 introduz o AM em termos de conceitos essenciais e o algoritmo de pesquisa local proposto. A secção 2 dedica-se à implementação dos circuitos combinatórios. A secção 3 aborda o problema de

convergência e a secção 4 compara os algoritmos memético e genético. Por último, a secção 5 faz a síntese conclusiva.

1. O Algoritmo Memético

Nas experiências realizadas e descritas no capítulo 3, relativas à síntese de circuitos lógicos com AGs, verifica-se que os indivíduos da população tendem a ficar semelhantes. Este facto pode levar a que o algoritmo tenha dificuldade em convergir ou a que tenha uma convergência prematura. Por essa razão surgiu a ideia de adoptar um algoritmo memético (AM) no projecto dos circuitos digitais.

O AM é um algoritmo evolutivo que inclui uma fase de optimização individual, como parte da estratégia de pesquisa, sendo a optimização individual implementada como uma pesquisa local.

Os AMs inspiram-se nos modelos de adaptação dos sistemas da natureza que combinam a adaptação evolutiva dos indivíduos das populações com a aprendizagem individual ao longo da vida (Krasnogor, 2002). Os AMs são meta-heurísticas que tiram partido dos operadores evolutivos na determinação de regiões de interesse do espaço de pesquisa. Além disso, os AMs incluem a pesquisa local, o que lhes permite encontrar rapidamente soluções boas em regiões pequenas do espaço de pesquisa (Merz e Freisleben 1999). Os AMs fundamentam-se no conceito de “meme” introduzido por Richard Dawkins, o qual representa uma unidade de evolução cultural que pode exhibir aperfeiçoamento local (Dawkins, 1976).

Na última década os AMs basearam-se na utilização de uma variedade de

métodos distintos para realizar o procedimento de melhoria local. Estudos recentes na escolha do método de pesquisa local mostram que esta escolha afecta significativamente a eficiência do problema de pesquisa (Ong e Keane, 2004).

A figura 5.1 ilustra o AM proposto neste trabalho.

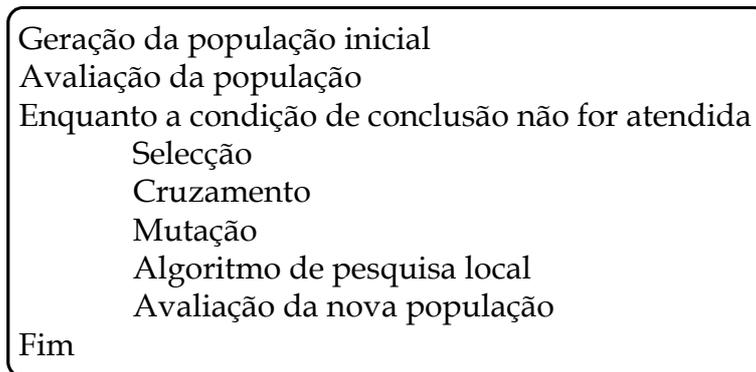


Figura 5. 1- Algoritmo memético

Como se mostra na figura 5.2, o AM proposto associa um AG com um algoritmo de pesquisa local. O AG corresponde ao algoritmo implementado no primeiro estágio de desenvolvimento deste trabalho que se encontra descrito no capítulo 3.

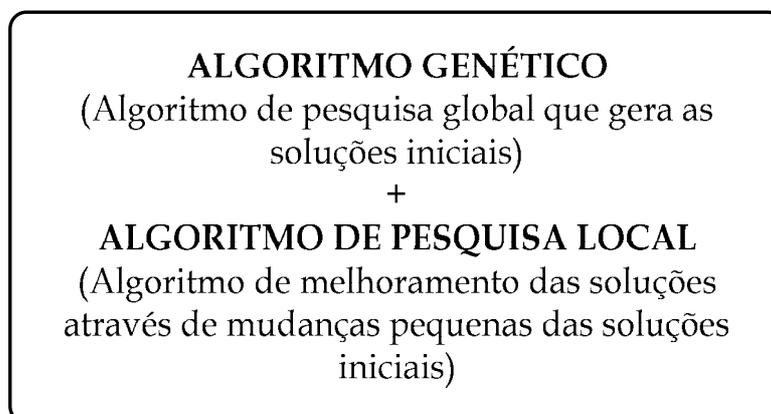


Figura 5. 2 - Algoritmo genético e algoritmo de pesquisa local

1.1. Pesquisa local

O método de pesquisa local investiga uma pequena área à volta da solução e adopta a melhor solução encontrada. Por outras palavras, o procedimento tenta encontrar uma melhor solução na vizinhança da solução corrente. Assim, se o algoritmo encontrar uma solução melhor, então a nova solução substitui a solução corrente.

Os métodos de pesquisa local são algoritmos iterativos que procuram o aperfeiçoamento da solução através de melhorias sucessivas. A forma mais simples de pesquisa local, usada nos problemas de optimização combinatória, alicerça-se na troca de elementos do cromossoma que representa a solução. No nosso caso houve necessidade de desenvolver um algoritmo de pesquisa local específico e bem adaptado ao cromossoma apresentado na figura 3.3 do capítulo 3. O principal desafio que se colocou foi a adaptação do modelo de pesquisa local com representação no domínio contínuo ao domínio discreto. Como o cromossoma é constituído por um conjunto de células, que depende do tamanho da matriz, optou-se por em cada uma das células se substituir o gene *<tipo de porta lógica>* por um vizinho, isto é incrementando ou decrementando o valor existente de uma unidade desde que os novos valores pertençam à gama de valores possíveis. Este algoritmo de pesquisa local foi designado por algoritmo de pesquisa local do tipo de porta (APLTP) e é apresentado na figura 5.3.

Ainda na pesquisa local e com o objectivo de estudar, em termos de tempo de processamento, o impacto de se iniciar a pesquisa pela primeira célula do cromossoma ou pela última, desenvolveram-se algumas experiências cujos resultados são apresentados nos gráficos das figuras 5.4 e 5.5.

Para toda a população
 Para todo o cromossoma
 Substituir o gene *tipo de porta* por um vizinho
 Fim para
 Se a nova solução tem melhor função de aptidão
 Nova solução substitui a solução antiga
 Fim para

Figura 5. 3 - Algoritmo de pesquisa local do tipo de porta (APLTP)

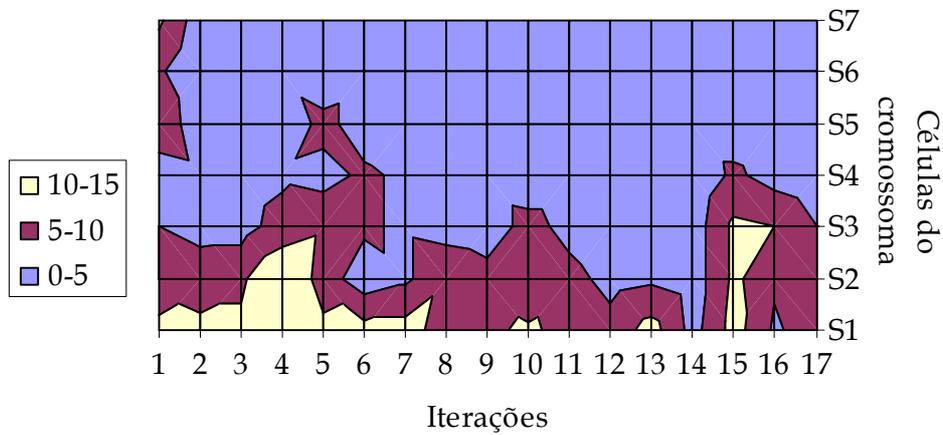


Figura 5. 4 - Algoritmo APLTP com início da pesquisa local na primeira célula do cromossoma para o circuito M21, com o conjunto de portas lógicas *Gset 2*.

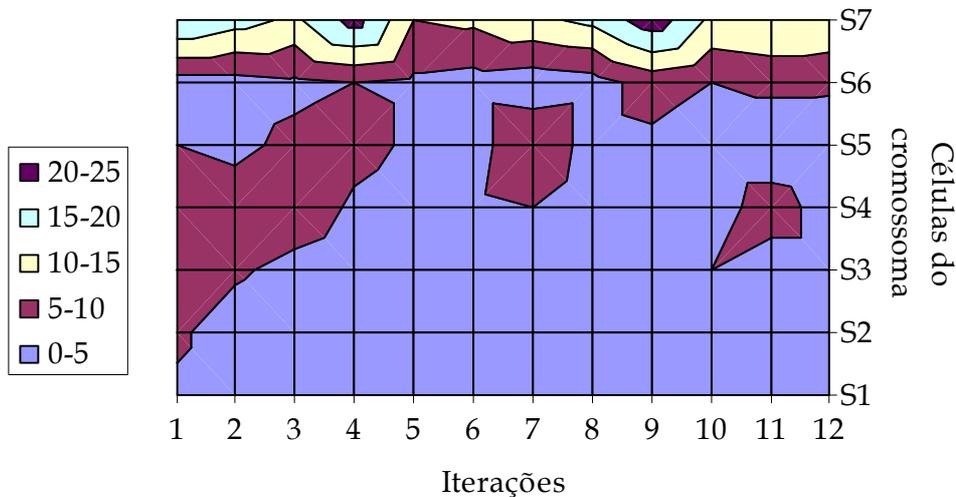


Figura 5. 5 - Algoritmo APLTP com início da pesquisa local na última célula do cromossoma para o circuito M21, com o conjunto de portas lógicas *Gset 2*.

Como se pode constatar pelos dois últimos gráficos, as trocas de genes mais significativas ocorrem sempre nas células por onde se inicia a pesquisa. No gráfico da figura 5.4 as células que apresentam mais trocas são as S_1 e S_2 e no gráfico da figura 5.5 são as células S_7 e S_6 . Em termos de tempo de processamento não foram detectadas diferenças que possam ser consideradas.

2. Experiências desenvolvidas

O AM implementado neste trabalho tem como base em termos de codificação, método de selecção e operadores genéticos, o AG descrito no capítulo 3 (Reis, Machado e Cunha, 2004).

A função de aptidão F adoptada foi apresentada no capítulo 4 com a designação F_e . O cálculo de F está dividido em duas partes f_1 e f_2 , onde f_1 mede a funcionalidade dos circuitos e a descontinuidade do erro e f_2 mede a simplicidade do circuito digital.

Numa primeira fase compara-se a saída \mathbf{Y} , produzida pelo AM ao gerar o circuito, com os valores pretendidos \mathbf{Y}_R , de acordo com a tabela de verdade que é percorrida *bit a bit*. Por outras palavras, f_{11} é incrementada de uma unidade por cada *bit* correcto da saída até f_{11} atingir o valor máximo f_{10} , que ocorre quando se obtém um circuito funcional.

Para se medir a variabilidade do erro de saída, f_{11} é decrementado de um valor $\delta \in [0, 1]$ para cada descontinuidade do erro $\mathbf{Y}_R - \mathbf{Y}$, onde descontinuidade significa passar de $\mathbf{Y}_R - \mathbf{Y} = 0$ para $\mathbf{Y}_R - \mathbf{Y} = 1$, ou vice-versa, quando se comparam duas linhas consecutivas da tabela de verdade (estruturada segundo um código de *Gray* dos *bits* de entrada).

Assim que o circuito atinge a funcionalidade, (*i. e.*, numa segunda fase) o AM tenta gerar circuitos com o menor número de portas lógicas. Isto significa que o circuito resultante deve conter o maior número de genes possível do <tipo de porta> \equiv <wire>. O índice f_2 , que avalia a simplicidade (o número de operações nulas), é incrementado de *uma unidade* (*zero unidades*) para cada *wire* (*porta lógica*) do circuito gerado. Desta forma a função de aptidão é calculada de acordo com o seguinte esquema:

- Primeira fase, funcionalidade do circuito:

$$f_{10} = 2^{n_i} \times n_o \quad (5.1)$$

$$f_{11} = f_{11} + 1 \text{ se } \{\text{bit } i \text{ de } \mathbf{Y}\} = \{\text{bit } i \text{ de } \mathbf{Y}_R\}, i = 1, \dots, f_{10} \quad (5.2)$$

$$f_1 = f_{11} - \delta \text{ se } \text{erro}_i \neq \text{erro}_{i-1}, i = 1, \dots, f_{10} \quad (5.3)$$

(quando se mede a descontinuidade)

- Segunda fase, simplicidade do circuito:

$$f_2 = f_2 + 1 \text{ se } \text{tipo de porta} = \text{wire} \quad (5.4)$$

$$F = \begin{cases} f_1, & F < f_{10} \\ f_1 + f_2, & F \geq f_{10} \end{cases} \quad (5.5)$$

onde n_i e n_o representam os números de entradas e saídas do circuito.

Para a realização deste conjunto de experiências utilizaram-se $n = 20$ simulações para cada um dos casos testados. O AM foi executado para projectar quatro circuitos lógicos combinatórios, o multiplexador dois para um (M21), o circuito de teste de paridade de 4 bits (TP4), o circuito somador completo de um bit (SOM1) e o multiplicador de 2 bits (MUL2) usando a função de aptidão

representada na equação 5.5. Adoptaram-se os quatro conjuntos de portas lógicas ilustrados na tabela 3.1 do capítulo 3 e no AG consideraram-se os parâmetros $TC = 95\%$, $TM = 20\%$ e $P = 100$.

2.1. Função de aptidão sem medida da descontinuidade do erro

O conjunto de experiências descritas nesta sub-secção foi realizado com a função de aptidão F sem incluir qualquer medida da descontinuidade do erro ($\delta = 0$).

Na figura 5.6 pode-se ver a evolução de F versus o número de gerações N necessárias para alcançar a solução no caso do circuito M21. Como este circuito tem $n_i = 3$ e $n_o = 1$ resulta $f_{10} = 8$ e $F \geq 12$.

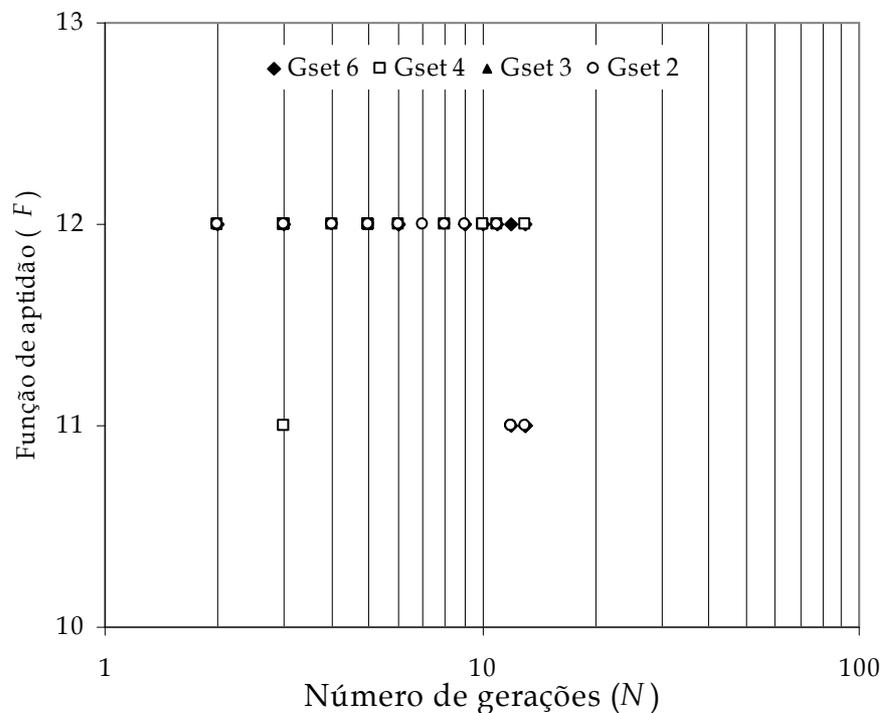


Figura 5. 6- Função de aptidão F ($\delta = 0$) versus o número de gerações N necessárias para obter a solução para o circuito M21, utilizando o AM.

Verifica-se que, neste caso, o melhor conjunto de portas lógicas é o *Gset 3*, dado que chega à solução com a menor média do número de gerações $\mu(N)$ e com a melhor média da função de aptidão $\mu(F)$.

O gráfico da figura 5.7 apresenta os resultados para o circuito SOM1. Este circuito tem $n_i = 3$ e $n_o = 2$, o que dá origem a $f_{10} = 16$ e $F \geq 20$. Analisando os resultados, constata-se que o melhor conjunto de portas lógicas é o conjunto *Gset 3*, pois requer um menor valor da média do número de gerações para obter a solução $\mu(N)$, acompanhada de uma boa média final da função de aptidão $\mu(F)$. Os melhores circuitos SOM1 obtidos têm uma função de aptidão final $F = 19$. Para este caso, a diferença entre os conjuntos de portas lógicas é reduzida.



Figura 5. 7- Função de aptidão F ($\delta = 0$) versus número de gerações N necessária para obter a solução para o circuito SOM1, utilizando o AM.

O terceiro caso consiste no circuito TP4 que apresenta $n_i = 4$ e $n_o = 1$, resultando em $f_{10} = 16$ e $F \geq 24$. Os melhores resultados foram conseguidos com o Gset 3 e podem ser analisados no gráfico da figura 5.8. O melhor circuito projectado possui função de aptidão $F = 25$.

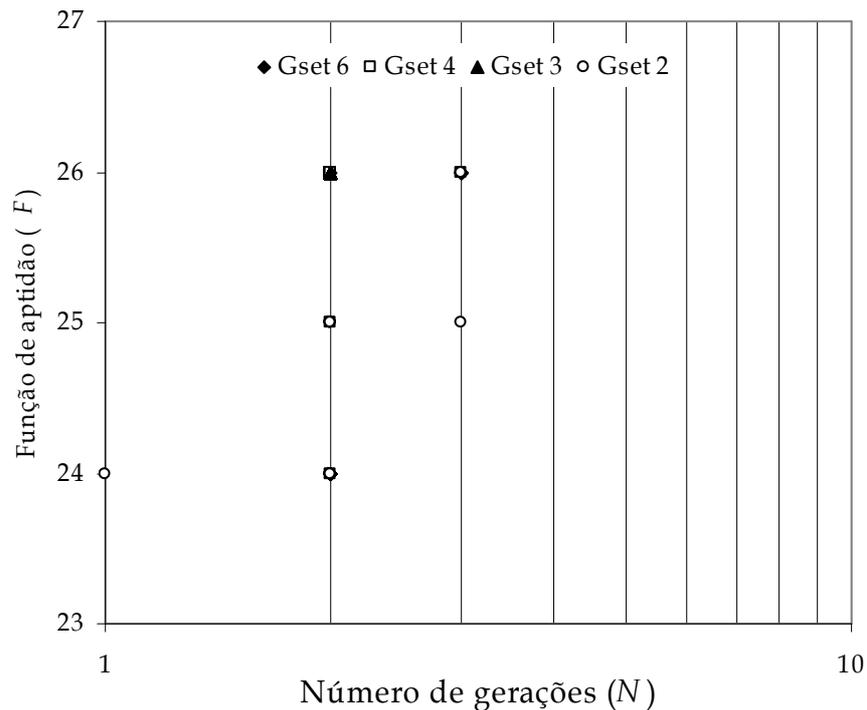


Figura 5. 8- Função de aptidão $F(\delta = 0)$ versus o número de gerações N para obter a solução para o circuito TP4, utilizando o AM.

O quarto circuito é o MUL2, com $n_i = 4$ e $n_o = 4$, dando origem a $f_{10} = 64$ e $F \geq 72$. Mais uma vez se conclui que o Gset 2 é o conjunto com melhor desempenho na síntese destes circuitos combinatórios (figura 5.9).

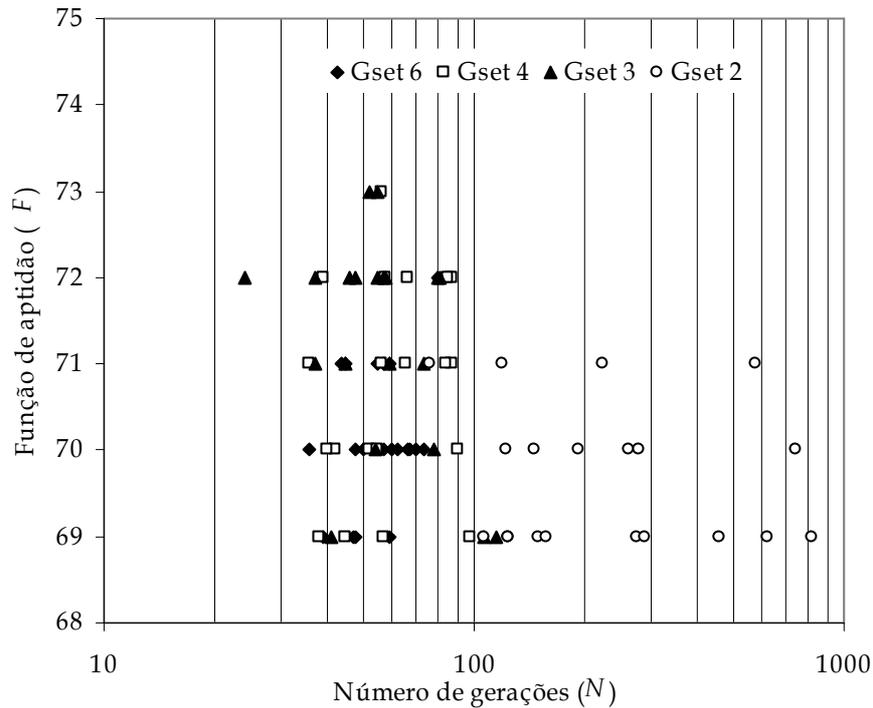


Figura 5. 9- Função de aptidão F ($\delta = 0$) versus o número de gerações N necessárias para obter a solução para o MUL2, utilizando o AM .

A tabela 5.1 mostra a média do número de gerações necessárias para alcançar a solução $\mu(N)$ e a média da função de aptidão $\mu(F)$ após a realização de vinte experiências para cada conjunto de portas lógicas. As figuras 5.10 e 5.11 ilustram estes resultados.

A figura 5.12 ilustra a média da função de aptidão $\mu(F)$ versus o número de gerações $\mu(N)$ necessárias para obter a solução, para todos os conjuntos de portas lógicas (*i. e.*, Gsets 2, 3, 4 e 6) e para os circuitos M21, SOM1, TP4 e MUL2.

Tabela 5. 1- Resultados para os circuitos M21, SOM1, TP4 e MUL, utilizando o AM

Conjunto	Circuito							
	M21		SOM1		TP4		MUL2	
	$\mu(N)$	$\mu(F)$	$\mu(N)$	$\mu(F)$	$\mu(N)$	$\mu(F)$	$\mu(N)$	$\mu(F)$
Gset 6	10,15	11,55	15,30	19,00	2,50	25,10	56,10	70,15
Gset 4	5,40	11,95	14,00	19,00	2,05	25,85	61,85	70,70
Gset 3	3,05	12,00	13,40	19,00	2,00	26,00	60,05	71,25
Gset 2	6,55	11,80	17,70	18,30	2,05	24,50	293,05	69,70

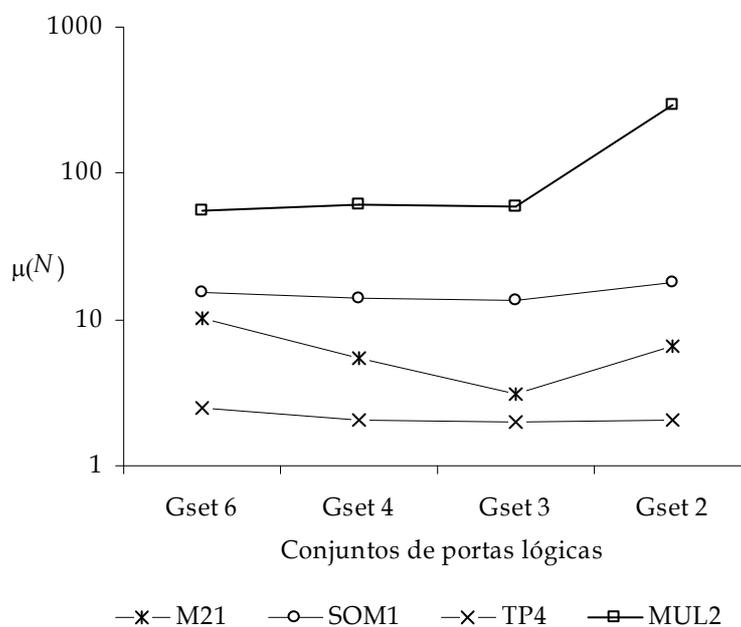


Figura 5. 10- Média do número de gerações para alcançar a solução $\mu(N)$ para os conjuntos de portas lógicas em avaliação, utilizando o AM.

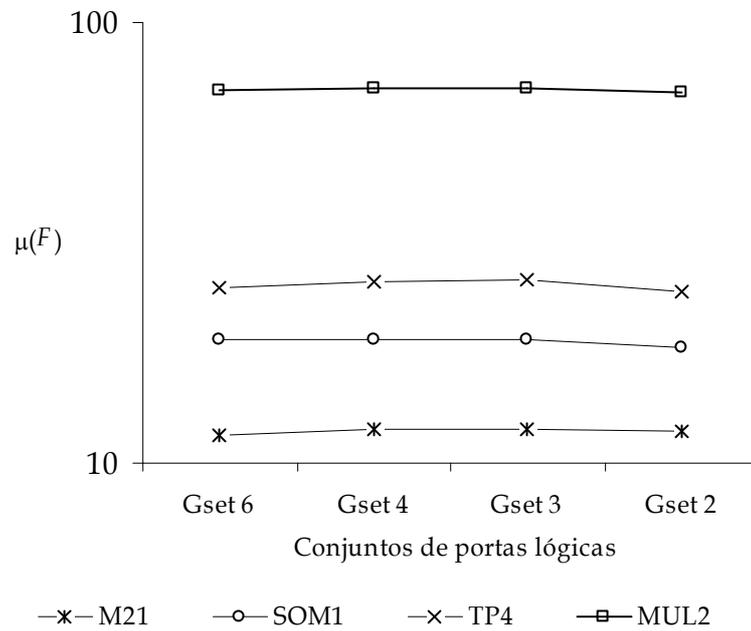


Figura 5.11 - Média da função de aptidão dos circuitos obtidos $\mu(F)$ para os conjuntos de portas lógicas em avaliação, utilizando o AG.

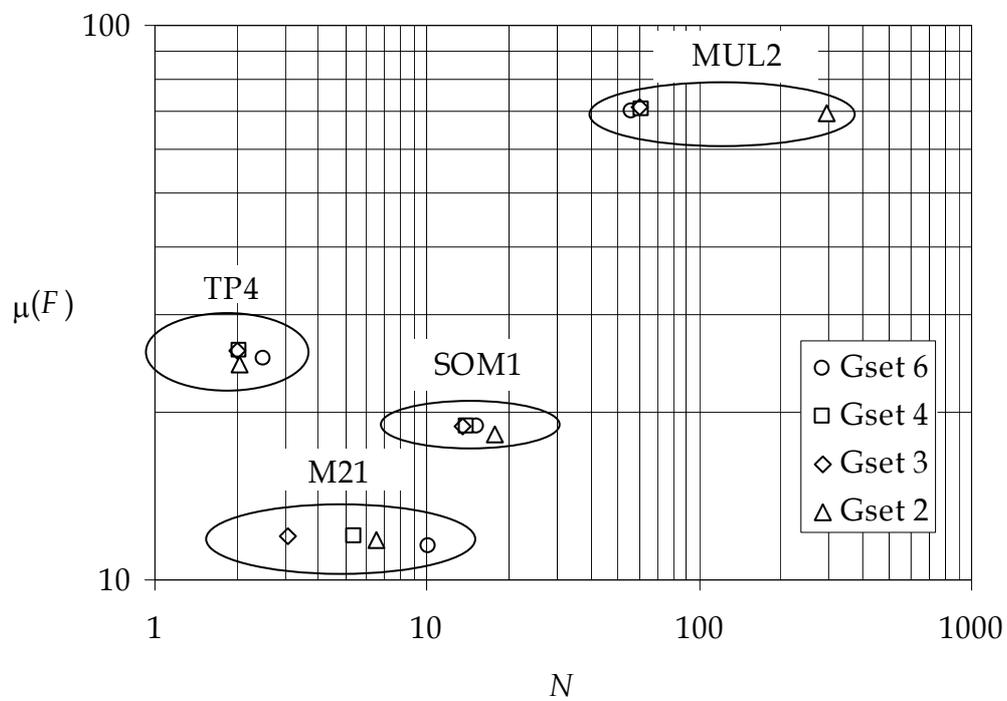


Figura 5.12 - Média da função de aptidão $\mu(F)$ versus o número de gerações necessárias para obter a solução N , utilizando o AM.

Comparando os quatro casos estudados com base na média do número de gerações necessárias para a obtenção das soluções $\mu(N)$ e na média resultante da função de aptidão $\mu(F)$, conclui-se que para o circuito MUL2 o conjunto de portas lógicas *Gset 6* é o que apresenta melhores resultados. Para os restantes circuitos (M21, SOM1 e TP4) é o *Gset 3*.

2.2. Função de aptidão com medida da descontinuidade do erro

Nesta sub-secção analisa-se o melhoramento do algoritmo AM quando se adopta uma função de aptidão que inclui a medida da descontinuidade do erro (*i. e.*, contabilizando $\delta = 0,5$). Desenvolveram-se experiências para os circuitos M21, TP4, SOM1 e MUL 2 com os conjuntos de portas lógicas *Gset 2*, *Gset 3*, *Gset 4* e *Gset 6*.

A figura 5.13 apresenta os resultados obtidos, com um tamanho de população $P = 10$, para os circuitos M21, TP4 e SOM1. Como o circuito MUL2 é um circuito de complexidade superior é necessário aumentar o tamanho da população para $P = 1000$. A figura 5.14 ilustra os resultados obtidos para este caso.

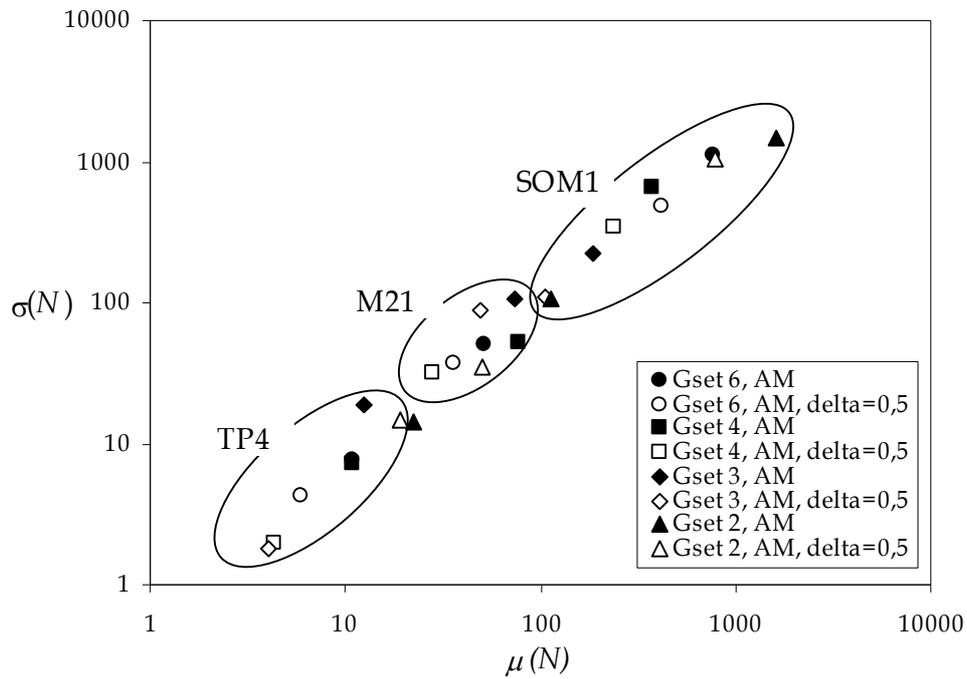


Figura 5. 13 – Desvio padrão $\sigma(N)$ versus média do número de gerações $\mu(N)$ para obter a solução, para os circuitos M21, TP4 e SOM1, com $P = 10$ e $\delta = 0,5$ (para os casos do AM com medida da descontinuidade do erro).

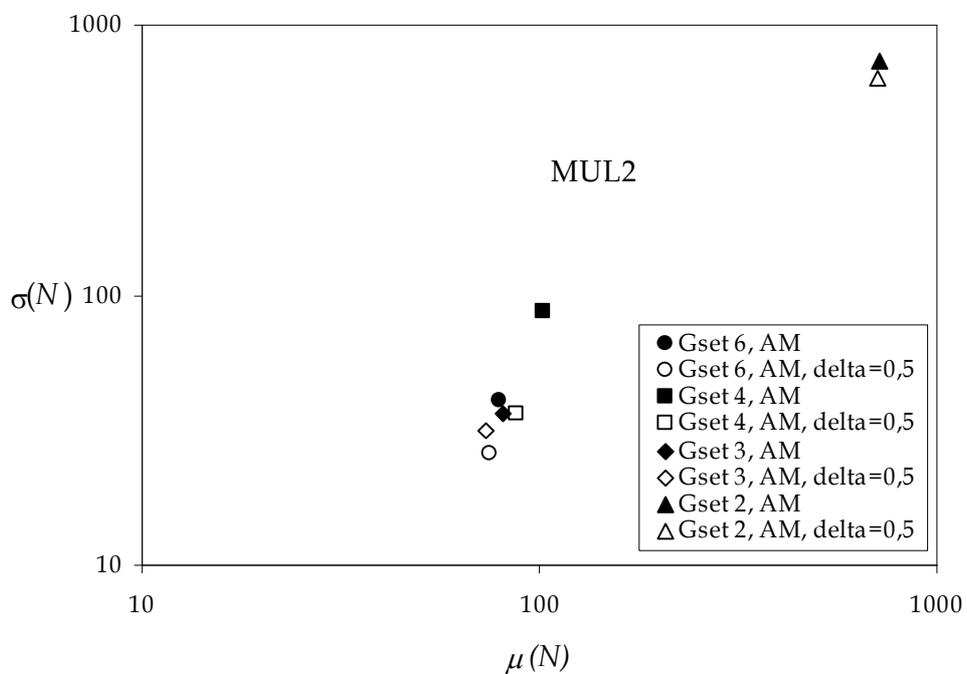


Figura 5. 14 – Desvio padrão $\sigma(N)$ versus média do número de gerações $\mu(N)$ para obter a solução, para o circuito MUL2, com $P = 1000$ e $\delta = 0,5$ (para os casos do AM com medida da descontinuidade do erro).

O melhoramento do algoritmo evolutivo AM através da aplicação da função de aptidão com a medida da descontinuidade do erro δ é mais evidente em termos do desvio padrão. Verifica-se, também, que o número médio de gerações para se obter a solução é ligeiramente melhorada quando se aplica a função de aptidão com a medida da descontinuidade do erro, para o mesmo circuito e mesmo conjunto de portas lógicas (Reis *et. al.*, 2006).

3. Análise da convergência

Esta secção endereça um tema importante dos algoritmos evolutivos em geral. Devido à sua natureza estocástica, estes algoritmos podem apresentar dificuldades na convergência para a solução final. Como tal, e com o objectivo de analisar esta questão no âmbito da síntese de circuitos lógicos combinatórios, analisa-se, de seguida, a média do número de gerações $\mu(N)$ e o desvio padrão $\sigma(N)$ para se obter a solução, considerando diferentes tamanhos de população P (Reis, Machado e Cunha, 2005a).

As figuras 5.15 a 5.18 exibem $\mu(N)$ *versus* P para o algoritmo AM na síntese dos circuitos M21, SOM1, TP4 e MUL2, para os conjuntos de portas lógicas $Gset 2$, $Gset 3$, $Gset 4$ e $Gset 6$.

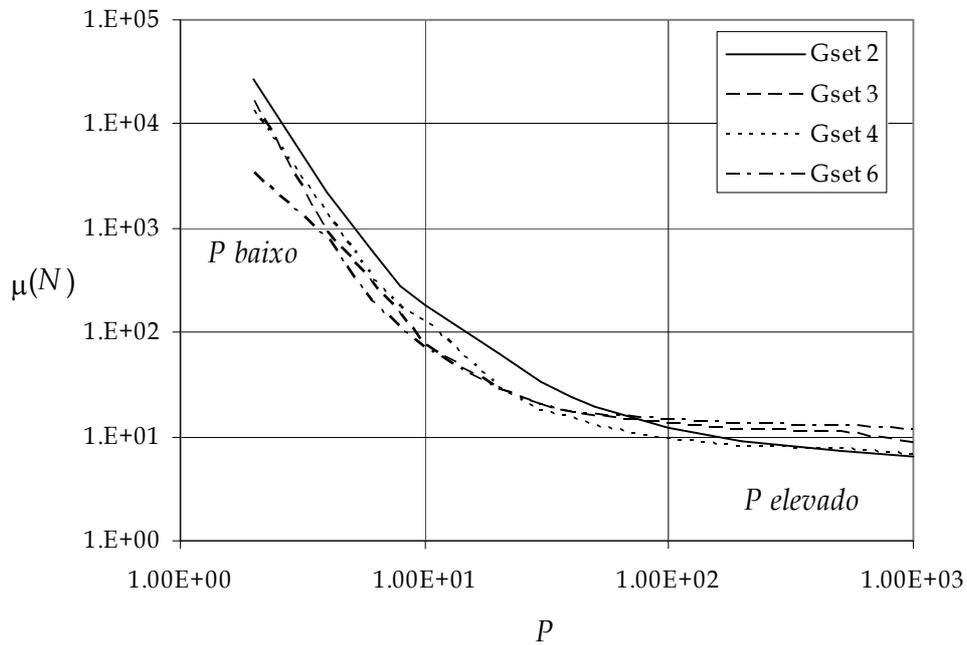


Figura 5.15 - Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito M21, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

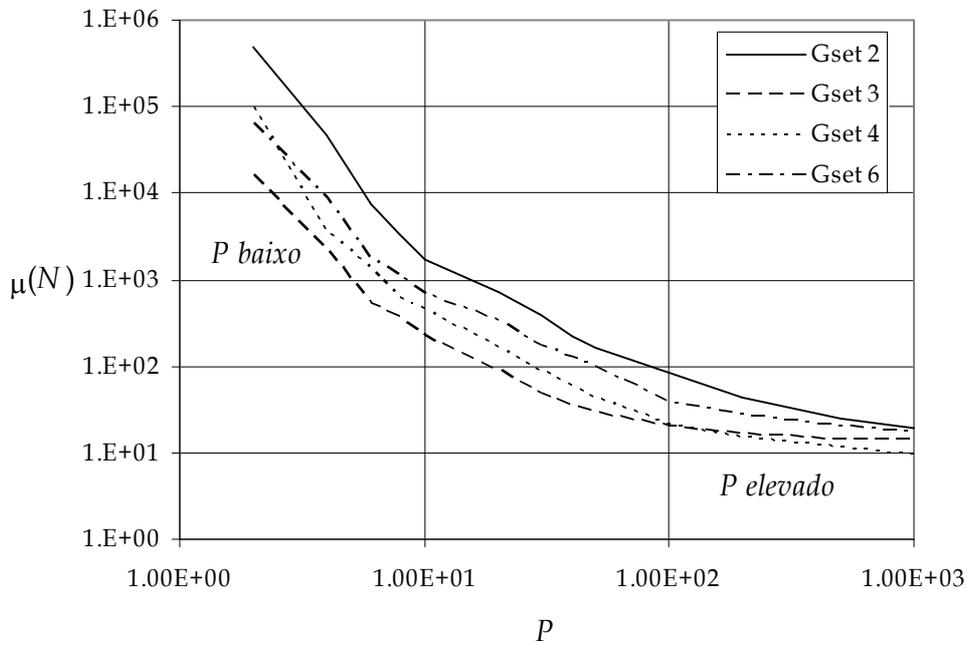


Figura 5.16 - Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito SOM1, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

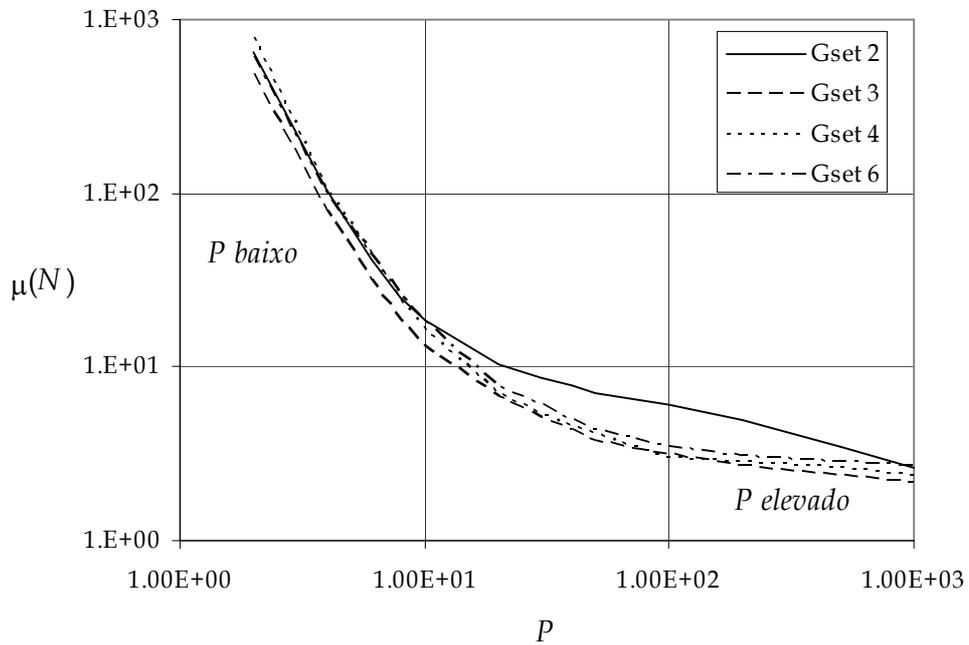


Figura 5.17 - Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito TP4, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

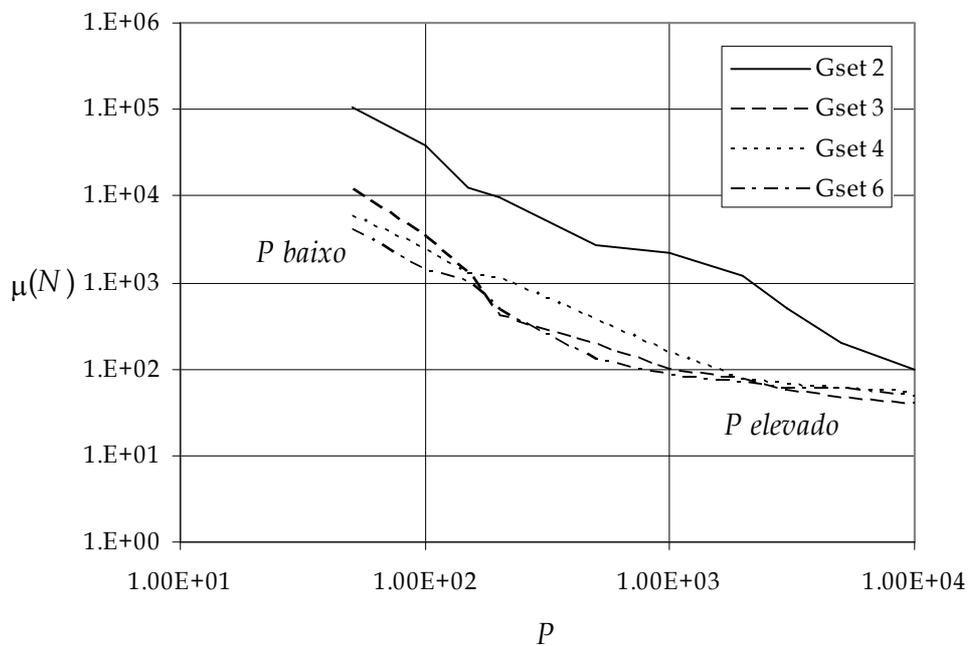


Figura 5.18 - Média do número de gerações para obter a solução $\mu(N)$ versus o tamanho da população P , para o circuito MUL2, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

Nos gráficos das quatro figuras anteriores é possível dividir os resultados em duas regiões, a saber, a região de “baixo P ” e a região de “elevado P ”, que seguem aproximadamente a expressão da forma:

$$\mu(N) \approx \alpha P^\beta \quad \alpha, \beta \in \mathfrak{R} \quad (5.6)$$

As tabelas 5.2 e 5.3 mostram os parâmetros (α, β) da equação 5.6 para os vários casos.

Tabela 5.2 - Parâmetros $\mu(N) \approx \alpha P^\beta$ para “ P baixo”

Conjunto	M21		TP4		SOM1		MUL2	
	α	β	α	β	α	β	α	β
Gset 6	2×10^4	-2,50	$2,8 \times 10^3$	-2,31	$6,7 \times 10^5$	-3,27	1×10^6	-1,43
Gset 4	$9,4 \times 10^4$	-2,52	$4,2 \times 10^3$	-2,39	1×10^6	-4,03	1×10^6	-1,40
Gset 3	$1,5 \times 10^5$	-3,43	$2,4 \times 10^3$	-2,39	$1,4 \times 10^5$	-3,09	4×10^7	-2,04
Gset 2	$2,6 \times 10^5$	-3,32	$3,6 \times 10^3$	-2,51	6×10^6	-3,65	5×10^7	-1,62

As figuras 5.19 a 5.22 mostram o desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o algoritmo AM, na síntese dos circuitos M21, SOM1, TP4 e MUL2 e para os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6.

Tabela 5.3 - Parâmetros $\mu(N) \approx \alpha P^\beta$ para "P elevado"

Conjunto	M21		TP4		SOM1		MUL2	
	α	β	α	β	α	β	α	β
Gset 6	24,87	-0,11	5,64	-0,11	2×10^2	-0,37	4×10^2	-0,23
Gset 4	35,88	-0,26	5,23	-0,11	$1,1 \times 10^2$	-0,36	4×10^2	-0,23
Gset 3	38,22	-0,22	7,56	-0,19	$1,1 \times 10^2$	-0,34	5×10^2	-0,28
Gset 2	35,44	-0,25	27,99	-0,34	$6,6 \times 10^2$	-0,51	1×10^8	-1,54

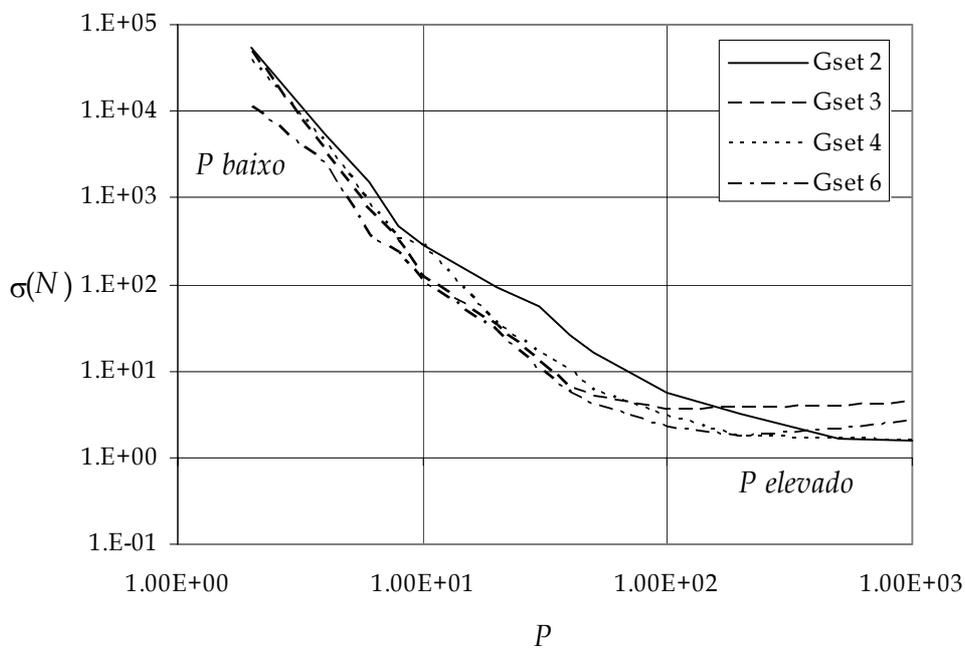


Figura 5.19 - Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito M21, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

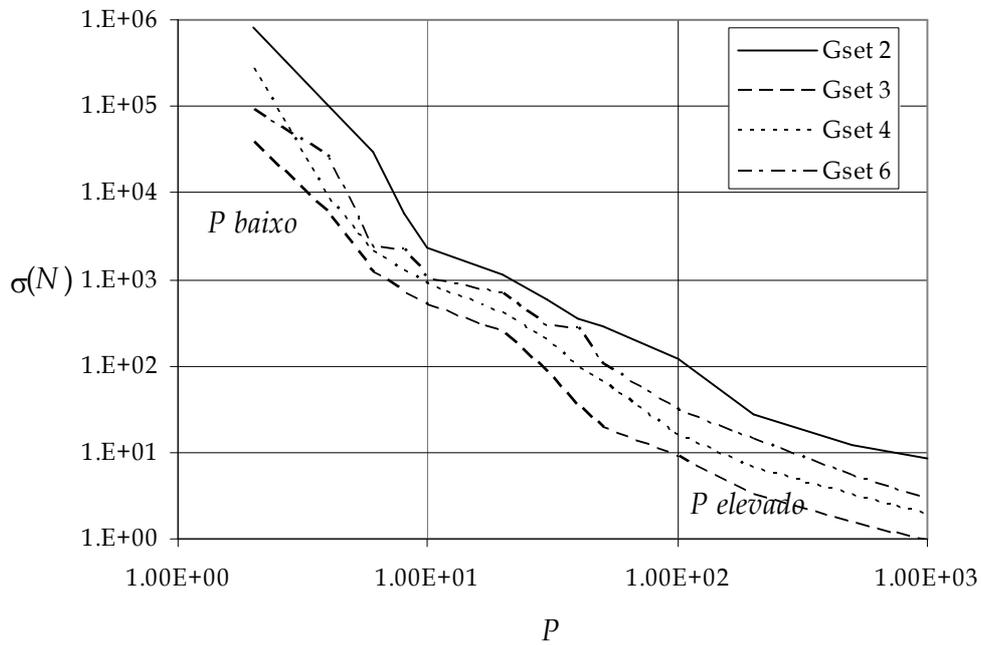


Figura 5. 20 - Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito SOM1, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

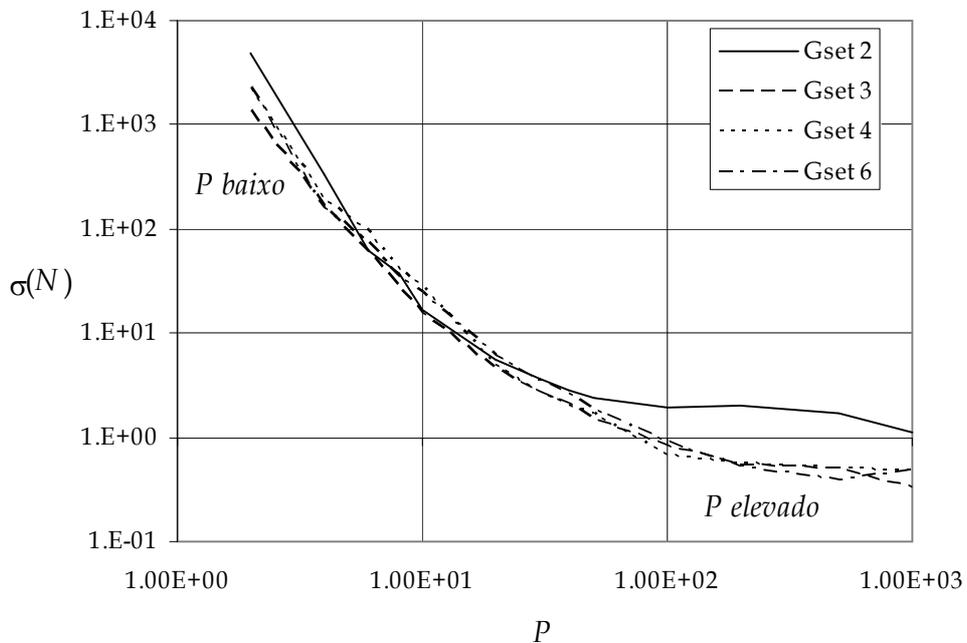


Figura 5. 21 - Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito TP4, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

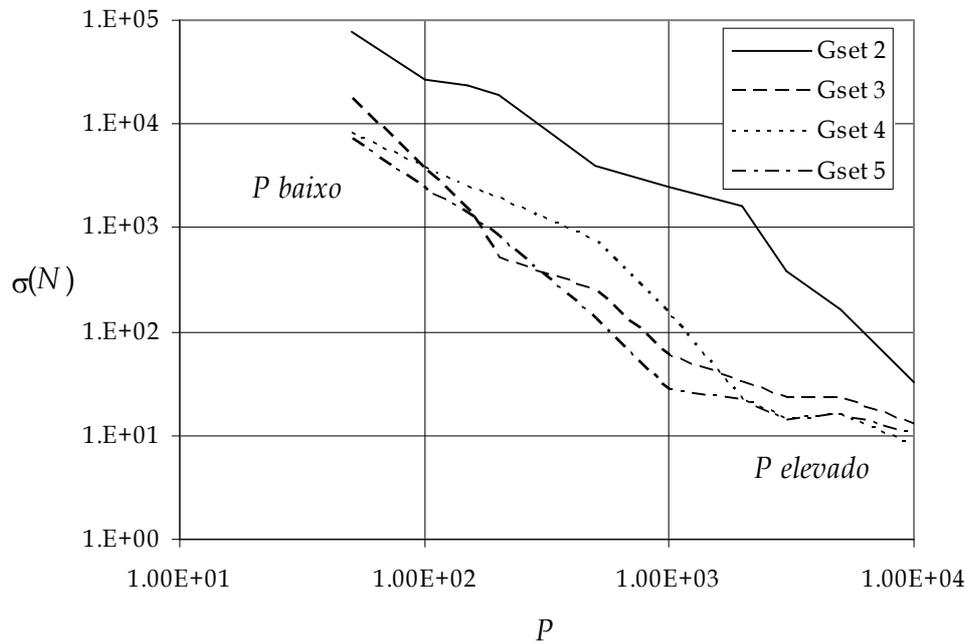


Figura 5.22 - Desvio padrão do número de gerações para obter a solução $\sigma(N)$ versus o tamanho da população P , para o circuito MUL2, com os conjuntos de portas lógicas Gset 2, Gset 3, Gset 4 e Gset 6, utilizando o AM.

O comportamento de $\sigma(N)$ é similar ao de $\mu(N)$, sendo possível, de forma idêntica, dividir o gráfico em duas áreas, a área de valores baixos de P e a área de valores elevados de P , que podem ser representadas por:

$$\sigma(N) \approx \chi P^\delta \quad \chi, \delta \in \mathfrak{R} \quad (5.7)$$

As tabelas 5.4 e 5.5 ilustram os parâmetros (χ, δ) da equação 5.7.

Tabela 5.4 - Parâmetros $\sigma(N) \approx \chi P^\delta$ para "P baixo"

Conjunto	M21		TP4		SOM1		MUL2	
	χ	δ	χ	δ	χ	δ	χ	δ
Gset 6	1x10 ⁴	-2,95	1,8x10 ⁴	-3,20	8x10 ⁴	-2,91	3x10 ⁶	-1,53
Gset 4	2,8x10 ⁵	-3,07	1x10 ⁴	-2,60	6x10 ⁶	-4,49	4,2x10 ⁵	-1,02
Gset 3	6x10 ⁵	-3,70	8,4x 0 ³	-2,75	3,1x10 ⁵	-2,97	2x10 ⁷	-1,84
Gset 2	5,4x10 ⁵	-3,32	7,4x10 ⁴	-3,93	6x10 ⁶	-3,00	7x10 ⁶	-1,17

Tabela 5.5 - Parâmetros $\sigma(N) \approx \chi P^\delta$ para "P elevado"

Conjunto	M21		TP4		SOM1		MUL2	
	χ	δ	χ	δ	χ	δ	χ	δ
Gset 6	0,58	0,22	0,08	0,26	3,2x10 ³	-1,02	4,7x10 ²	-0,41
Gset 4	2,6	-0,07	0,66	-0,04	6,2x10 ²	-0,84	1,5x10 ³	-0,56
Gset 3	2,42	0,08	11,99	-0,53	1,7x10 ²	-0,76	2,2x10 ³	-0,56
Gset 2	0,37	0,21	24,42	-0,44	1,5x10 ³	-0,76	8x10 ¹⁰	-2,36

Este estudo revelou a importância da atribuição ao tamanho da população de um valor equilibrado. Por um lado, com base nos resultados obtidos no capítulo 3 sabe-se que o tempo de processamento para a solução no AG diminui com o tamanho da população até ser atingido um limite mínimo inferior, por

outro lado constata-se agora que o desempenho do AM melhora até ser atingido um limite máximo superior.

Nesta sequência é possível estabelecer que, para o AM, na síntese dos circuitos M21, TP4, SOM1 e MUL2, com os conjuntos de portas lógicas 2, 3, 4 e 6, o valor ideal do tamanho da população é $P = 100$.

4. AM versus AG

Esta secção dedica-se ao estudo comparativo entre os algoritmos AG e AM desenvolvidos para a síntese de circuitos lógicos combinatórios.

A tabela 5.6 apresenta a média ($\mu(N)$), o desvio padrão ($\sigma(N)$) do número de gerações para alcançar a solução e a média da função de aptidão $\mu(F)$ para cada um dos conjuntos de portas lógicas estudados, para os algoritmos AG e AM, para o circuito M21. Pode-se constatar que o melhor caso ocorre para o conjunto de portas lógicas *Gset 3* com o AM, dado que é o caso onde se obtém o menor $\mu(N)$ e a melhor $\mu(F)$.

Tabela 5. 6 - Resultados para o circuito M21 com os algoritmos AG e AM

Conjunto	$\mu(N)$		$\sigma(N)$		$\mu(F)$	
	AG	AM	AG	AM	AG	AM
Gset 6	27,15	10,15	10,00	3,65	10,25	11,55
Gset 4	19,75	5,40	4,29	3,23	10,35	11,95
Gset 3	13,55	3,05	2,98	1,10	10,65	12,00
Gset 2	12,05	6,55	2,78	3,69	11,15	11,80

A tabela 5.7 apresenta a média ($\mu(N)$), o desvio padrão ($\sigma(N)$) do número de gerações necessárias para alcançar a solução e a média da função de aptidão $\mu(F)$ para cada um dos conjuntos de portas lógicas estudados, para os algoritmos AG e AM, para o circuito SOM1. Uma vez mais, o melhor caso ocorre com o conjunto de portas lógicas *Gset 3* com o algoritmo AM.

A tabela 5.8 apresenta a média $\mu(N)$, o desvio padrão $\sigma(N)$ do número de gerações necessárias para alcançar a solução e a média da função de aptidão $\mu(F)$ para cada um dos conjuntos de portas lógicas estudados, para os algoritmos AG e AM, para o circuito TP4. O conjunto de portas lógicas *Gset 3* apresenta, em conjugação com o algoritmo AM, os melhores resultados.

A tabela 5.9 apresenta a média $\mu(N)$, o desvio padrão $\sigma(N)$ do número de gerações necessárias para alcançar a solução e a média da função de aptidão $\mu(F)$ para cada um dos conjuntos de portas lógicas estudados, para os algoritmos AG e AM, para o circuito MUL2. Os melhores resultados são obtidos com o algoritmo AM, mas neste caso o conjunto de portas lógicas *Gset 6* é o que apresenta os melhores resultados em termos de $\mu(N)$, sendo o *Gset 3* superior no que diz respeito a $\mu(F)$.

Tabela 5. 7 - Resultados para o circuito SOM1 com os algoritmos AG e AM

Conjunto	$\mu(N)$		$\sigma(N)$		$\mu(F)$	
	AG	AM	AG	AM	AG	AM
Gset 6	72,45	15,30	52,98	1,75	18,15	19,00
Gset 4	53,65	14,00	29,11	0,86	18,35	19,00
Gset 3	32,40	13,40	10,60	0,50	18,45	19,00
Gset 2	34,86	17,70	6,44	4,59	18,57	18,30

Tabela 5. 8 - Resultados para o circuito TP4 com os algoritmos AG e AM

Conjunto	$\mu(N)$		$\sigma(N)$		$\mu(F)$	
	AG	AM	AG	AM	AG	AM
Gset 6	32,55	2,50	8,85	0,51	21,70	25,10
Gset 4	20,40	2,05	5,05	0,22	21,95	25,85
Gset 3	13,75	2,00	1,80	0,00	22,65	26,00
Gset 2	7,95	2,05	4,10	0,39	23,95	24,50

Tabela 5. 9 - Resultados para o circuito MUL2 com os algoritmos AG e AM

Conjunto	$\mu(N)$		$\sigma(N)$		$\mu(F)$	
	AG	AM	AG	AM	AG	AM
Gset 6	1699	56.10	1713	11.59	69.15	70.15
Gset 4	1183	61.85	1652	20.05	69.50	70.70
Gset 3	432	60.05	595	22.85	70.25	71.25
Gset 2	362	293.05	357	225.32	70.45	69.70

A figura 5.23 ilustra a média da função de aptidão $\mu(F)$ versus a média do número de gerações necessárias para obter a solução $\mu(N)$, para os algoritmos AG e AM, para todos os conjuntos de portas lógicas e para os quatro circuitos combinatórios.

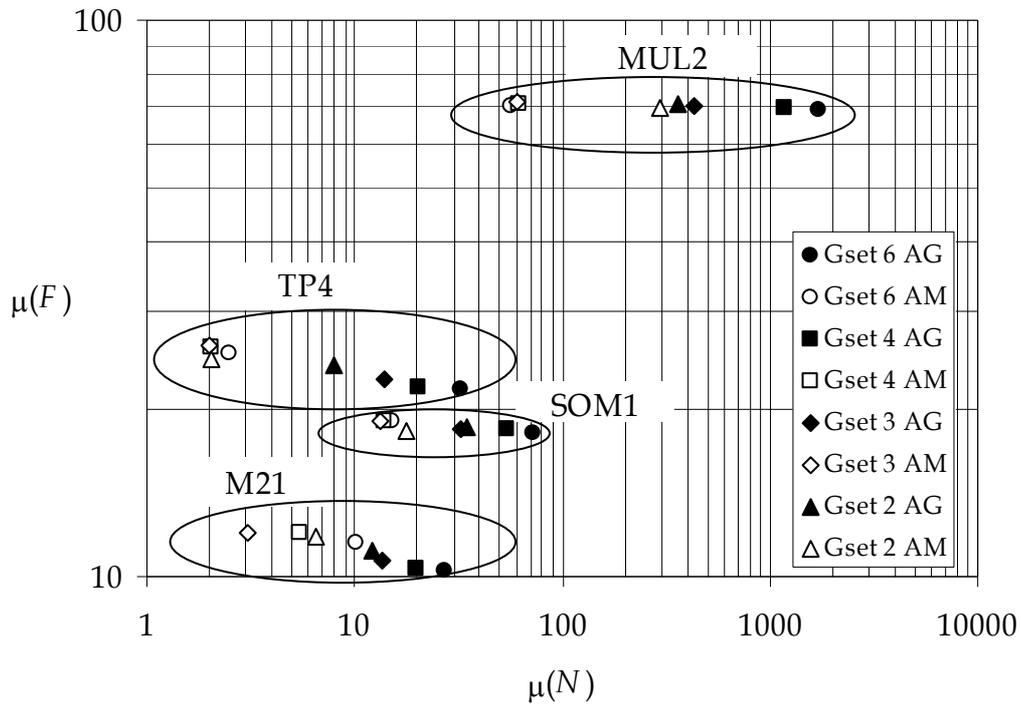


Figura 5. 23 - Média da função de aptidão $\mu(F)$ versus o número de gerações necessárias para obter a solução $\mu(N)$, para $P = 3000$.

Os resultados obtidos permitem concluir que o algoritmo AM apresenta desempenho superior para todos os conjuntos de portas lógicas e para todos os circuitos, em particular na perspectiva de $\mu(N)$. O conjunto de portas lógicas *Gset 3* demonstra ser o conjunto mais eficiente.

4.1. Convergência

Nesta sub-secção pretende fazer-se um estudo comparativo da convergência entre os algoritmos AM e AG.

Os gráficos apresentados nas figuras 5.24 e 5.25, revelam, respectivamente, o

tempo de processamento TPS versus $(\sigma(N), \mu(N))$ e a sua projecção no plano horizontal $(\sigma(N), \mu(N))$ no projecto do circuito TP4 com o conjunto de portas lógicas $Gset 2$. Ignorando o TPS (Reis, Machado e Cunha, 2004), atingem-se melhores resultados para tamanhos de população elevados. No entanto, em termos de TPS , os melhores resultados são alcançados para, respectivamente, $P = 100$ e $P = 20$ para os casos do AG e do AM.

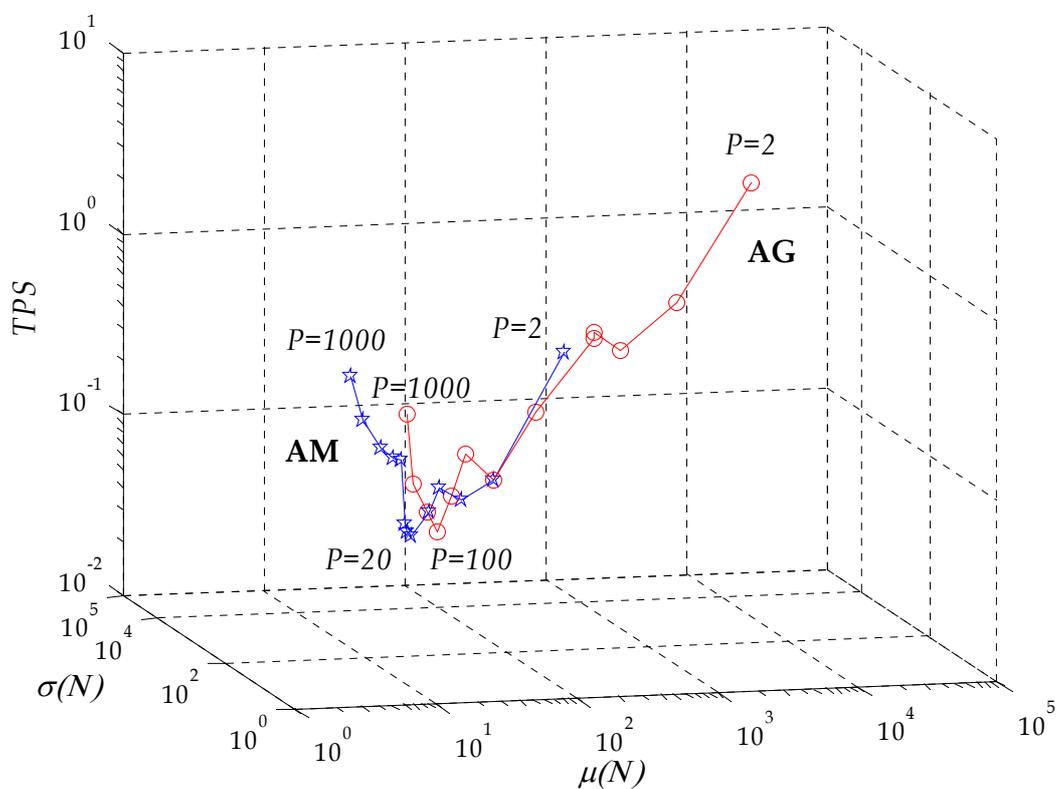


Figura 5. 24 - Tempo de processamento TPS versus o desvio padrão do número de gerações para obter a solução $\sigma(N)$ e a média do número de gerações para obter a solução $\mu(N)$, para os algoritmos AM e AG, com o conjunto de portas lógicas $Gset 2$ e o circuito TP4.

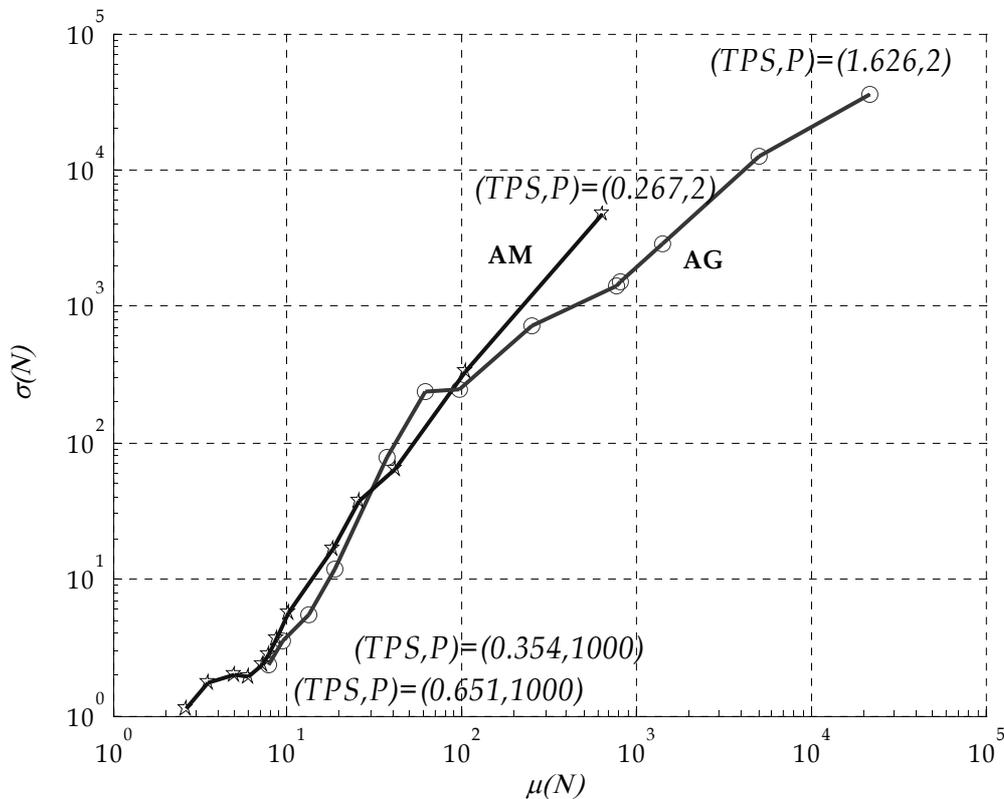


Figura 5. 25 - Projecção no plano horizontal de TPS versus o desvio padrão do número de gerações para obter a solução $\sigma(N)$ e a média do número de gerações para obter a solução $\mu(N)$, para os algoritmos AM e AG, com o conjunto de portas lógicas $Gset 2$ e o circuito TP4.

4.2. Escala

A questão do aumento exponencial de escala, na síntese de circuitos combinatórios, foi já abordada no capítulo 3 em relação ao AG. Esta secção abrange o mesmo problema relativamente ao AM, comparando-o com o algoritmo AG.

A figura 5.26 ilustra a evolução de $\mu(F)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para um crescente número de *bits*. A família de circuitos de teste de paridade é constituída por {2, 3, 4, 5, 6} *bit*. Analisando os

traçados verifica-se que, depois de decorrido o transitório inicial, tem-se a lei exponencial dada por:

$$\mu(F) = \alpha e^{\beta \mu(N)} \quad \alpha, \beta \in \mathfrak{R} \quad (5.8)$$

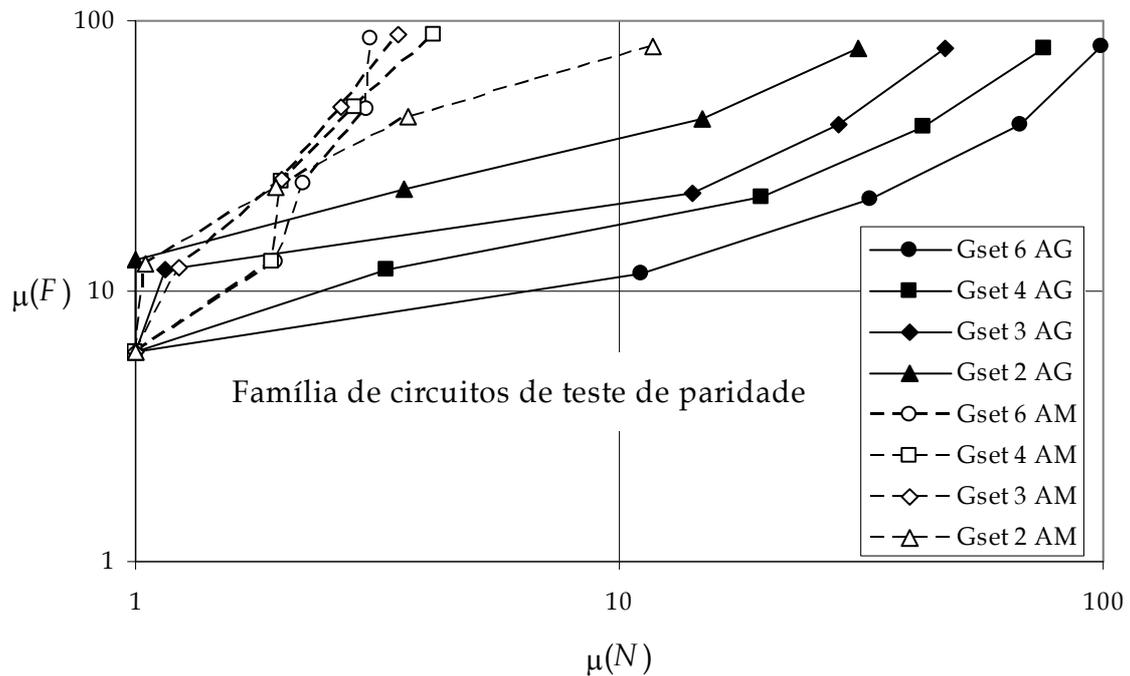


Figura 5.26 - $\mu(F)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para os algoritmos AG e AM, com os conjuntos de portas lógicas *Gset 2*, *Gset 3*, *Gset 4* e *Gset 6*, para $P = 3000$.

Os coeficientes (α, β) que resultam para cada um dos conjuntos de portas lógicas e para cada um dos algoritmos, podem ser visualizados na tabela 5.10. Para o AG e com respeito ao coeficiente α constata-se que o conjunto de portas lógicas *Gset 2* é o que apresenta melhor valor. Por seu lado, os conjuntos *Gset 3* e *Gset 4* são similares e o conjunto *Gset 6* é o que apresenta menor desempenho (Reis, Machado e Cunha, 2005b). Em termos do coeficiente β , é possível agrupar os conjuntos de portas lógicas *Gset 6* e *Gset 4*, com desempenhos inferiores, e os

conjuntos *Gset 2* e *Gset 3* com desempenhos superiores. Por outro lado, em relação ao AM, pode agrupar-se os conjuntos de portas lógicas *Gset 6, 4 e 3* para ambos os coeficientes (α , β) enquanto que o conjunto *Gset 2* exibe um comportamento inferior.

Tabela 5.10 - Coeficientes (α , β) da equação 5.8

Conjunto	AG		AM	
	α	β	α	β
<i>Gset 6</i>	9,8	0,0214	2	1,06
<i>Gset 4</i>	12,3	0,0257	1,92	1,14
<i>Gset 3</i>	12,2	0,0408	2,33	1,17
<i>Gset 2</i>	21,2	0,0433	8,44	0,47

5. Resumo do capítulo

Este capítulo apresenta um algoritmo híbrido, mais precisamente um algoritmo memético aplicado à síntese de circuitos lógicos combinatórios. Este algoritmo tem por base um AG no qual é inserido um algoritmo de pesquisa local o que permite, para cada indivíduo da população (*i. e.*, para cada potencial solução do problema), efectuar uma procura minuciosa no espaço de pesquisa vizinho dessa solução. Esta técnica reduz o número de gerações para se obter a solução e também o respectivo desvio padrão sendo, por este motivo, uma boa alternativa dentro dos algoritmos evolutivos.

Referências

- Dawkins, R., "The Selfish Gene". Oxford University Press, New York, 1976.
- Krasnogor, N., "Studies on the Theory and Design Space of Memetic Algorithms". PhD thesis, University of the West of England, Bristol, June, 2002.
- Merz, P. e Freisleben, B. "A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem," in 1999 Congress on Evolutionary Computation, pp. 2063-2070, IEEE Press, 1999.
- Moscato, P. "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms", Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.
- Ong, J. Y. S. e Keane, A. J. "Meta-Lamarckian in Memetic Algorithm", IEEE Transactions On Evolutionary Computation, Vol. 8, No. 2, pp. 99-110, April 2004.
- Reis, C., Tenreiro Machado, J. e Boaventura Cunha, J., Evolutionary Design of Combinational Logic Circuits, Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Vol. 8, No. 5, pp. 507-513, Sep. 2004.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, A Memetic Algorithm for Logic Circuit Design, 2005 WSEAS International. Conference on

Dynamical Systems and Control, Venice, Italy, pp. 598-603, November, 2005a.

Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. An Evolutionary Hybrid Approach in the Design of Combinational Digital Circuits, WSEAS Transactions on Systems, Issue 12, Vol. 4, pp.2338-2345, December, 2005b.

Reis, C., Machado, J. A. T., Figueiredo, L. e Boaventura Cunha, J. A Hybrid Algorithm for Logic Circuit Synthesis, Knowledge and Decision Technologies, pp 297-303. April, 2006.

Capítulo 6

SÍNTESE DE CIRCUITOS COM OPTIMIZAÇÃO POR ENXAMES DE PARTÍCULAS

Introdução

A designação Optimização por Enxames de Partículas (OEP) refere-se a um algoritmo que parte de uma população composta por um conjunto aleatório de soluções designadas por “partículas”. No esquema OEP, cada partícula voa através do espaço de pesquisa com uma velocidade que é ajustada dinamicamente de acordo com o seu histórico de comportamento. Por este motivo, as partículas têm tendência a voar em direcção à melhor região do espaço de procura ao longo do processo de pesquisa. Este comportamento sucede-se até ser encontrada a solução.

O presente capítulo propõe um algoritmo baseado na filosofia OEP para a síntese de circuitos lógicos combinatórios. Nesta ordem de ideias, a secção 1 apresenta uma breve revisão sobre o OEP, a secção 2 descreve o algoritmo baseado em OEP para desenho de circuitos e a secção 3 exhibe os resultados das simulações efectuadas. Por último, a secção 4 faz a síntese do capítulo.

1. Optimização por Enxames de Partículas

O termo Optimização por Enxames de Partículas refere-se a uma família relativamente recente de algoritmos que podem ser usados para encontrar soluções óptimas (ou perto do óptimo) em problemas numéricos e qualitativos. É de fácil implementação na maioria das linguagens de programação e já deu provas de ser bastante eficiente num vasto conjunto de problemas de optimização. A investigação que deu origem à teoria de enxames de partículas foi iniciada por cientistas não ligados à área da computação, mais concretamente por ornitologistas, biólogos e psicólogos.

No entanto, Eberhart e Kennedy foram os primeiros a introduzir o algoritmo de optimização por enxames de partículas (Kennedy e Eberhart, 1995), que consiste num método de optimização inspirado na inteligência de grupos de populações biológicas. Este método foi descoberto através da simulação de modelos sociais simplificados de bandos de pássaros, cardumes de peixes e da teoria dos enxames.

1.1. Parâmetros

Na OEP, em vez de serem usados operadores genéticos, como no caso dos AGs, cada partícula ajusta a sua trajectória de voo de acordo com a sua própria experiência e a experiência dos seus companheiros. Cada partícula (indivíduo) é tratada como um ponto de um espaço de dimensão D , sendo manipulada como se descreve, em seguida, no algoritmo OEP original:

$$v_{id} = v_{id} + c_1 \text{rand}() (p_{id} - x_{id}) + c_2 \text{rand}() (p_{gd} - x_{id}) \quad (6.1a)$$

$$x_{id} = x_{id} + v_{id} \quad (6.1b)$$

onde c_1 e c_2 são constantes positivas, $\text{rand}()$ é uma função aleatória na gama de $[0, 1]$, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ representa a partícula i , $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ é a melhor posição prévia (a posição correspondente ao melhor valor de função de aptidão) da partícula, o símbolo g representa o índice da melhor partícula entre todas as partículas da população e $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ é a taxa de variação de posição (velocidade) da partícula i .

As expressões 6.1 representam a trajectória de voo de uma população de partículas. A equação 6.1a descreve o modo como é actualizada dinamicamente a velocidade e a equação 6.1b descreve a actualização de posição de cada uma das partículas. A equação 6.1a divide-se em três partes, a saber: a parte do momento, a parte cognitiva e a parte social. Na primeira parte, a velocidade varia recursivamente e não pode ser modificada abruptamente. Assim, a nova velocidade é ajustada com base na velocidade actual. A segunda parte representa a aprendizagem da sua própria experiência de voo. A terceira parte, consiste na aprendizagem da experiência de voo do grupo (Shi e Eberhart, 1998).

O primeiro novo parâmetro acrescentado ao algoritmo original de OEP foi a “inércia” (Clerc e Kennedy, 2002). Nesta sequência, a equação dinâmica do OEP com o peso da inércia foi modificada para:

$$v_{id} = wv_{id} + c_1 \text{rand}() (p_{id} - x_{id}) + c_2 \text{rand}() (p_{gd} - x_{id}) \quad (6.2a)$$

$$x_{id} = x_{id} + v_{id} \quad (6.2b)$$

onde w constitui o valor da inércia que introduz um balanceamento entre as habilidades de pesquisa global e local. Uma inércia elevada facilita a pesquisa global. Ao contrário, uma inércia pequena facilita a pesquisa local.

Posteriormente foi introduzido um outro parâmetro, designado por coeficiente de restrição k , na esperança de garantir a convergência da OEP (Clerc e Kennedy, 2002). A equação 6.3a mostra um método simplificado de incorporação deste parâmetro, onde k é função de c_1 e c_2 , tal como se apresenta na equação 6.4.

$$v_{id} = k \left[v_{id} + c_1 \text{rand}() (p_{id} - x_{id}) + c_2 \text{rand}() (p_{gd} - x_{id}) \right] \quad (6.3a)$$

$$x_{id} = x_{id} + v_{id} \quad (6.3b)$$

$$k = 2 \left(2 - \phi - \sqrt{\phi^2 - 4\phi} \right)^{-1} \quad \text{onde } \phi = c_1 + c_2, \phi > 4. \quad (6.4)$$

1.2. Topologias

Existem duas topologias de OEP diferentes, a versão global e a versão local. Na versão global, cada partícula voa através do espaço de pesquisa com uma velocidade que é dinamicamente ajustada de acordo com o seu melhor desempenho e o melhor desempenho de todas as partículas até ao momento. Por outro lado, na versão local cada partícula ajusta a sua velocidade de acordo com o seu melhor desempenho e o melhor desempenho atingido até ao

momento na sua vizinhança. A vizinhança de cada partícula é normalmente definida como o conjunto de partículas geograficamente mais próximas dessa partícula em todas as direcções.

1.3. Algoritmo

O OEP é um algoritmo evolutivo envolvendo conceitos simples, fácil de implementar e computacionalmente eficiente. A figura 6.1 ilustra o procedimento original para a implementação do algoritmo OEP.

1. Inicializar a população
2. Avaliar a função de aptidão de cada partícula
3. Modificar as velocidades com base em *gbest*
4. Actualizar as posições das partículas
5. Terminar na condição de conclusão
6. Voltar a 2

Figura 6. 1- Processo de Optimização por Enxame de Partículas

As diferentes versões de algoritmos OEP são:

- OEP de valores reais, que é a versão original do OEP e adaptado para a resolução de problemas com representação das grandezas através de valores reais;
- A versão de OEP binária, para a resolução de problemas binários;
- A versão de OEP discreta, adaptada para a resolução de problemas baseados em eventos.

Para estender a versão binária/discreta à versão de valores reais, a parte mais crucial consiste em interpretar o significado dos conceitos de trajectória e velocidade no espaço binário/discreto.

Kennedy e Eberhart usam a velocidade como a probabilidade de determinar se x_{id} (um *bit*) estará num estado ou no outro (0 ou 1). A fórmula OEP da equação 6.1 permanece inalterada, com excepção de que, neste caso, p_{id} e x_{id} são inteiros na gama $[0,0; 1,0]$ e é usada uma função de transformação $S(v_{id})$ para desempenhar esta modificação. A resultante alteração de posição é definida pelas seguinte regra:

$$\text{se } [rand() < S(v_{id})] \quad \text{então } x_{id} = 1; \quad \text{senão } x_{id} = 0 \quad (6.5)$$

onde a função $S(v)$ representa uma transformação de limitação sigmoideal e $rand()$ é um número aleatório seleccionado de uma distribuição uniforme na gama de valores $[0,0; 1,0]$.

2. Algoritmo OEP para síntese de circuitos

2.1. Codificação dos circuitos e parâmetros OEP

O algoritmo OEP implementado neste trabalho utiliza a codificação dos circuitos digitais que foi apresentada e detalhada no capítulo 3. De seguida, descreve-se a opção tomada relativamente aos parâmetros OEP utilizados.

A população inicial de circuitos digitais, aqui designados por partículas, é gerada de forma aleatória. A velocidade inicial, de cada partícula, é inicializada

em zero, sendo as velocidades que se seguem calculadas utilizando a equação 6.2a. As novas posições resultam da equação 6.2b. Desta forma, cada potencial solução voa através do espaço de pesquisa. Para cada gene do cromossoma que representa a partícula calcula-se a correspondente velocidade. Por este motivo, as novas posições são em número igual ao número de genes do cromossoma. Se os novos valores dos genes *<entrada1>* e *<entrada2>* resultarem fora da gama admissível, é usada uma função de re-inserção: se o novo gene *<tipo de porta>* não for permitido, então é gerado um novo gene de forma aleatória.

Estas partículas têm memória e cada uma delas guarda informação da sua melhor posição encontrada até ao momento (*pbest*) e a respectiva função de aptidão. O enxame tem informação dos *pbest* de todas as partículas e a partícula com o valor de função de aptidão mais elevado é designada por *global best* (*gbest*).

O conceito básico da técnica OEP baseia-se em acelerar cada partícula em direcção ao seu *pbest* e *gbest* com um valor aleatório. No entanto, neste caso concreto, utiliza-se também uma espécie de operador de mutação que introduz uma nova célula em 10% da população. Este operador de mutação altera as características de uma determinada célula da matriz. Desta forma, a mutação é responsável pela modificação do tipo de porta lógica e das duas entradas, significando que pode surgir no cromossoma uma célula completamente nova.

Para executar o algoritmo OEP é necessário definir o número de indivíduos *P* para criar a população inicial de partículas. Esta população mantém sempre o mesmo tamanho ao longo das gerações, até ser atingida a solução.

2.2. Funções de aptidão

Em relação às funções de aptidão, adoptam-se as duas funções de aptidão descritas no capítulo 4, ou seja, a função de aptidão estática F_e e a função de aptidão dinâmica F_d .

O cálculo de F_e está dividido em duas partes f_1 e f_2 , onde f_1 mede a funcionalidade dos circuitos e a descontinuidade do erro e f_2 mede a simplicidade do circuito digital.

Numa primeira fase compara-se a saída \mathbf{Y} , produzida pelo OEP ao gerar o circuito, com os valores pretendidos \mathbf{Y}_R , de acordo com a tabela de verdade, e comparando *bit a bit*. Por outras palavras, f_{11} é incrementada de uma unidade por cada *bit* correcto da saída até que f_{11} atinja o valor máximo f_{10} , que ocorre quando se obtém um circuito funcional.

Para se medir a variabilidade do erro de saída, f_{11} é decrementado de $\delta \in [0, 1]$ para cada descontinuidade do erro $\mathbf{Y}_R - \mathbf{Y}$, onde descontinuidade significa passar de $\mathbf{Y}_R - \mathbf{Y} = 0$ para $\mathbf{Y}_R - \mathbf{Y} = 1$, ou vice-versa, quando se comparam duas linhas consecutivas da tabela de verdade.

Assim que o circuito atinge a funcionalidade, numa segunda fase, o AM tenta gerar circuitos com o menor número de portas lógicas. Isto significa que o circuito resultante deve conter o maior número de genes possível de *<tipo de porta>* \equiv *<wire>*. O índice f_2 , que avalia a simplicidade (o número de operações nulas), é incrementado de *uma unidade* (*zero unidades*) para cada *wire* (*porta lógica*) do circuito gerado. Desta forma, a função de aptidão é calculada de acordo com:

- Primeira fase, funcionalidade do circuito:

$$f_{10} = 2^{n_i} \times n_o \quad (6.6)$$

$$f_{11} = f_{11} + 1 \text{ se } \{\text{bit } i \text{ de } \mathbf{Y}\} = \{\text{bit } i \text{ de } \mathbf{Y}_R\}, i = 1, \dots, f_{10} \quad (6.7)$$

$$f_1 = f_{11} - \delta \text{ se } \text{erro}_i \neq \text{erro}_{i-1}, i = 1, \dots, f_{10} \quad (6.8)$$

(quando se mede a descontinuidade)

- Segunda fase, simplicidade do circuito:

$$f_2 = f_2 + 1 \text{ se tipo de porta} = \text{wire} \quad (6.9)$$

$$F_e = \begin{cases} f_1, & F_e < f_{10} \\ f_1 + f_2, & F_e \geq f_{10} \end{cases} \quad (6.10)$$

onde n_i e n_o representam o número de entradas e saídas do circuito.

A função de aptidão dinâmica F_d pode ser calculada da seguinte forma:

$$F_d = F_e + KD^\alpha [F_e] \quad (6.11)$$

onde $-1 \leq \alpha \leq 1$ representa a ordem fraccionária diferencial (integral) para valores positivos (negativos) de α e K é o “ganho” do termo dinâmico.

A equação 6.11 pode ser apresentada separando as componentes integral e diferencial, como se mostra na equação 6.12:

$$F_d = F_e + K_I I^\lambda [F_e] + K_D D^\mu [F_e] \quad (6.12)$$

onde $0.0 \leq \lambda \leq 1.0$ é o coeficiente integral de ordem fraccionária, $0.0 \leq \mu \leq 1.0$ é o coeficiente diferencial de ordem fraccionária e K_I, K_D são, respectivamente, os “ganhos” dos termos dinâmicos integral e diferencial.

3. Experiências desenvolvidas

As experiências desenvolvidas nesta secção dividem-se em dois grupos de acordo com a função de aptidão utilizada. Assim, a sub-secção 3.1 trata das experiências com a função de aptidão estática F_e definida anteriormente na equação 6.10. A sub-secção 3.2 dedica-se às simulações com a função de aptidão dinâmica F_d (equações 6.11 e 6.12). Desenvolveram-se $n = 20$ simulações para cada um dos casos testados. Considera-se o projecto de quatro circuitos lógicos combinatórios: o multiplexador dois para um (M21), o circuito de teste de paridade de 4 bits (TP4), o circuito somador completo de um bit (SOM1) e o multiplicador de 2 bits (MUL2). Usaram-se os quatro conjuntos de portas lógicas que a tabela 3.1 (apresentada no capítulo 3) ilustra, com $w = 0,5$; $c_1 = 1,5$; $c_2 = 2$ e $P = 3000$.

3.1. Com a função de aptidão estática F_e

O conjunto de experiências descritas a seguir foi realizado com a função de aptidão F_e sem medida da descontinuidade do erro ($\delta = 0$).

A figura 6.2 mostra o desvio padrão $\sigma(N)$ versus a média do número de gerações $\mu(N)$ para se obter a solução, para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, com $P = 3000$, para o algoritmo OEP.

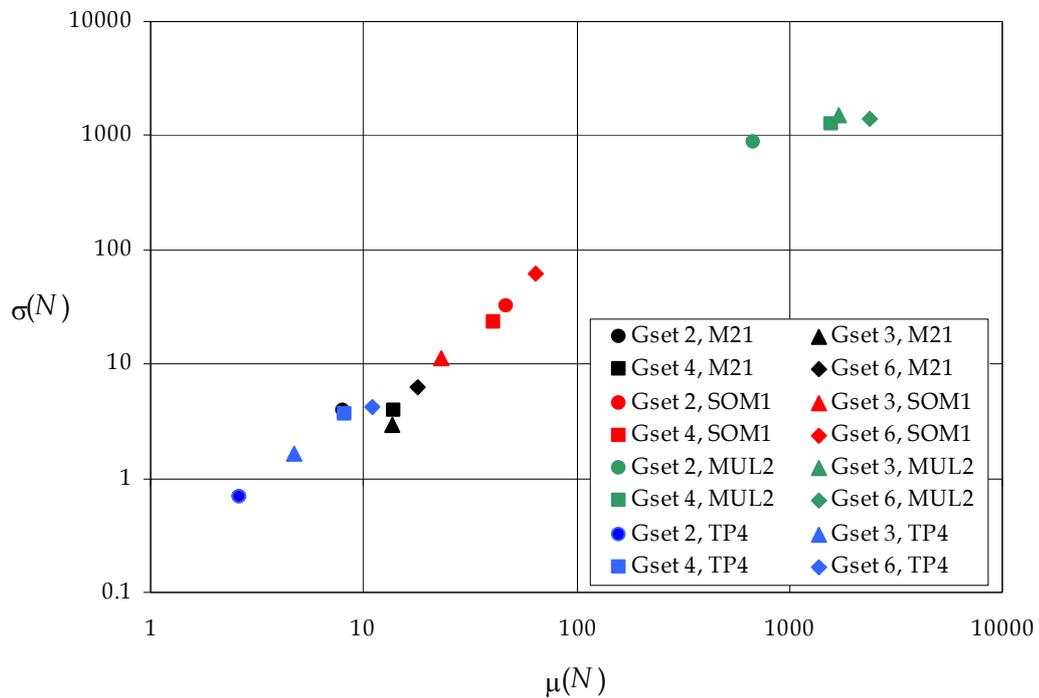


Figura 6.2- $\sigma(N)$ versus $\mu(N)$ com $P = 3000$ e $F_e (\delta = 0)$ para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para o algoritmo OEP.

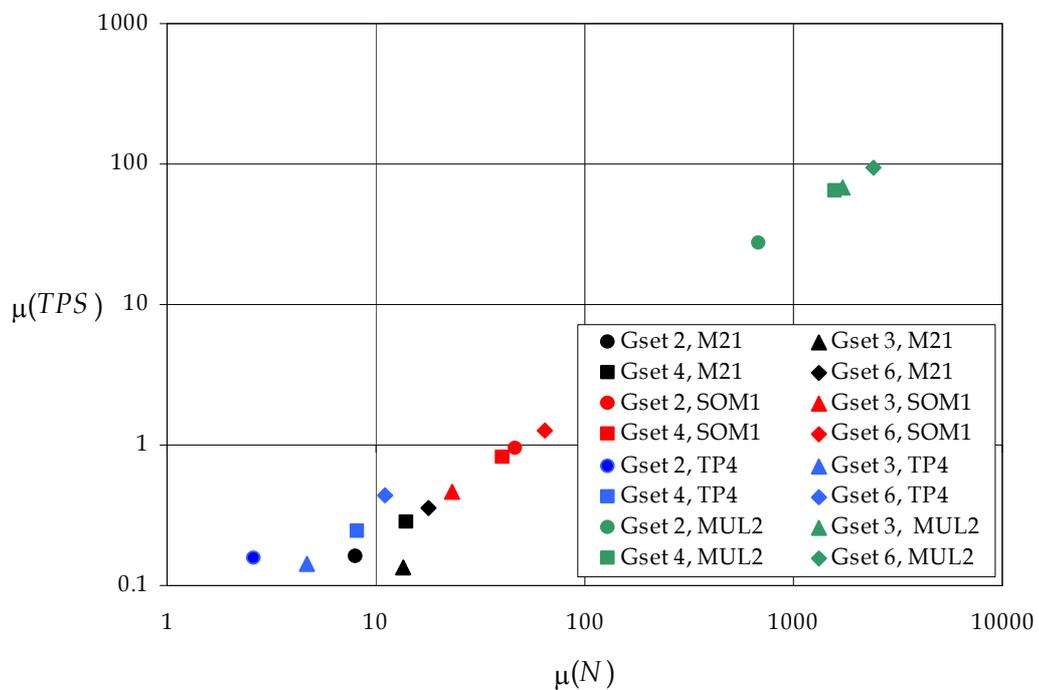


Figura 6.3- $\mu(TPS)$ versus $\mu(N)$ com $P = 3000$ e $F_e (\delta = 0)$ para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para o algoritmo OEP.

A figura 6.3 ilustra a média do tempo de processamento para obter a solução $\mu(TPS)$ versus a média do número de gerações $\mu(N)$ para obter a solução, para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para o algoritmo OEP.

Com base na análise dos últimos dois gráficos conclui-se que o conjunto de portas lógicas *Gset 2* apresenta resultados superiores em termos dos índices $\mu(N)$, $\sigma(N)$ e $\mu(TPS)$ para os circuitos M21, TP4 e MUL2. Relativamente ao circuito SOM1, o conjunto de portas lógicas que se destaca é o *Gset 3*. O *Gset 6* é o conjunto de portas lógicas com menor desempenho na síntese dos quatro circuitos testados.

O conjunto de experiências que se seguem utiliza a função de aptidão F_e com medida da descontinuidade do erro (*i. e.*, utilizando $\delta \in [0, 1]$), para gerar dois circuitos aritméticos, o somador completo de 1 *bit* (SOM1) e o subtrator completo de 1 *bit* (SUB1) (Reis, Machado e Boaventura, 2006c).

As figuras 6.4 e 6.5 apresentam a média do número de gerações para obter a solução $\mu(N)$ versus δ para o algoritmo OEP, para os circuitos {SOM1, SUB1} e os conjuntos de portas lógicas {2, 3, 4, 6}.

As figuras 6.6 e 6.7 descrevem o desvio padrão para obter a solução $\sigma(N)$ versus δ para o algoritmo OEP, para os circuitos {SOM1, SUB1} e os conjuntos de portas lógicas {2, 3, 4, 6}.

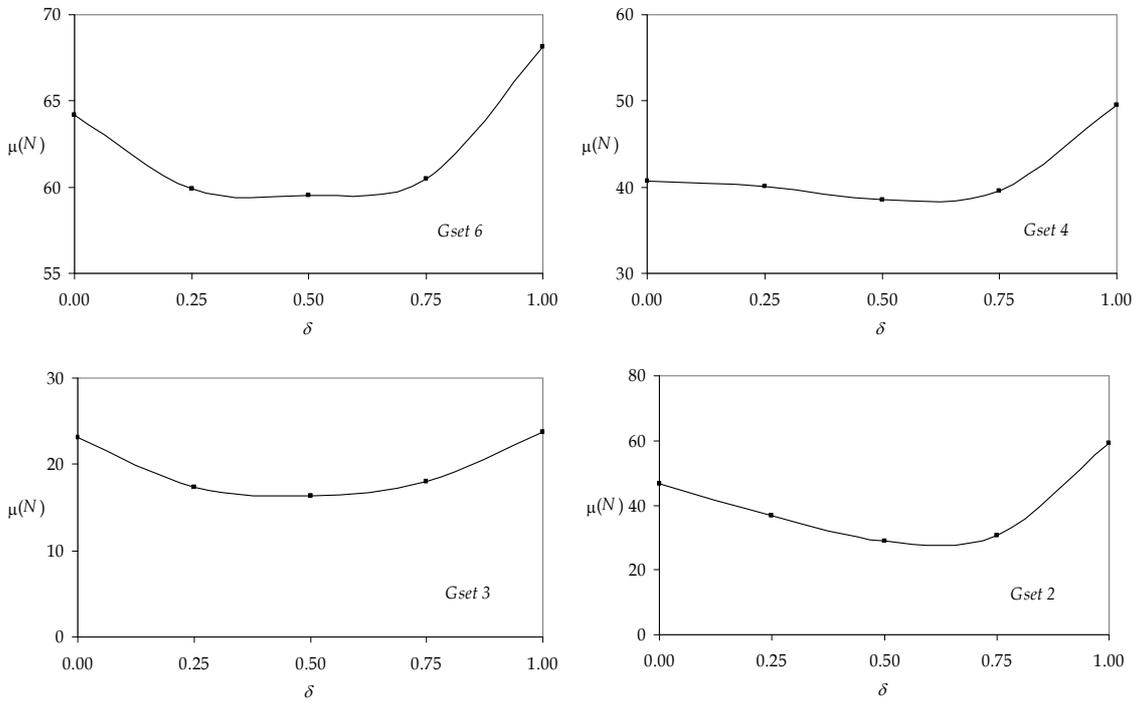


Figura 6. 4 - $\mu(N)$ para o circuito SOM1 *versus* $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}.

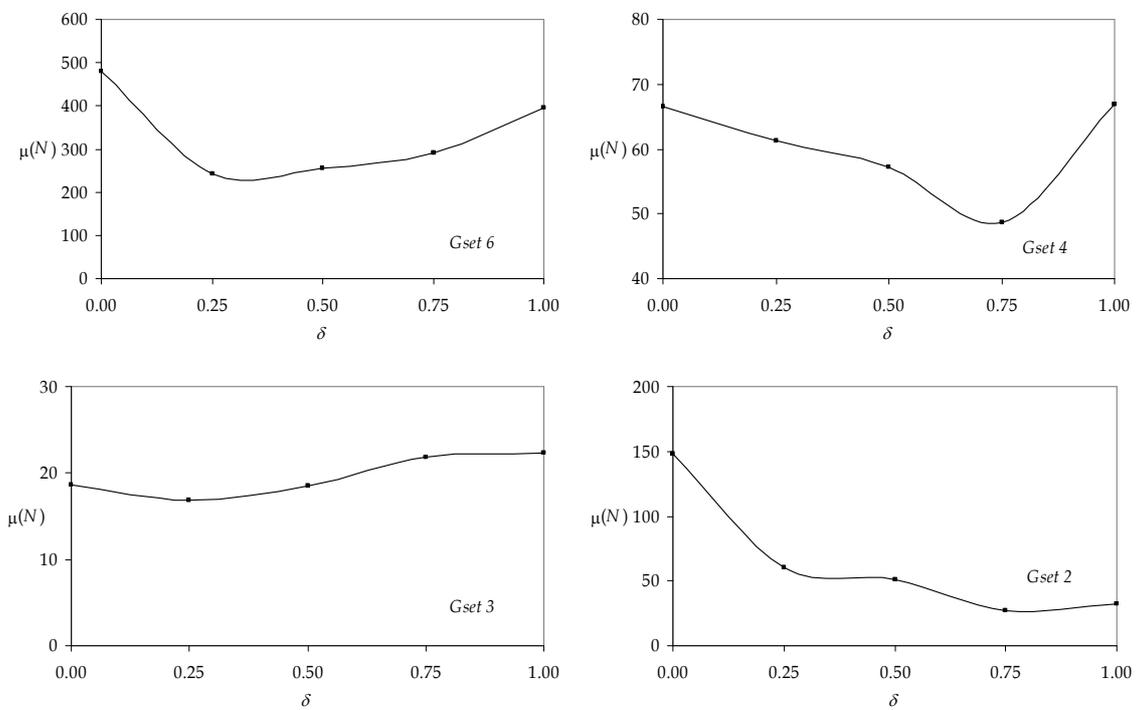


Figura 6. 5 - $\mu(N)$ para o circuito SUB1 *versus* $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}.

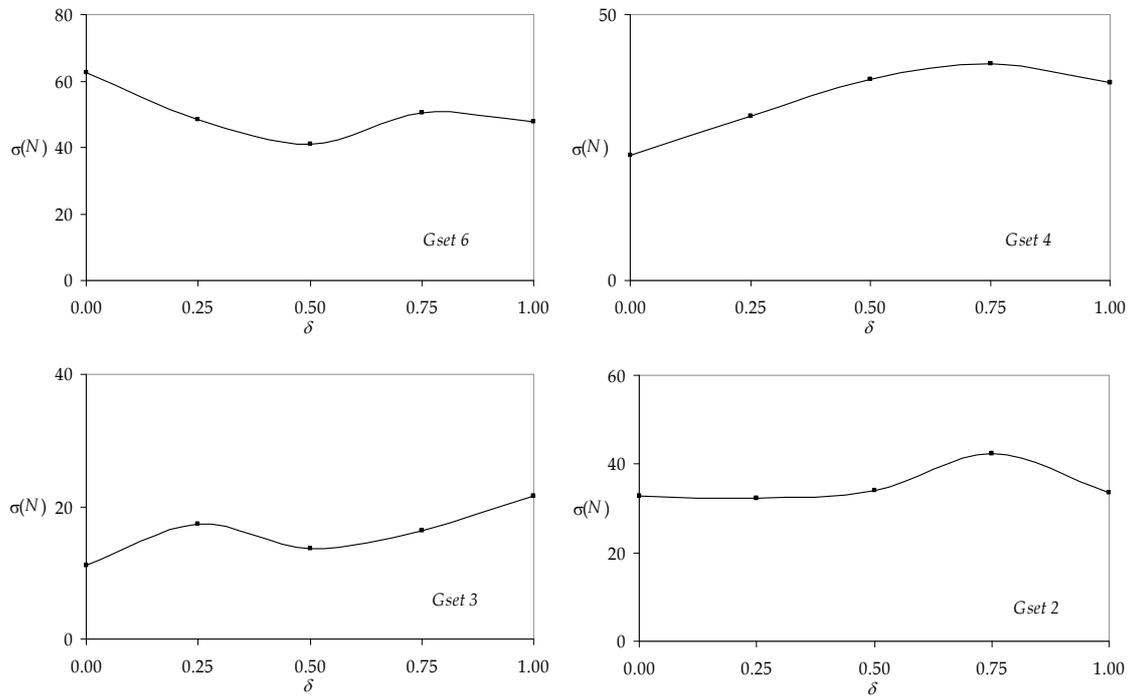


Figura 6.6 - $\sigma(N)$ para o circuito SOM1 *versus* $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}.

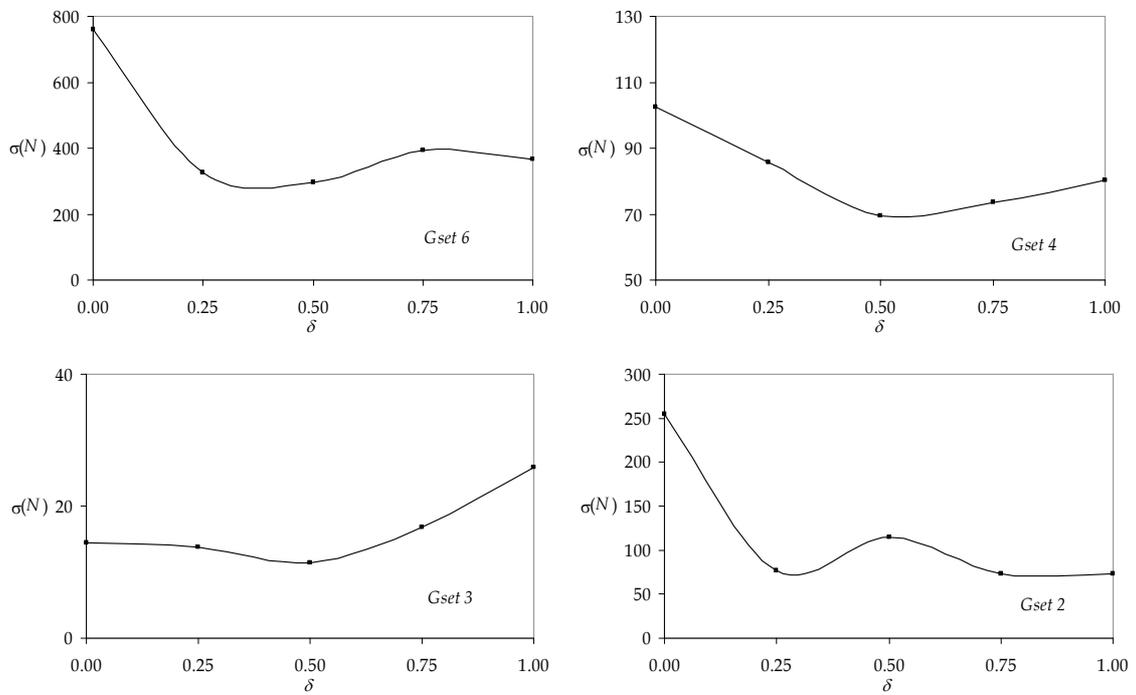


Figura 6.7 - $\sigma(N)$ para o circuito SUB1 *versus* $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2}.

As figuras 6.4 a 6.7 revelam que o conjunto de portas lógicas *Gset 3* apresenta desempenho superior para todos os valores de δ . Uma análise mais detalhada indica uma resposta melhor para valores de $\delta \approx 0,5$, para os dois circuitos aritméticos e para todos os conjuntos de portas lógicas.

A figura 6.8 ilustra os resultados comparativos entre os quatro conjuntos de portas lógicas para os dois circuitos aritméticos.

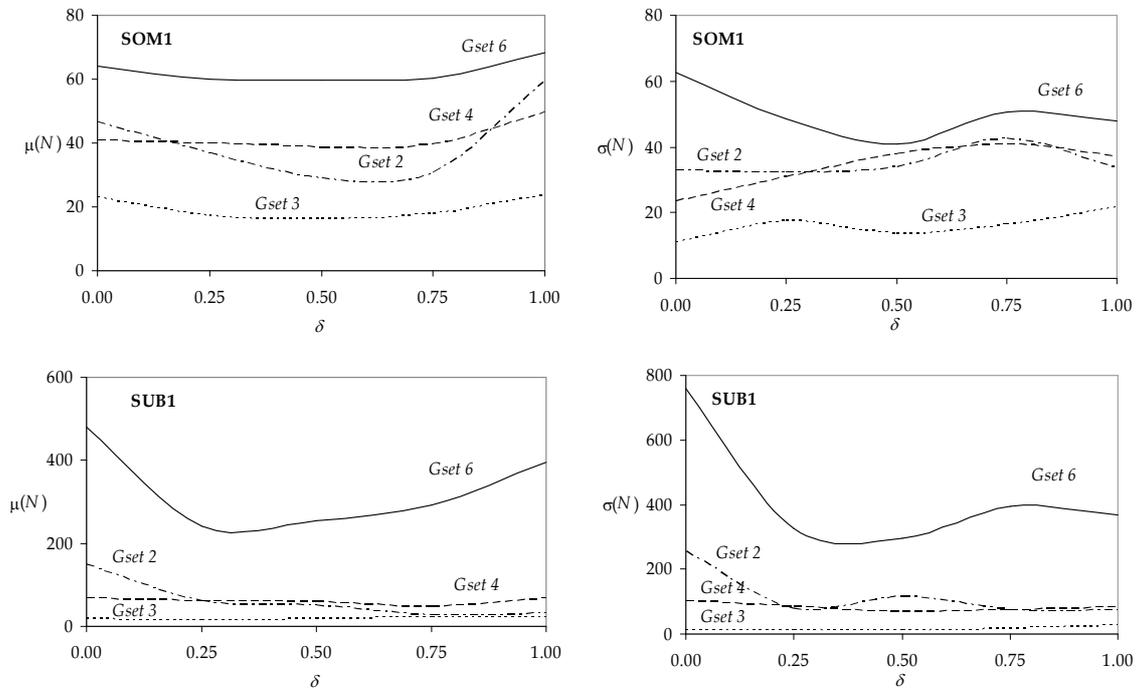


Figura 6. 8 - $\mu(N)$ e $\sigma(N)$ versus $\delta \in [0,1]$, com os conjuntos de portas lógicas {6, 4, 3, 2} para os circuitos SOM1 e SUB1.

3.2. Com a função de aptidão dinâmica F_d

Esta sub-seccção apresenta os resultados obtidos com a função de aptidão dinâmica F_d (Reis *et. al.*, 2006b).

As figuras 6.9 e 6.10 exibem respectivamente o desvio padrão $\sigma(N)$ do número de gerações necessárias para obter a solução e a média do tempo de processamento $\mu(TPS)$ para obter a solução *versus* a média do número de gerações $\mu(N)$ para obter a solução para o algoritmo OEP, utilizando F_d , os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}. Conclui-se que a função de aptidão F_d conduz a resultados superiores aos obtidos com a função de aptidão estática, em particular para o circuito MUL2 e no que diz respeito a $\mu(TPS)$.

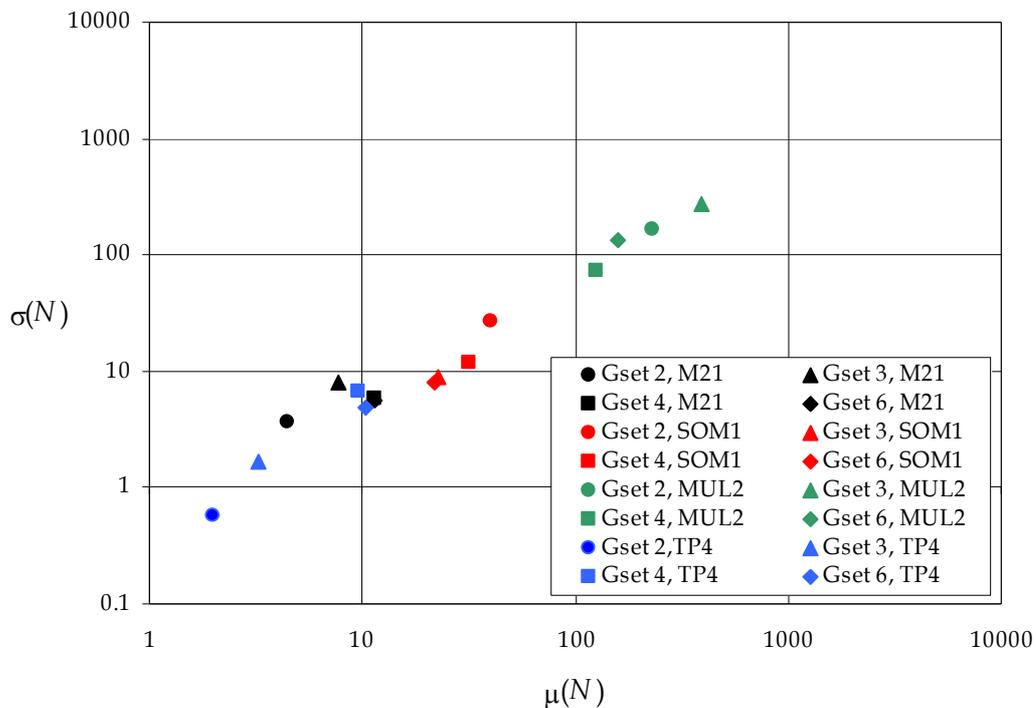


Figura 6.9 - $\sigma(N)$ *versus* $\mu(N)$ para o algoritmo OEP, $P = 3000$ e F_d .

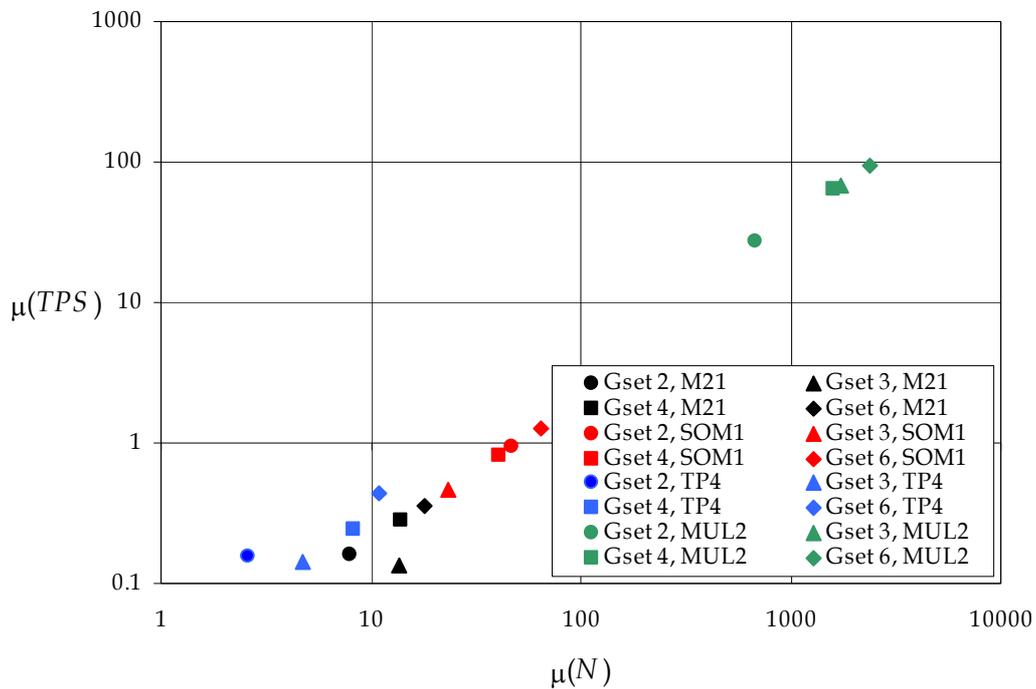


Figura 6.10 - $\mu(TPS)$ versus $\mu(N)$ para o algoritmo OEP, $P = 3000$ e F_d .

3.3. Função de aptidão estática versus dinâmica

Os gráficos da figura 6.11 comparam as funções estática e dinâmica (F_e com F_d) na síntese de quatro circuitos lógicos combinatórios (M21, SOM1, TP4 e MUL2), em termos de $\mu(N)$ e $\sigma(N)$.

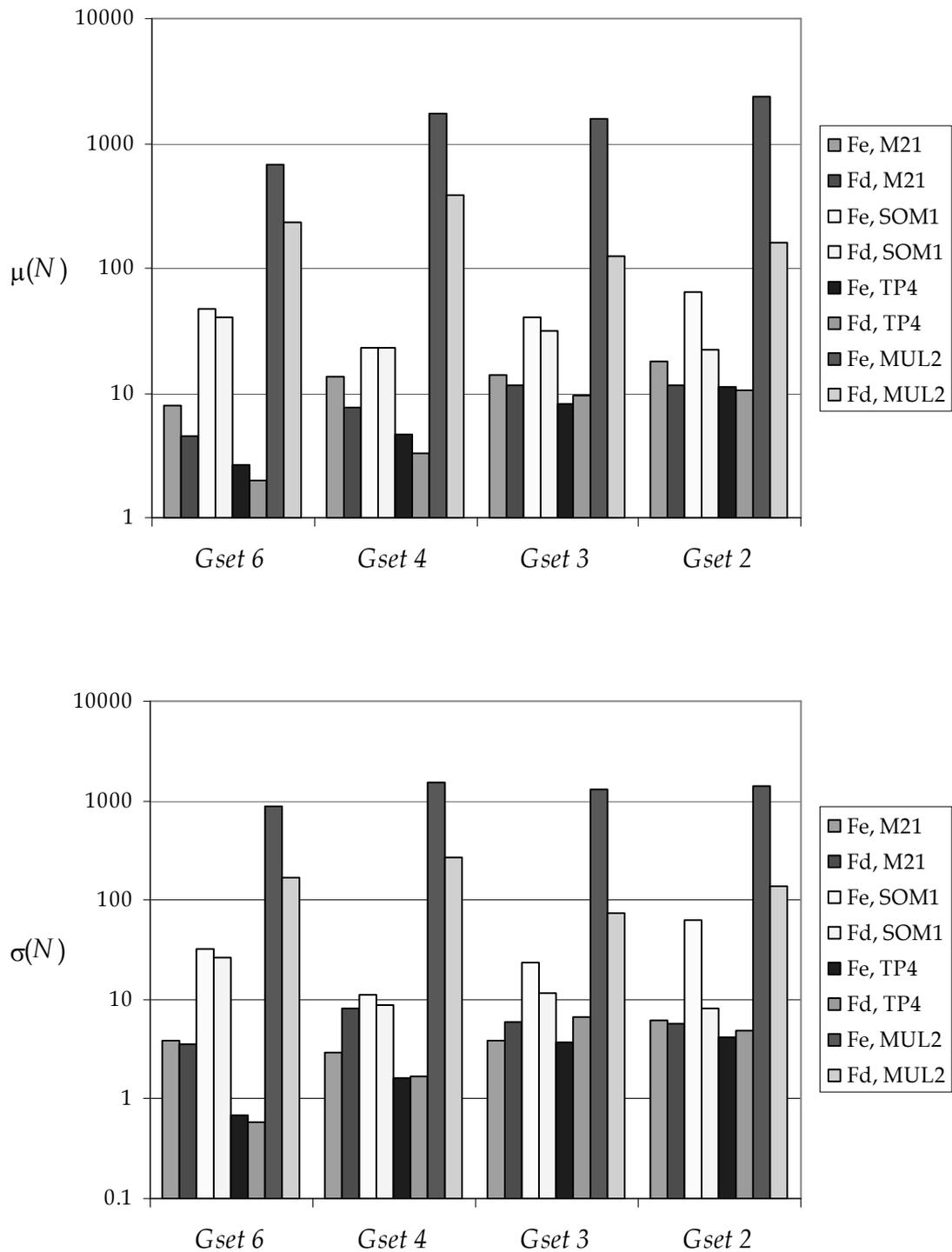


Figura 6.11 - $\mu(N)$ e $\sigma(N)$ para o algoritmo OEP e $P = 3000$, com F_e e F_d .

Da análise dos dois gráficos da figura 6.11 é possível constatar que F_d apresenta resultados superiores a F_e em termos de $\mu(N)$ para todos os circuitos e todos os conjuntos de portas lógicas, excepto em dois casos concretos: na síntese do circuito SOM1 com o *Gset* 4 e na síntese do circuito TP4 com o *Gset* 3. Relativamente a $\sigma(N)$ pode-se, da mesma forma, afirmar a superioridade de F_d quando comparada com F_e , excepto nos casos: síntese dos circuitos M21 e TP4 com o *Gset* 4, síntese do circuito M21 com o *Gset* 3 e síntese do circuito TP4 com o *Gset* 2.

3.4. Comparação dos algoritmos genético, memético e OEP

Esta sub-secção dedica-se à análise comparativa entre os três algoritmos desenvolvidos no âmbito desta tese: o algoritmo genético, o memético e o de optimização por enxames de partículas.

3.4.1. Índices $\mu(N)$, $\sigma(N)$ e $\mu(TPS)$

Inicia-se o estudo de comparação pela análise de $\mu(N)$, $\sigma(N)$ e $\mu(TPS)$, respectivamente, média do número de gerações necessárias para obter a solução, desvio padrão do número de gerações para obter a solução e média do tempo de processamento para obter a solução (Reis, Machado e Boaventura, 2006a). Estas experiências foram executadas com os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, com $P = 3000$, para os algoritmos AG e AM.

A figura 6.12 mostra o desvio padrão $\sigma(N)$ *versus* a média, do número de gerações $\mu(N)$ necessárias para se obter a solução, para os circuitos {M21,

SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, com $P = 3000$, para os algoritmos AG e AM.

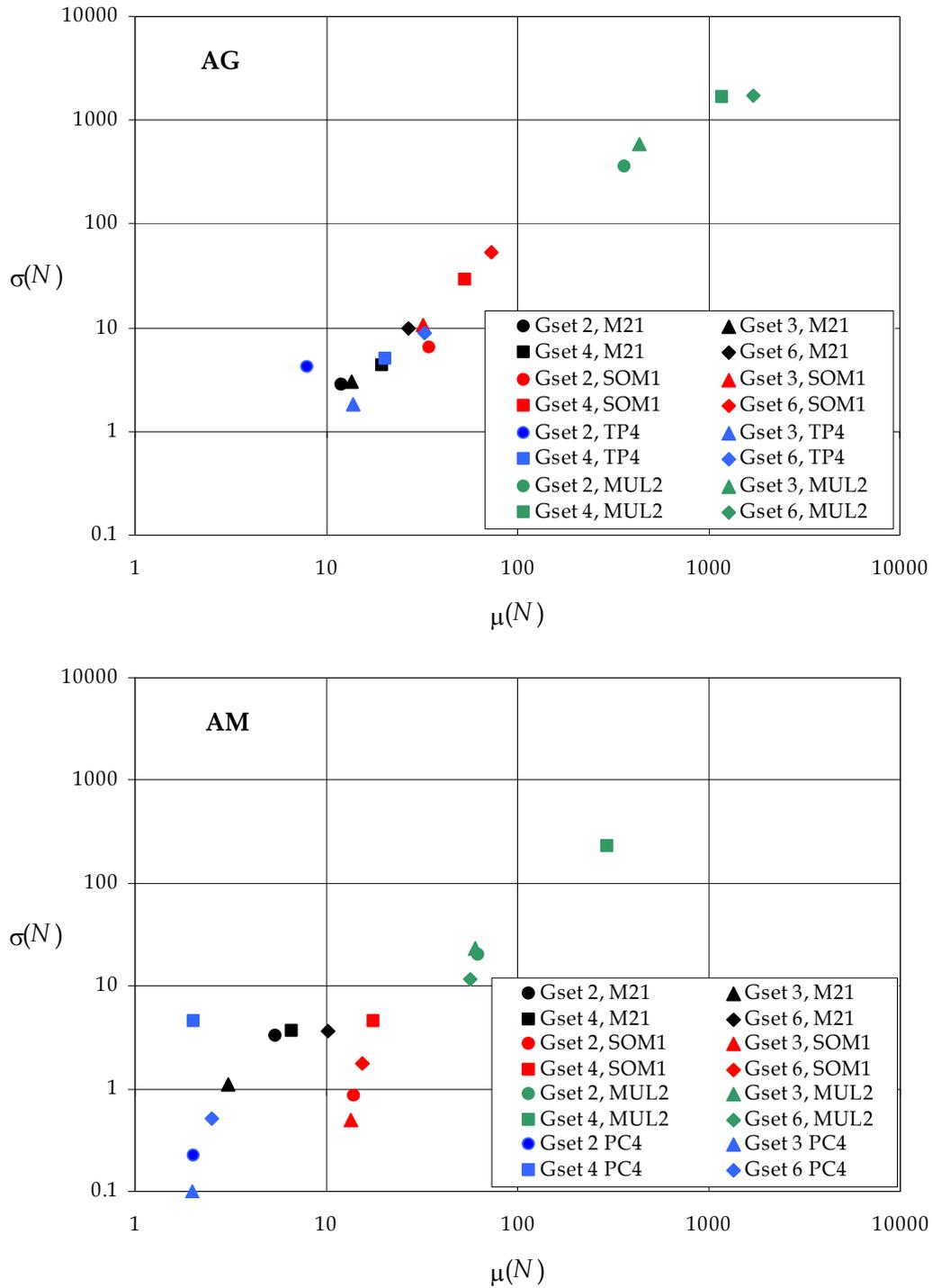


Figura 6. 12– $\sigma(N)$ versus $\mu(N)$ com $P = 3000$ e $F_e (\delta = 0)$ para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para os algoritmos AG e AM.

Pela análise das figuras 6.2 e 6.12, é possível constatar que o circuito MUL2 é o circuito mais complexo, enquanto os circuitos M21 e TP4 são os circuitos de menor grau de complexidade. Verifica-se também que, com o AG, o conjunto de portas lógicas $G_{set 6}$ é o menos eficiente para todos os circuitos testados. A observação pormenorizada destas duas figuras revela que os traçados dos gráficos seguem uma lei potência do tipo:

$$\sigma(N) = a[\mu(N)]^b \quad a, b \in \mathfrak{R} \quad (6.13)$$

Em termos dos índices $\sigma(N)$ versus $\mu(N)$, o AM apresenta os melhores resultados, para todos os circuitos e todos os conjuntos de portas lógicas. No que diz respeito aos outros dois algoritmos, o OEP é superior (inferior) ao AG para circuitos complexos (simples).

A figura 6.13 ilustra a média do tempo de processamento para obter a solução $\mu(TPS)$ versus a média do número de gerações $\mu(N)$ para obter a solução para os algoritmos {AG, AM}, para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}.

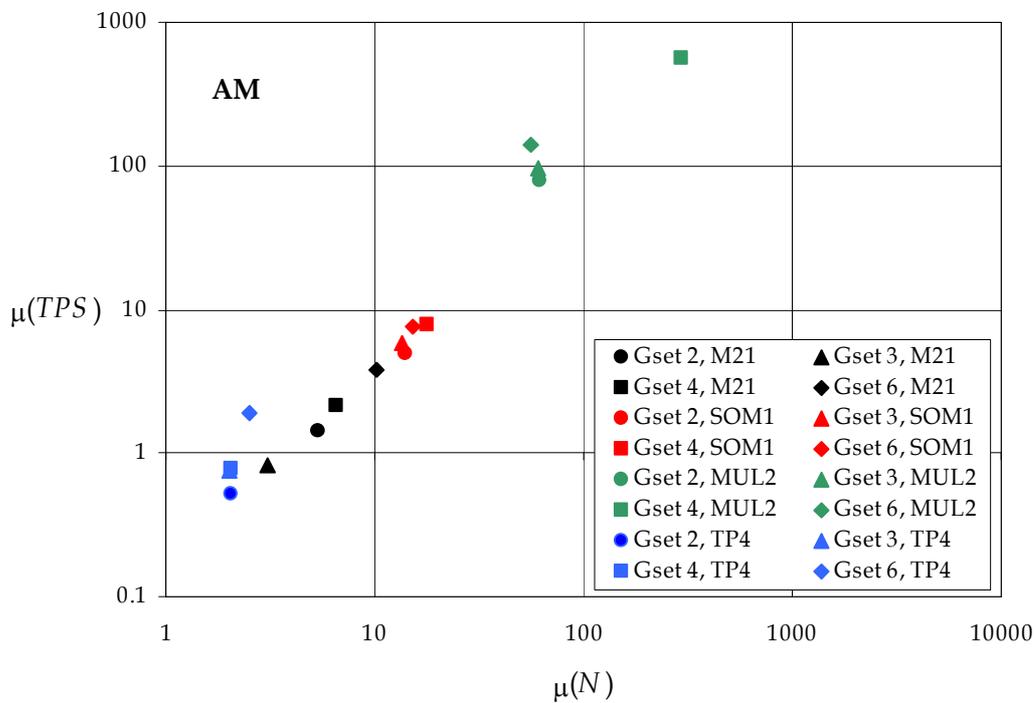
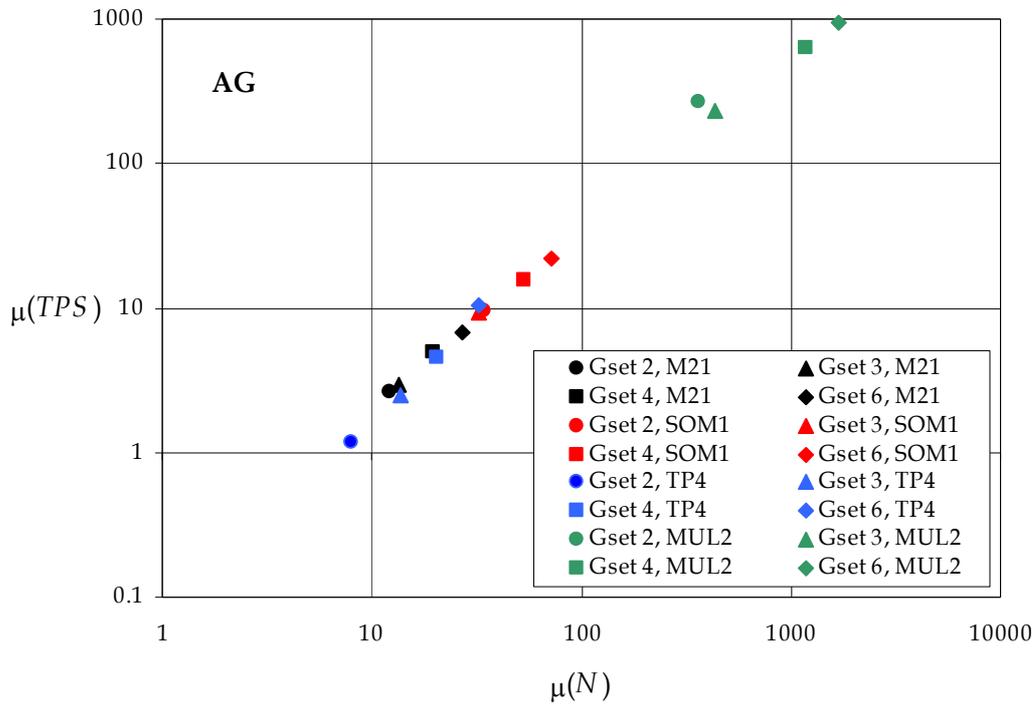


Figura 6. 13– $\mu(TPS)$ versus $\mu(N)$ com $P = 3000$ e $F_e (\delta = 0)$ para os circuitos {M21, SOM1, TP4, MUL2} e os conjuntos de portas lógicas {2, 3, 4, 6}, para os algoritmos AG e AM.

Analisando os gráficos das figuras 6.3 e 6.13 é evidente que o algoritmo OEP demonstra ser cerca de dez vezes mais rápido que os algoritmos AM e AG.

Mais uma vez, os traçados identificam-se com uma lei potência do tipo:

$$\mu(TPS) = c[\mu(N)]^d \quad c, d \in \mathfrak{R} \quad (6.14)$$

A tabela 6.1 contém os valores numéricos dos parâmetros (a, b) e (c, d) dos três algoritmos.

Tabela 6. 1– Parâmetros (a, b) e (c, d) das equações 6.13 e 6.14

Algoritmo	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
AG	0,0365	1,602	0,1526	1,1734
AM	0,0728	1,2602	0,2089	1,3587
OEP	0,2677	1,1528	0,0141	1,1233

A figura 6.14 apresenta as linhas de tendência para os três algoritmos, com quais se obtiveram os valores dos parâmetros ilustrados na tabela 6.1.

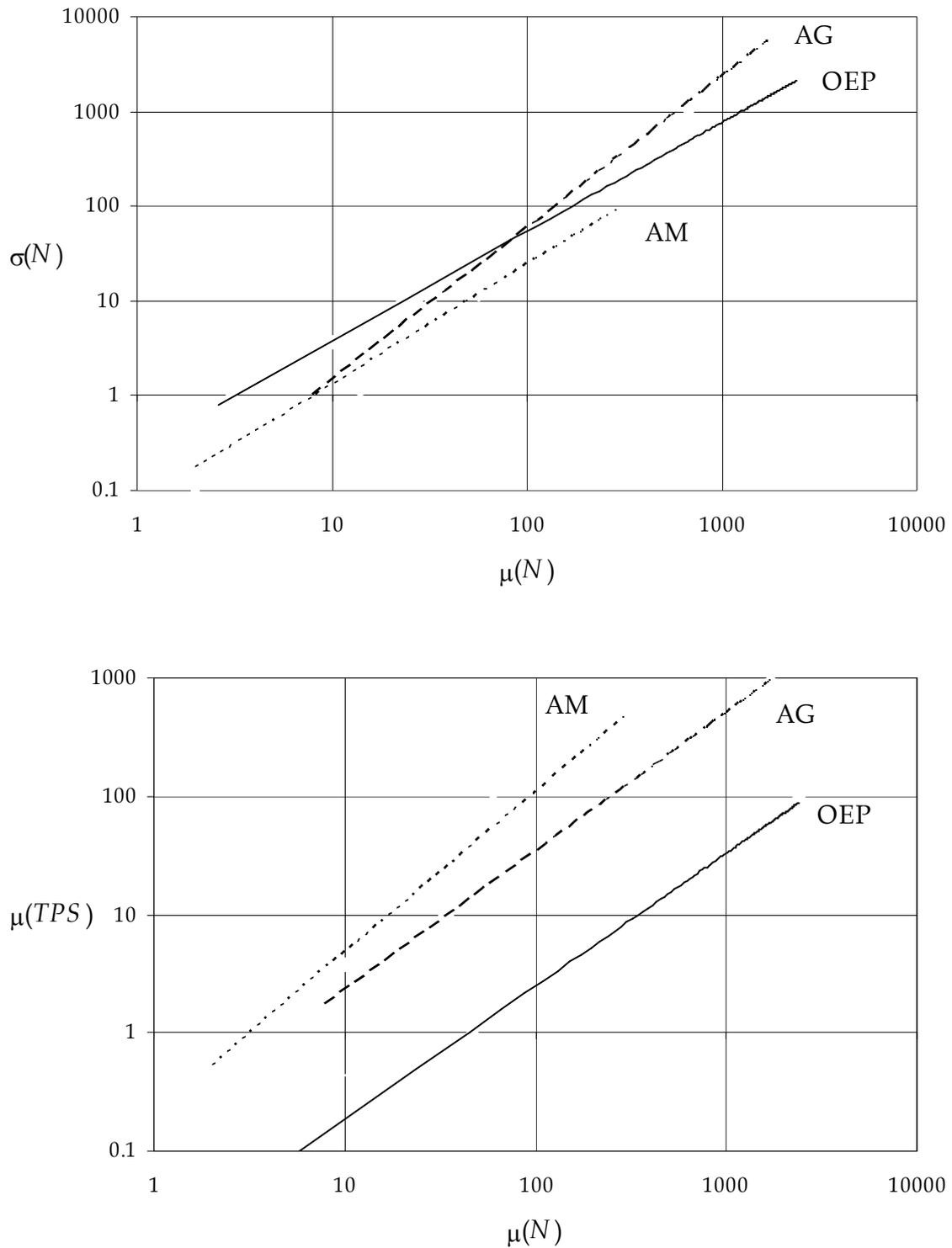


Figura 6. 14– Linhas de tendência para $\sigma(N)$ versus $\mu(N)$ e $\mu(TPS)$ versus $\mu(N)$ para os algoritmos AG, AM e OEP.

No que diz respeito ao tempo de processamento para se obter a solução comprova-se que o algoritmo OEP conduz a melhores valores.

3.4.2. Tamanho da população

Estudou-se também a influência do tamanho da população P em cada um dos algoritmos, genético, memético e optimização por enxames de partículas na síntese dos circuitos lógicos combinatórios (M21, SOM1, SUB1, TP4) com os conjuntos de portas lógicas {2, 3, 4, 6} (Reis, Machado e Boaventura, 2006a). As experiências foram desenvolvidas para valores de $P = 100, 500, 1000$ e 3000 .

As figuras 6.15 a 6.18 apresentam os resultados obtidos em termos do $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para os circuitos {M21, SOM1, SUB1, TP4} com os conjuntos de portas lógicas {2, 3, 4, 6}, os algoritmos {AG, AM, OEP} e para $P = 100, 500, 1000, 3000$.

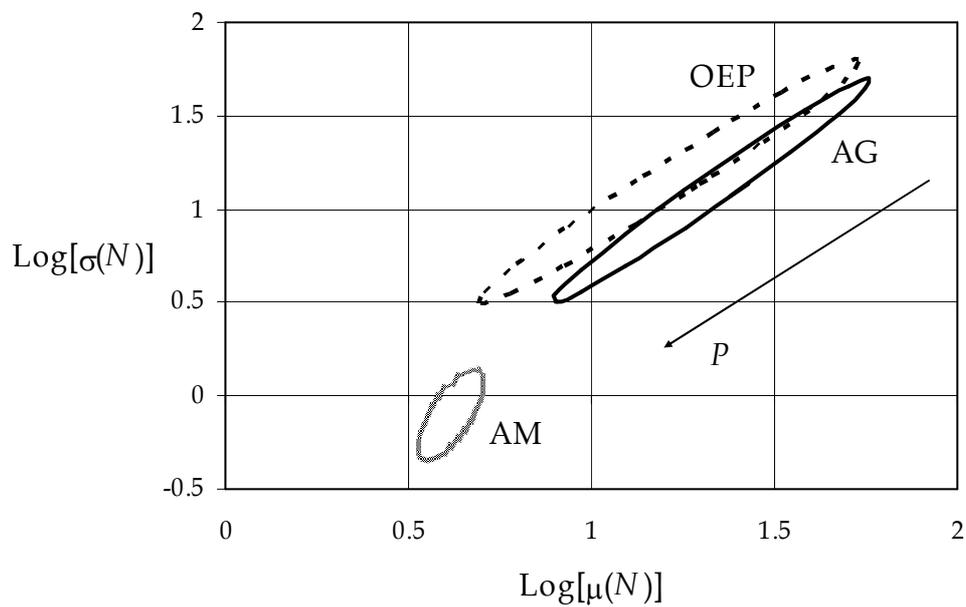


Figura 6. 15– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito M21 com $P = \{100, 500, 1000, 3000\}$

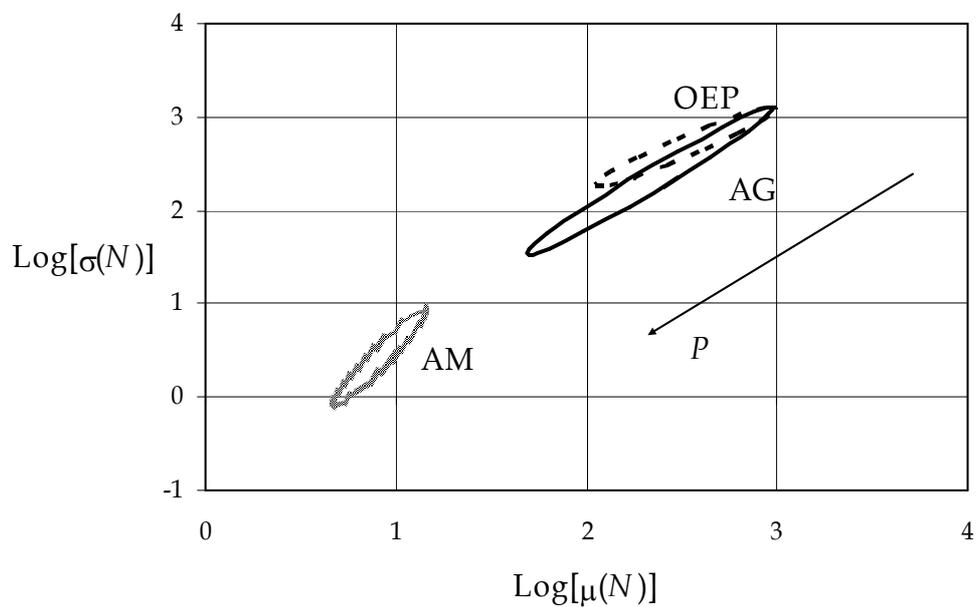


Figura 6. 16– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito SOM1 com $P = \{100, 500, 1000, 3000\}$ e os algoritmos {AG, AM, OEP}.

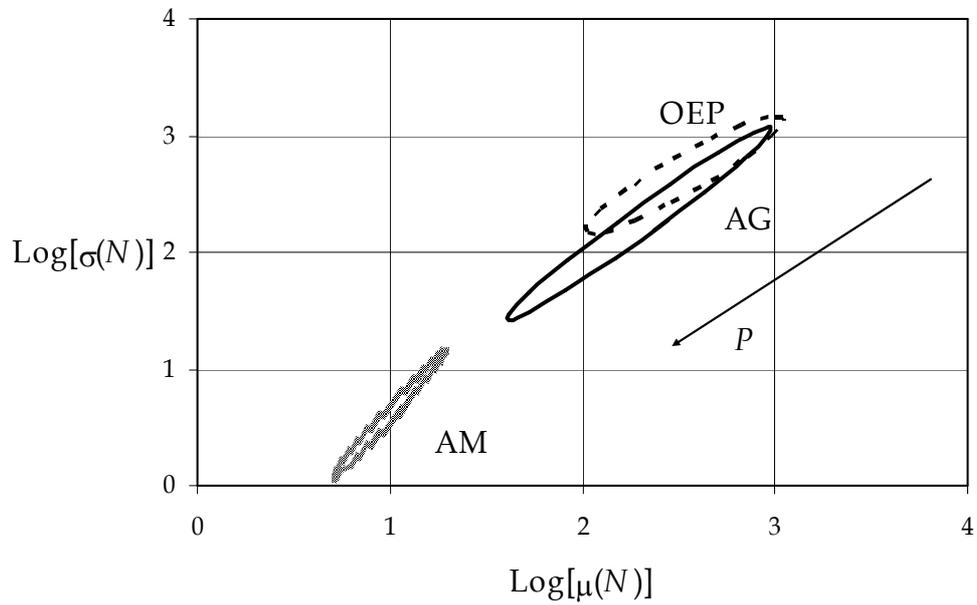


Figura 6. 17– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito SUB1 com $P = \{100, 500, 1000, 3000\}$ e os algoritmos {AG, AM, OEP}.

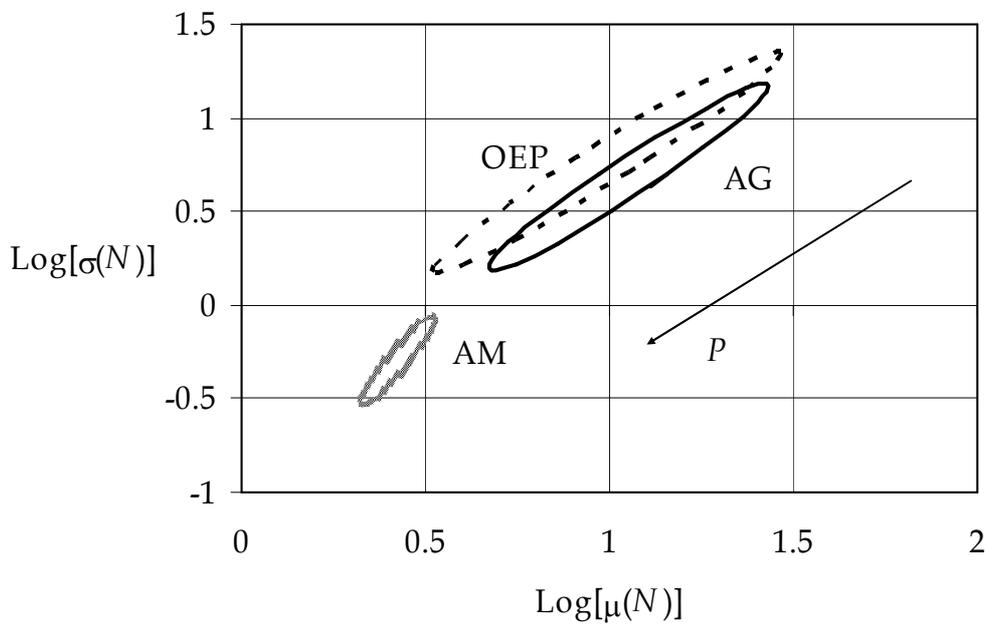


Figura 6. 18– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o circuito TP4 com $P = \{100, 500, 1000, 3000\}$ e os algoritmos {AG, AM, OEP}.

Os pontos no espaço $\{\text{Log}[\sigma(N)] \text{ versus } \text{Log}[\mu(N)]\}$ foram aproximados pela distribuição de probabilidade Gaussiana bidimensional. As elipses que podem ser visualizadas nos gráficos correspondem à delimitação dos pontos em cada caso.

Como se pode observar através dos gráficos representados nas figuras anteriores, o algoritmo memético é o que revela melhor desempenho para todos os circuitos lógicos e que $\mu(N)$ e $\sigma(N)$ variam inversamente com o tamanho da população P . Os algoritmos genético e optimização por enxames de partículas apresentam resultados muito similares, em particular para os circuitos M21 e TP4. Para os circuitos SOM1 e SUB1 o algoritmo OEP é menos sensível a P quando comparado com o AG.

3.4.3. Complexidade dos circuitos

Esta sub-secção apresenta um estudo que permitiu a classificação dos circuitos lógicos combinatórios analisados quanto ao seu grau de complexidade.

As figuras 6.19 a 6.21 apresentam $\text{Log}[\sigma(N)] \text{ versus } \text{Log}[\mu(N)]$ com $P = 100, 500, 1000, 3000$ para os circuitos {M21, SOM1, SUB1, TP4} com os conjuntos de portas lógicas {2, 3, 4, 6} e para os algoritmos {AG, AM, OEP}.

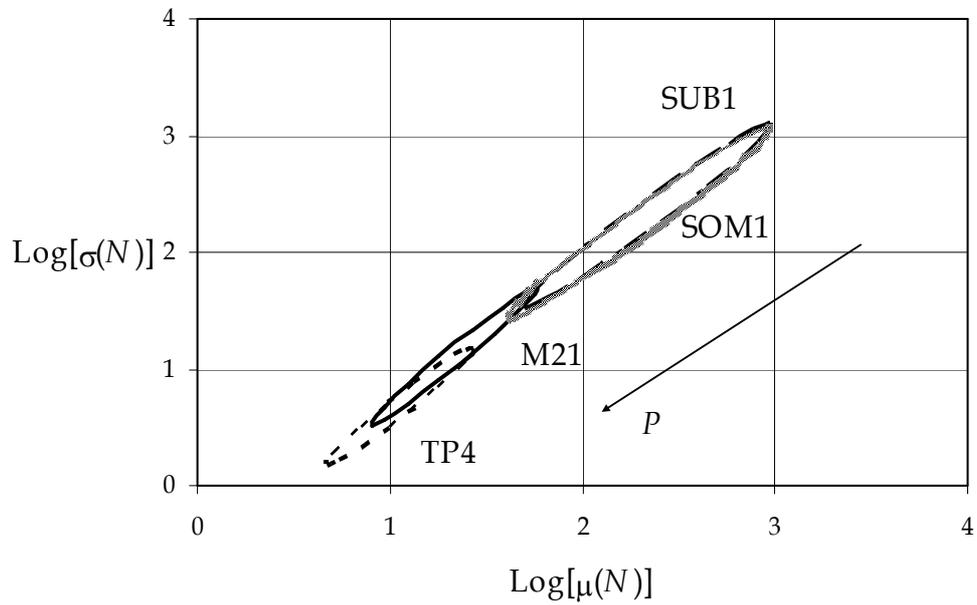


Figura 6. 19– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o AG com $P = \{100, 500, 1000, 3000\}$ e para os circuitos $\{M21, SOM1, SUB1, TP4\}$.

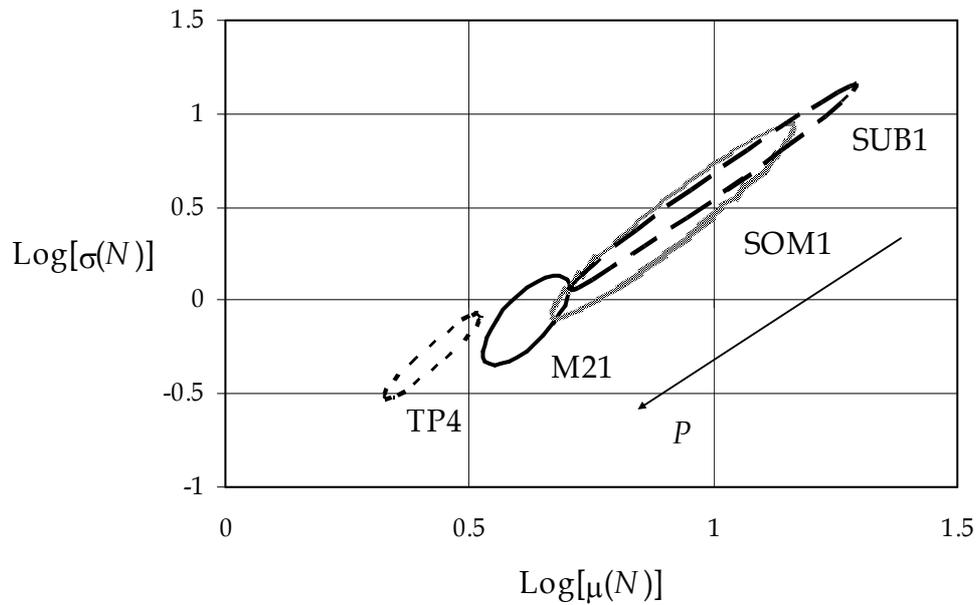


Figura 6. 20– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o AM com $P = \{100, 500, 1000, 3000\}$ e para os circuitos $\{M21, SOM1, SUB1, TP4\}$.

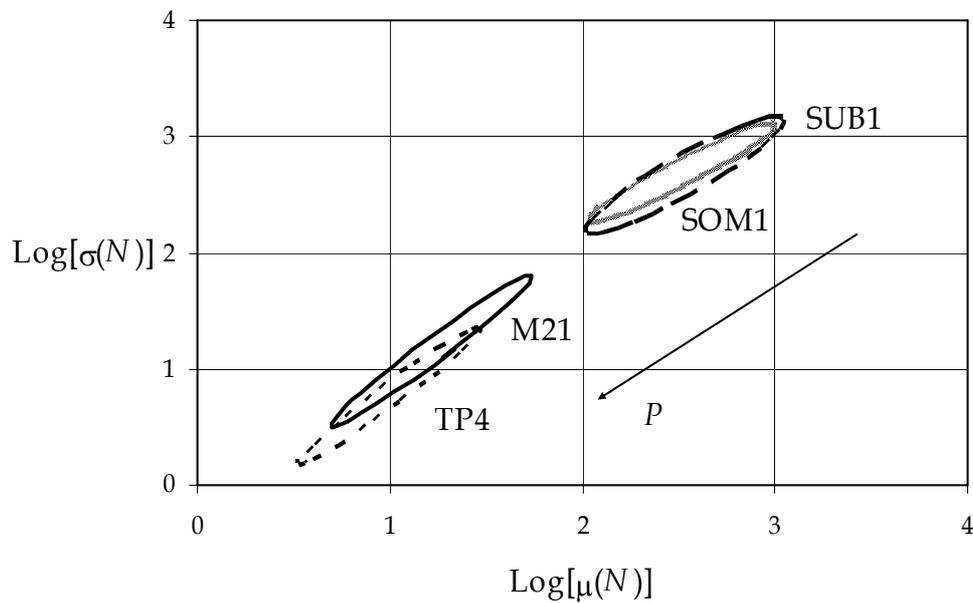


Figura 6. 21– $\text{Log}[\sigma(N)]$ versus $\text{Log}[\mu(N)]$ para o OEP com $P = \{100, 500, 1000, 3000\}$ e para os circuitos $\{M21, SOM1, SUB1, TP4\}$.

Analisando os gráficos anteriores é possível classificar a complexidade dos circuitos lógicos combinatórios por algoritmo evolutivo. Para os três algoritmos a sequência de aumento de complexidade dos circuitos é: TP4, M21, SOM1, SUB1. No caso do OEP, a complexidade dos circuitos está claramente dividida em duas áreas, a área dos circuitos SOM1, SUB1 e a área dos circuitos M21, TP4.

3.4.4. Problema de escala

As figuras 6.22 e 6.23 mostram a evolução de $\sigma(N)$ versus $\mu(N)$ e de $\mu(TPS)$ versus $\mu(N)$, respectivamente, para a família de circuitos de teste de paridade, quando se exige um número crescente de bits. A família de circuitos de teste de paridade considerada consiste em $\{2, 3, 4, 5, 6\}$ bit.

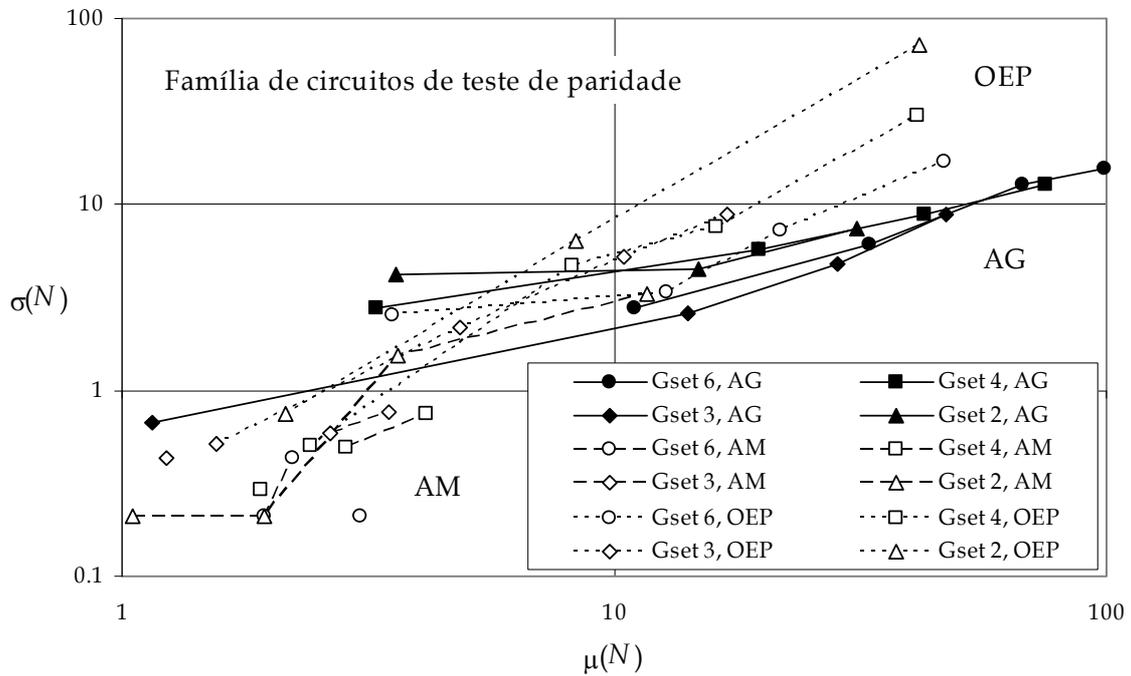


Figura 6.22 - $\sigma(N)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para os algoritmos AG, AM e OEP e para os conjuntos de portas lógicas {2, 3, 4, 6} e $P = 3000$.

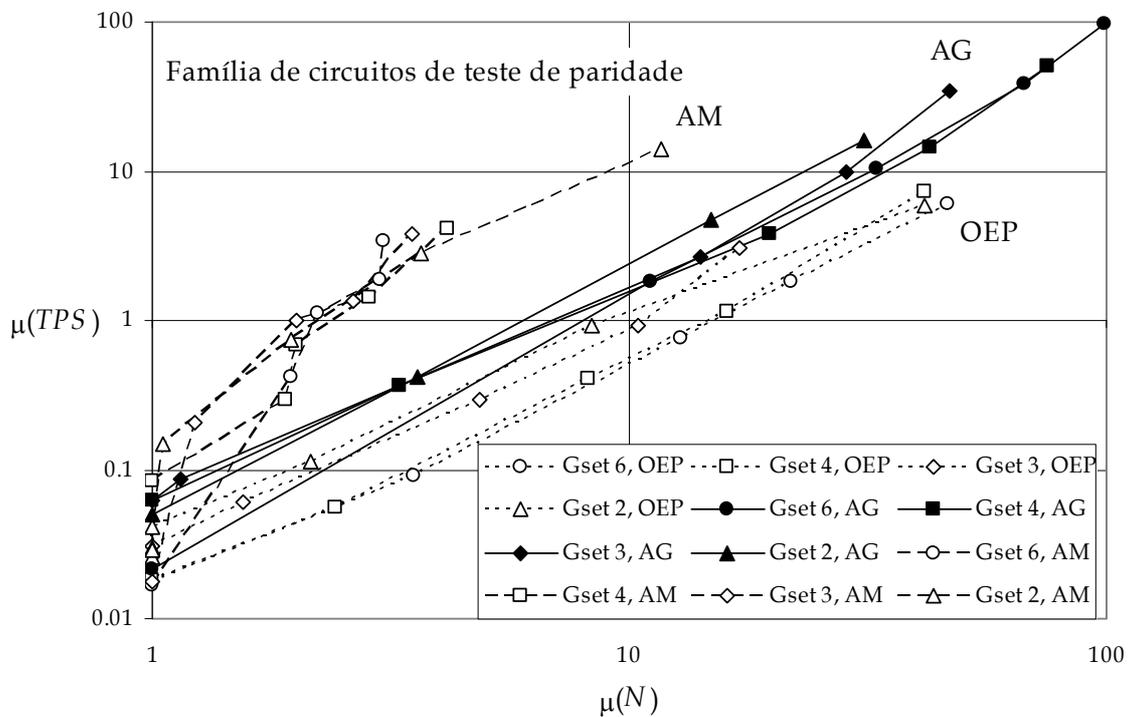


Figura 6.23 - $\mu(TPS)$ versus $\mu(N)$ para a família de circuitos de teste de paridade, para os algoritmos AG, AM e OEP e para os conjuntos de portas lógicas {2, 3, 4, 6} e $P = 3000$.

3.4.5. Função de aptidão sequencial

Esta sub-seção analisa os algoritmos AG, AM e OEP, utilizando a função de aptidão estática F_e , descrita anteriormente na equação 6.10. Dado que F_e incorpora duas fases estuda-se o seu impacto. Assim, no primeiro conjunto de experiências os algoritmos avaliam os indivíduos com a função de aptidão que usa apenas f_1 . No segundo conjunto de experiências, a função de aptidão avalia f_1 e f_2 em sequência.

A figura 6.24 mostra o desvio padrão $\sigma(N)$ versus a média do número de gerações $\mu(N)$ para o circuito SOM1, para todos os conjuntos de portas lógicas, para os três algoritmos, utilizando as duas possibilidades da função de aptidão F_e .

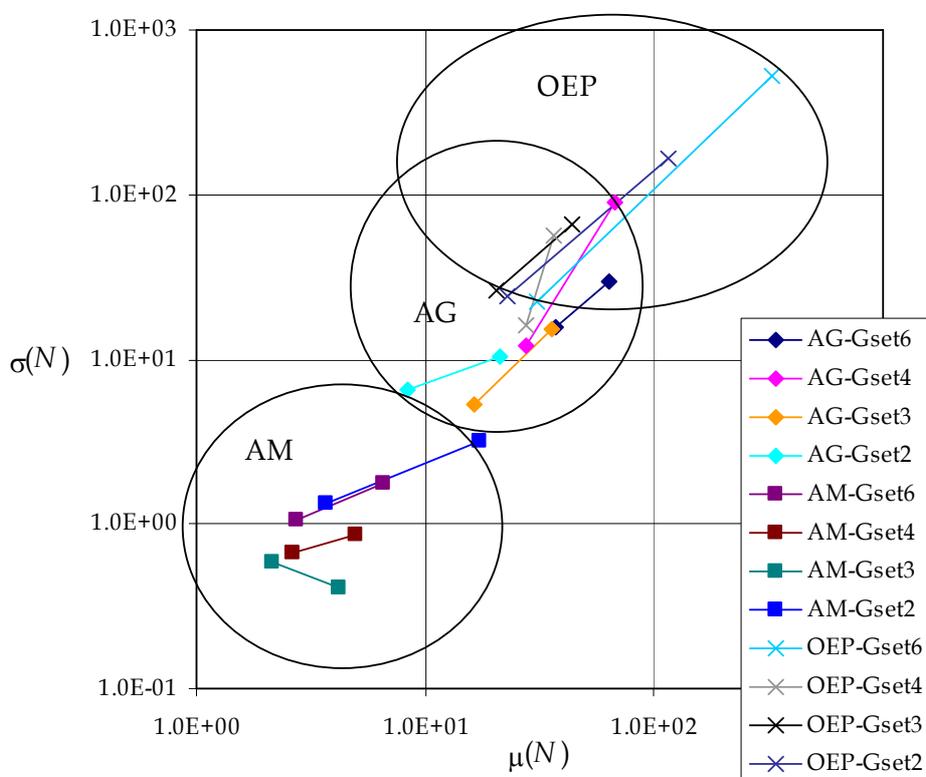


Figura 6.24 – Desvio padrão $\sigma(N)$ versus a média do número de gerações $\mu(N)$ para obter a solução para o circuito SOM1, com os algoritmos {AG, AM, OEP} usando a função de aptidão com f_1 e com $f_1 + f_2$.

Cada círculo na figura 6.24 contém os resultados para um algoritmo específico. Cada segmento de recta tem os dois valores que correspondem aos resultados com f_1 (à esquerda do segmento) e com $f_1 + f_2$ (à direita do segmento) na avaliação da função de aptidão.

Aplicando uma equação de ajuste aos resultados, verifica-se que $\sigma(N)$ segue uma lei potência com respeito a $\mu(N)$, de acordo com a equação:

$$\sigma(N) = 0,1497 \mu(N)^{1,4321} \quad (6.15)$$

A figura 6.25 mostra o desvio padrão $\sigma(N)$ *versus* a média do número de gerações para obter a solução $\mu(N)$ para o circuito SUB1, para todos os conjuntos de portas lógicas e para os três algoritmos, utilizando as duas possibilidades da função de aptidão F_e .

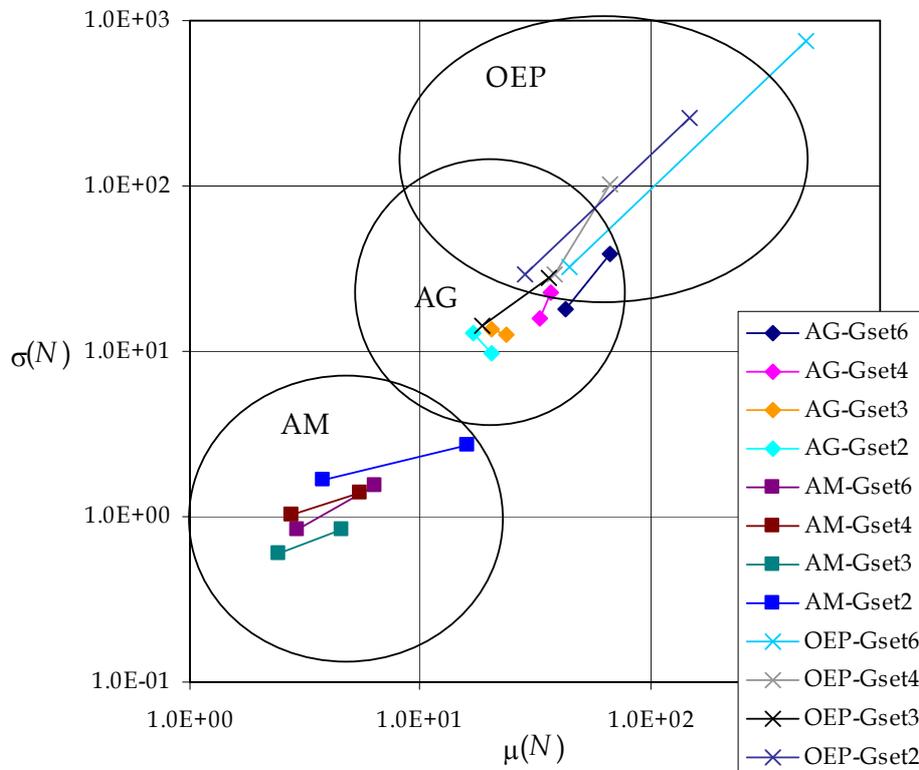


Figura 6. 25 – Desvio padrão $\sigma(N)$ *versus* a média do número de gerações $\mu(N)$ para obter a solução para o circuito SUB1, com os algoritmos {AG, AM, OEP} usando a função de aptidão com f_1 e com $f_1 + f_2$.

Uma vez mais $\sigma(N)$ segue uma lei potência em relação a $\mu(N)$ de acordo com a expressão:

$$\sigma(N) = 0,1663\mu(N)^{1,3887} \quad (6.16)$$

Analisando as equações 6.15 e 6.16, constata-se que, em ambas, o expoente é superior a 1, o que revela ter $\sigma(N)$ uma maior sensibilidade do que $\mu(N)$, relativamente à inclusão de f_2 na função de aptidão F_e .

Os traçados dos gráficos das figuras 6.24 e 6.25 indicam uma pior convergência quando se aplica a função de aptidão constituída pelas duas componentes $f_1 + f_2$. A tabela 6.2 apresenta a razão entre os dois casos em teste, *i. e.*: $\mu(N)]_{f_1+f_2} / \mu(N)]_{f_1}$ e $\sigma(N)]_{f_1+f_2} / \sigma(N)]_{f_1}$. Como se verifica, na maioria dos casos, temos razões superiores a 1.

Tabela 6. 2 - Razões $\mu(N)]_{f_1+f_2} / \mu(N)]_{f_1}$ e $\sigma(N)]_{f_1+f_2} / \sigma(N)]_{f_1}$ para as funções de aptidão sequenciais.

Circuito Aritmético	Algoritmo	Conjunto	$\mu(N)]_{f_1+f_2} / \mu(N)]_{f_1}$	$\sigma(N)]_{f_1+f_2} / \sigma(N)]_{f_1}$
SOM1	AG	Gset 6	1,69	1,90
		Gset 4	2,45	7,38
		Gset 3	2,19	2,86
		Gset 2	2,56	1,59
	AM	Gset 6	2,36	1,65
		Gset 4	1,89	1,28
		Gset 3	1,95	0,70
		Gset 2	4,70	2,39
			Gset 6	10,64

SUB1	OEP	Gset 4	1,33	3,48
		Gset 3	2,15	2,55
		Gset 2	5,04	6,84
	AG	Gset 6	1,55	2,20
		Gset 4	0,90	0,71
		Gset 3	1,15	0,93
		Gset 2	1,21	0,75
	AM	Gset 6	2,15	1,84
		Gset 4	1,98	1,36
		Gset 3	1,86	1,38
		Gset 2	4,21	1,62
	OEP	Gset 6	10,79	23,62
Gset 4		1,72	3,52	
Gset 3		0,51	0,52	
Gset 2		5,19	8,59	

3.4.5.1. Problema de Escala

Tal como já foi objecto de análise também vai ser abordado o problema de escala na síntese de circuitos, nomeadamente para o caso da utilização da função de aptidão dividida em duas fases. Considera-se, para o efeito, a família de circuitos de teste de paridade com $\{2, 3, 4, 5, 6\}$ bit.

As figuras 6.26 a 6.31 exibem os índices $\mu(N)$ e $\sigma(N)$ para a família de circuitos de teste de paridade e os algoritmos $\{AG, AM, OEP\}$.

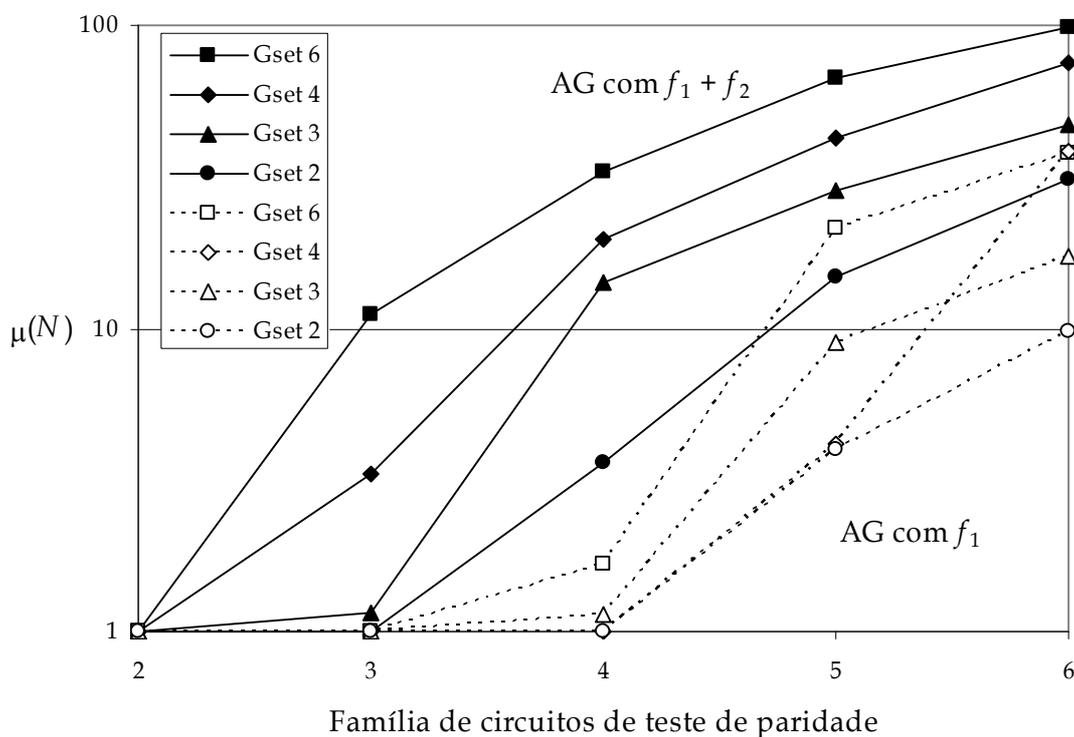


Figura 6. 26 - $\mu(N)$ para a família de circuitos de teste de paridade de {2, 3, 4, 5, 6} bit com o AG, aplicando a função de aptidão com f_1 e com $f_1 + f_2$.

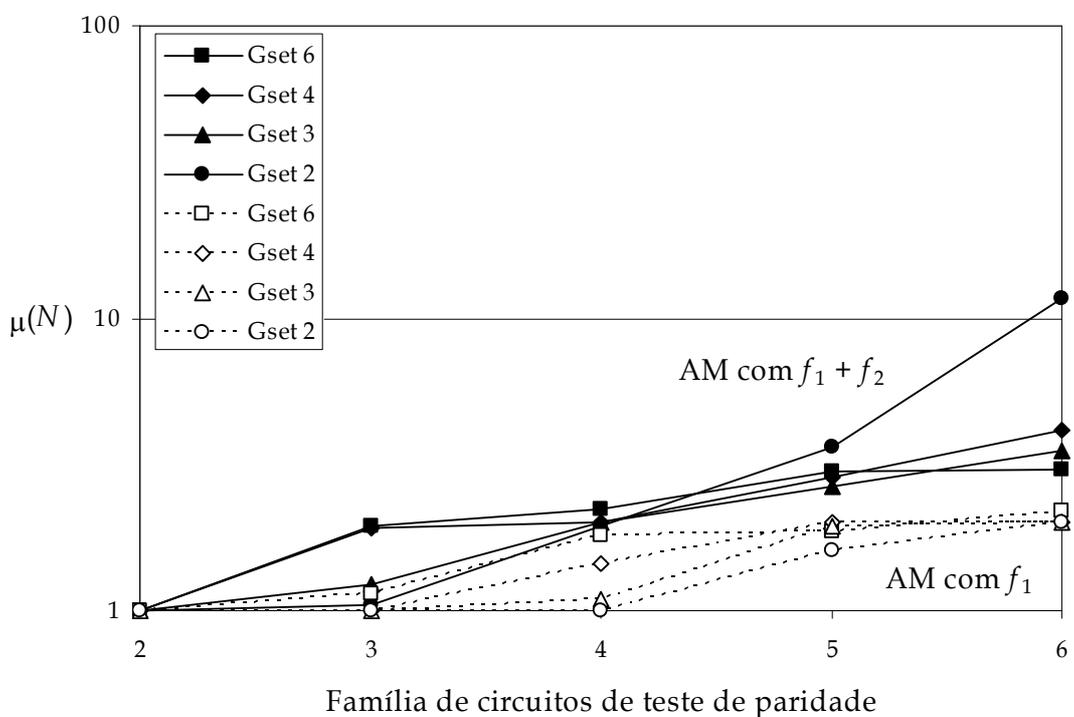


Figura 6. 27 - $\mu(N)$ para a família de circuitos de teste de paridade de {2, 3, 4, 5, 6} bit com o AM, aplicando a função de aptidão com f_1 e com $f_1 + f_2$.

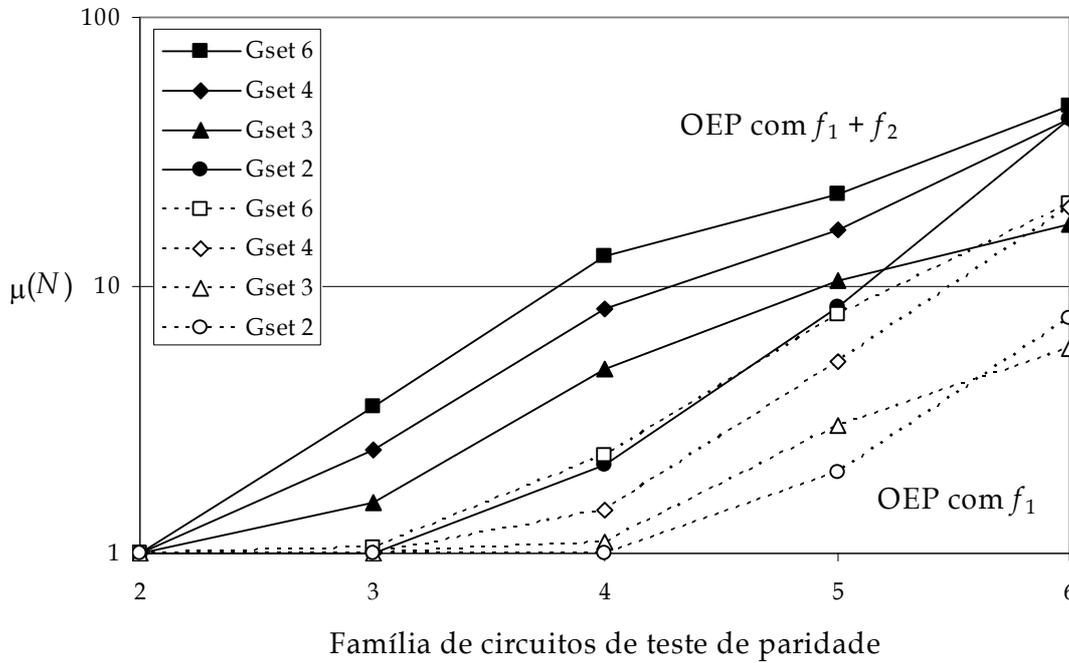


Figura 6. 28 - $\mu(N)$ para a família de circuitos de teste de paridade de {2, 3, 4, 5, 6} bit com o OEP, aplicando a função de aptidão com f_1 e com $f_1 + f_2$.

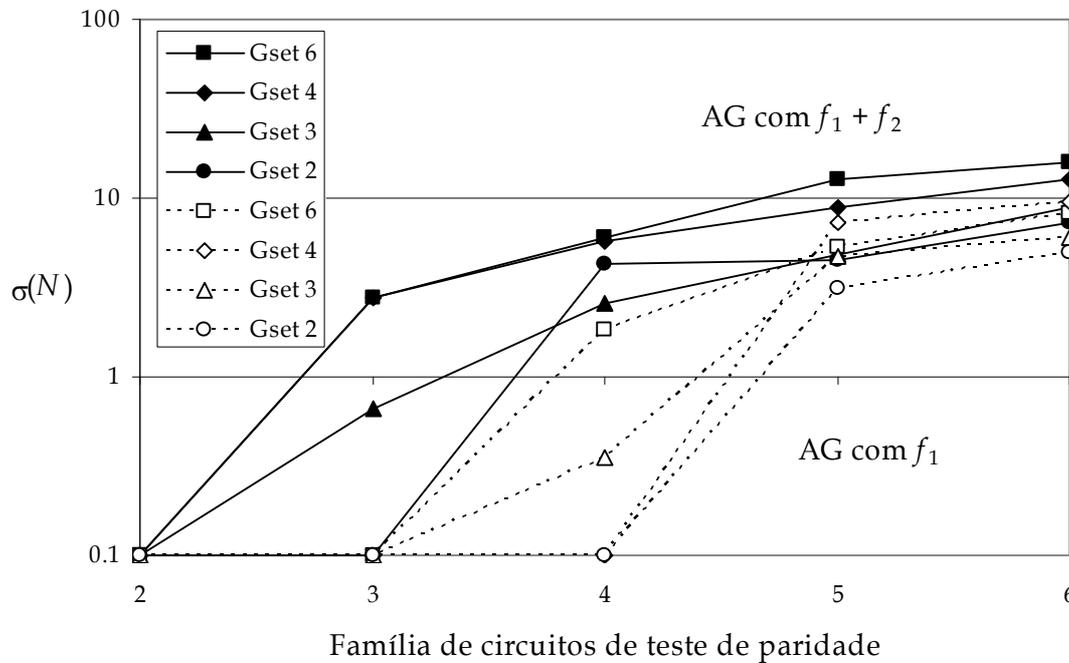


Figura 6. 29 - $\sigma(N)$ para a família de circuitos de teste de paridade de {2, 3, 4, 5, 6} bit com o AG, aplicando a função de aptidão com f_1 e com $f_1 + f_2$.

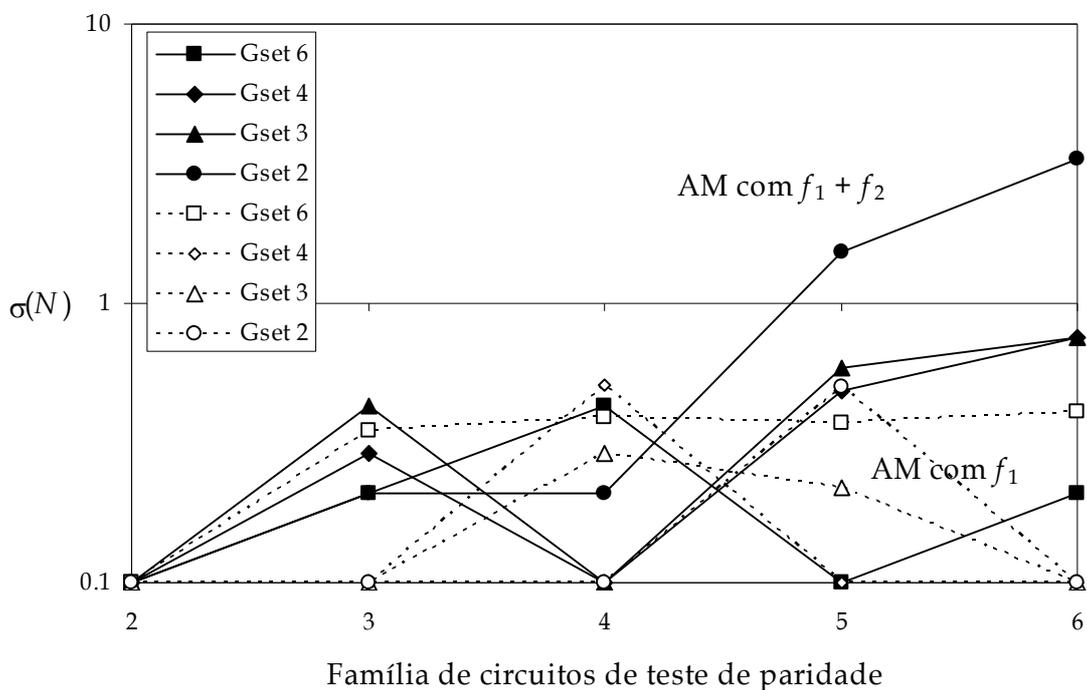


Figura 6.30 - $\sigma(N)$ para a família de circuitos de teste de paridade de {2, 3, 4, 5, 6} bit com o AM, aplicando a função de aptidão com f_1 e com $f_1 + f_2$.

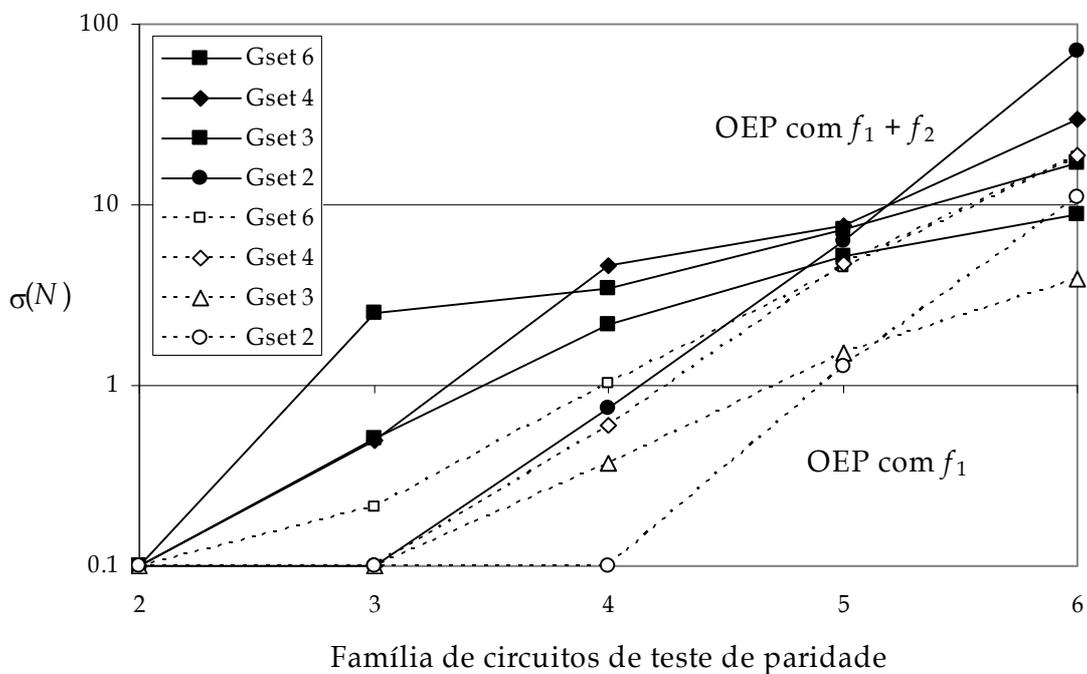


Figura 6.31 - $\sigma(N)$ para a família de circuitos de teste de paridade de {2, 3, 4, 5, 6} bit com o OEP, aplicando a função de aptidão com f_1 e com $f_1 + f_2$.

Analisando as figuras 6.26 a 6.31 verifica-se que, para os três algoritmos e para a família de circuitos de teste de paridade, a função de aptidão sequencial necessita de requisitos computacionais mais elevados devido aos valores superiores de $\mu(N)$ e de $\sigma(N)$.

4. Resumo do capítulo

Este capítulo apresenta a Inteligência dos Enxames (IE), mais especificamente um algoritmo de OEP para a síntese de circuitos lógicos combinatórios. A IE baseia-se na propriedade de um sistema se adaptar, ou seguir, comportamentos de grupo de agentes que interagem localmente com o seu ambiente, fornecendo a base através da qual é possível explorar a resolução colectiva (ou distribuída) de problemas sem controlo centralizado ou fundamentado num modelo global. O OEP é um algoritmo de pesquisa baseado numa população inicial gerada aleatoriamente. Neste esquema cada partícula voa, no espaço de pesquisa, com uma velocidade que é ajustada dinamicamente de acordo com o seu historial de comportamento, o que determina que voe tendencialmente em direcção à melhor área de pesquisa ao longo de todo o processo.

O algoritmo OEP demonstrou ser uma técnica de computação evolutiva bem adaptada ao desenho de circuitos digitais dado o seu muito bom desempenho em termos de tempo de processamento, quer pelo reduzido número de gerações necessárias quer pela simplicidade dos cálculos que permite uma rápida convergência mesmo quando o número de gerações é mais elevado.

Os resultados obtidos neste capítulo revelaram que o tamanho da população tem uma influência significativa no desempenho dos algoritmos estudados e

que $\text{Log}[\sigma(N)]$ - desvio padrão do número de gerações necessárias para atingir uma solução - tem uma dependência linear com $\text{Log}[\mu(N)]$ - média do número de gerações necessárias para atingir uma solução - ou seja, $\text{Log}[\sigma(N)] \sim \text{Log}[\mu(N)]$. Este facto evidencia um comportamento de natureza mais determinística para o OEP e um comportamento desejável dado que para uma baixa média do número de gerações a dispersão também se reduz.

Foi adoptada uma metodologia que permite um esquema de classificação dos circuitos lógicos combinatórios em termos do seu grau de complexidade.

Referências

- Clerc, M. e J. Kennedy, J. The Particle Swarm: explosion, stability, and convergence in a multi-dimensional complex space. In IEEE Transactions on Evolutionary Computation, vol. 6, pp. 58-73, 2002.
- Kennedy, J. e Eberhart, R. C. Particle Swarm Optimization. In Proceedings of the IEEE International Conference in Neural Networks, pp 1942-1948, November, 1995.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Evolutionary Techniques in Circuit Design and Optimization", WSEAS Transactions on Power Systems, Issue 7, Vol. 1, pp.1337-1342, July, 2006a.
- Reis, C., Tenreiro Machado, J. A., Galhano, A. e Boaventura Cunha, J. Circuit Synthesis Using Particle Swarm Optimization, IEEE - ICCS 2006, International Conference on Computational Cybernetics, Tallinn, Estonia, pp 149-154, August, 2006b.
- Reis, C., Tenreiro Machado, J. A. e Boaventura Cunha, J. Arithmetic Circuits Design Using Swarm Intelligence, WACI 2006 - Workshop on Applications of Computational Intelligence, Coimbra, Portugal, November, 2006c.
- Shi, Y. e Eberhart, R. C. A Modified Particle Swarm Optimizer. In Proceedings. of the 1998 International Conference on Evolutionary Computation, pp. 69-73, May, 1998.

Capítulo 7

CONCLUSÕES

Introdução

Este capítulo apresenta uma síntese do trabalho desenvolvido no decorrer da tese, expõe as principais conclusões e indica as perspectivas de desenvolvimento futuro. Assim, a secção 1 descreve os aspectos mais relevantes do trabalho realizado, a secção 2 faz a síntese conclusiva e a secção 3 aponta as direcções futuras.

1. Trabalho realizado

Nesta tese foram revistos vários tópicos da Computação Evolutiva, nomeadamente os Algoritmos Genéticos e os Algoritmos Meméticos (dentro dos Algoritmos Evolutivos) e os Algoritmos de Optimização por Enxames de Partículas (dentro da Inteligência dos Enxames). Estes algoritmos foram

aplicados à síntese de circuitos digitais, mais especificamente à síntese de circuitos lógicos combinatórios.

Foi proposto um algoritmo genético para a síntese, com portas lógicas, de circuitos combinatórios, no qual é possível especificar o conjunto de portas lógicas com as quais se pretende implementar os circuitos.

Elaborou-se um estudo sobre o tempo de processamento *versus* o tamanho da população no algoritmo genético implementado.

Foi introduzida a medida da descontinuidade do erro na avaliação dos circuitos através da função de aptidão. Foi também aplicado cálculo fraccionário à função de aptidão implementada nos algoritmos.

Foi também desenvolvido um outro algoritmo evolutivo, nomeadamente um algoritmo memético, para o projecto de circuitos digitais. Este algoritmo apresenta como principal vantagem a inclusão de um método de pesquisa local, o que faz com que o algoritmo atinja a solução em menor número de gerações.

O último algoritmo proposto pertence à classe dos algoritmos da inteligência dos enxames e foi desenvolvido com as duas vertentes de avaliação da função de aptidão: a estática e a dinâmica.

Foi desenvolvido um método de classificação de circuitos lógicos combinatórios, no que diz respeito ao grau de complexidade dos circuitos.

Estudou-se também o impacto da utilização da uma função de aptidão que incorpora duas fases em sequência. Este estudo foi realizado para os três algoritmos propostos nesta tese.

O crescimento exponencial das tabelas de verdade, à medida que o número de entradas e de saídas dos circuitos aumenta, origina o problema de escala na

síntese de circuitos digitais. Esta problemática foi igualmente abordada nos diversos capítulos desta tese e com todos os algoritmos implementados.

2. Síntese conclusiva

Os algoritmos propostos para a síntese de circuitos lógicos combinatórios apresentam como resultado final circuitos funcionais e otimizados em termos de complexidade, isto é, com o menor número possível de portas lógicas.

Para todos os casos estudados nesta tese, os algoritmos provaram ser eficientes, sendo visível que, para o algoritmo genético e para o algoritmo de optimização por enxames de partículas, se obtêm desempenhos superiores com a diminuição da complexidade dos conjuntos de portas lógicas.

Analisando a influência do tamanho da população no tempo de processamento para a obtenção da solução, conclui-se que o melhor tempo de processamento ocorre para tamanhos de populações relativamente pequenos.

Foram propostas duas técnicas com o objectivo de melhorar o desempenho dos algoritmos implementados, baseadas na forma de avaliação da função de aptidão. No que diz respeito à função de aptidão clássica, que se designou de estática, conclui-se que é possível obter resultados superiores medindo a descontinuidade do erro. Por outro lado, o novo conceito de função de aptidão, que se designou de dinâmica de ordem fraccionária, evidencia uma metodologia que ultrapassa a abordagem tradicional da função de aptidão estática.

A função de aptidão dinâmica foi implementada através de dois métodos de aplicação de cálculo fraccionário, o primeiro usando séries de Taylor e o segundo usando a aproximação de fracções de Padé. Verificou-se que os resultados obtidos com ambas as técnicas são similares; no entanto, com a implementação por aproximação de Padé obtêm-se tempos de processamento mais eficientes.

A integração de um algoritmo de pesquisa local no algoritmo genético permitiu o melhoramento das capacidades de pesquisa do algoritmo original o que permitiu obter resultados superiores em termos da média e do desvio padrão do número de gerações necessárias para alcançar as soluções. As experiências realizadas provaram que o algoritmo memético proposto é bastante eficiente na síntese de circuitos lógicos combinatórios em comparação com as abordagens clássicas dos algoritmos genéticos.

Na área da inteligência dos enxames, o algoritmo de optimização por enxames de partículas revelou ter um bom desempenho na síntese de circuitos lógicos combinatórios, com a vantagem adicional de ser um algoritmo extremamente eficiente no que diz respeito a tempos computacionais.

A aplicação de funções de aptidão sequenciais para a avaliação dos circuitos digitais obtidos pelos diversos algoritmos propostos permitiu constatar que:

- Quando a função de aptidão é dividida em objectivos sequenciais e dependentes é necessário maior esforço computacional para se garantir todas as fases dessa mesma função;
- A convergência torna-se mais difícil.

3. Desenvolvimentos futuros

Tendo em conta o trabalho desenvolvido e uma vez estabelecidas as respectivas conclusões, é chegada a altura de uma reflexão sobre as possíveis perspectivas de desenvolvimento futuro. Assim, as que se afiguram mais evidentes são:

- Dar continuidade à implementação de novos algoritmos da inteligência artificial em geral aplicada à síntese de circuitos lógicos combinatórios;
- Desenvolver novos conceitos de funções de aptidão, nomeadamente com aplicação do cálculo fraccionário;
- Implementar uma metodologia que permita a sintonização automática dos parâmetros da função de aptidão dinâmica envolvidos em cada caso;
- Explorar os algoritmos em circuitos lógicos combinatórios de maior complexidade.

Outros aspectos que podem ser alvo de uma possível investigação no futuro, são:

- O desenvolvimento de novas técnicas com o objectivo de minimizar o problema de escala dos circuitos combinatórios;
- A síntese de circuitos sequenciais com algoritmos da computação evolutiva.

Em suma, nesta área do conhecimento predominantemente científica e tecnológica, não é arriscado dizer que irão surgir novas perspectivas de evolução do presente trabalho. Assim, ele próprio não é mais do que uma geração do “algoritmo evolutivo” geral que levará a uma “solução” óptima após decorrido um número suficiente de “iterações”.

Apêndice A

SISTEMAS DIGITAIS

Introdução

A electrónica é um ramo da física onde se estudam os fenómenos das cargas eléctricas, as propriedades e o comportamento dos electrões, fotões, partículas elementares e ondas electromagnéticas. Na actualidade a electrónica é vista como uma especialidade tecnológica e divide-se em dois grandes campos, ou áreas, que adquiriram individualidade própria. A primeira área, mais clássica, é a Electrónica Analógica que trabalha com variáveis de sinais contínuos. Originalmente, o termo “analógico” descrevia a relação entre um sinal e uma tensão ou uma corrente que representavam o sinal. A segunda área é a Electrónica Digital, de carácter mais específico que a anterior, que adquiriu uma grande importância, devido aos enormes avanços que se produziram no campo da integração de múltiplos componentes num único circuito integrado e às extraordinárias características destes. A utilização de dispositivos digitais provocou alterações profundas no modo de conceber e de controlar sistemas electrónicos.

Um sistema digital é aquele que usa números discretos, em especial números binários, ou símbolos não-numéricos tais como letras ou ícones, para entradas, processamento, transmissão, armazenamento ou saídas, em vez de um espectro contínuo de valores (como nos sistemas analógicos).

A distinção entre “digital” e “analógico” pode referir-se ao método de entradas, armazenamento e transferência de dados, ao modo de funcionamento interno de um instrumento, ou ao tipo de dispositivo de saída. O termo surge da mesma fonte do termo dígito (*digitus*: do Latim, dedo - conta pelos dedos - porque estes são usados para contagem discreta).

O termo digital é mais vulgarmente usado na computação e na electrónica, especialmente onde a informação do mundo real é convertida para a forma binária numérica, como ocorre no áudio digital e na fotografia digital. Tais meios de transmissão de sinais transmitem um ou dois impulsos electrónicos ou ópticos, “1” lógico (presença de impulso) ou “0” (ausência de impulso).

1. Circuitos Digitais

Os circuitos digitais têm por base um número de níveis discretos da tensão ao contrário dos circuitos analógicos que usam tensões contínuas para representar as variáveis directamente. Os circuitos digitais constituem a representação mecânica mais comum da álgebra de Boole (Boole, 1958) e são parte integrante e fundamental de todos os computadores digitais. Nos nossos dias, os computadores digitais têm um papel na sociedade moderna de tal forma crescente e promissor que, normalmente, se intitula a nossa era como sendo a era da informação. Os computadores estão presentes nas transacções

comerciais, nas comunicações, transportes, medicina, entretenimento, etc. Monitoram o tempo e o ambiente. No mundo industrial, são amplamente utilizados no projecto, produção, distribuição e vendas. Contribuíram para muitas descobertas científicas e inúmeros desenvolvimentos das ciências da engenharia que não seriam possíveis de outra forma.

A característica mais surpreendente de um computador digital advém da generalidade (Mano, 2001). O computador pode seguir um conjunto de instruções, designado de programa, que opera os dados que são fornecidos. O utilizador pode especificar ou alterar o programa ou os dados de acordo com as necessidades. Como resultado desta flexibilidade, os computadores digitais de uso geral podem realizar uma variedade grande de tarefas de informação e processamento nas mais diversas áreas de aplicação. O computador de uso geral é o melhor exemplo que se conhece de um sistema digital. A característica de um sistema digital consiste na manipulação de elementos de informação discretos. Qualquer conjunto que está restrito a um número finito de elementos contém informação discreta. Exemplos de conjuntos discretos são os 10 dígitos decimais, as 26 letras do alfabeto e as 52 cartas de jogo. Os primeiros computadores digitais eram essencialmente usados para computação numérica. Neste caso, os elementos discretos eram os dígitos, daí ter surgido o termo computador digital. Na figura A.1 pode-se ver o primeiro computador digital.

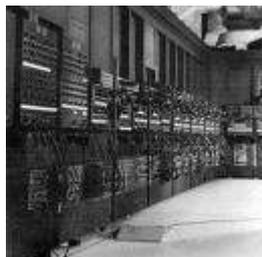


Figura A. 1- ENIAC - 1946. Primeiro computador digital electrónico de grande escala

Os elementos de informação discretos são representados, nos sistemas digitais, por grandezas físicas designadas por sinais.

1.1. Estrutura dos sistemas digitais

Os engenheiros utilizam vários métodos para minimizar funções lógicas, com o objectivo de reduzir a complexidade, reduzindo desta forma o número de erros e o custo dos circuitos digitais. Os métodos mais usados incluem as tabelas de verdade, os mapas de *Karnaugh* e desenvolvimento automático da álgebra booleana.

As representações são cruciais no desenho de circuitos digitais. A forma clássica de representação dos circuitos digitais é com o conjunto equivalente de portas lógicas. Outra forma, normalmente com menos electrónica, é a construção de um sistema equivalente com interruptores electrónicos (normalmente transístores). Uma das maneiras mais fáceis é ter simplesmente uma memória que contenha a tabela de verdade. O endereço da memória é alimentado com as entradas e os dados de saída da memória são as saídas do circuito.

Estas representações têm formatos digitais de ficheiros que podem ser processados por programas de computador para uma análise automática. Para se escolher as representações é necessário considerar os tipos de sistemas digitais. A maioria dos sistemas digitais divide-se em “sistemas combinatórios” e “sistemas sequenciais”. Um sistema combinatório apresenta sempre a mesma saída para entradas idênticas. Basicamente um sistema combinatório é a representação de um conjunto de funções lógicas.

Um sistema sequencial é um sistema combinatório no qual algumas das saídas são realimentadas novamente como entradas. Isto faz com que a máquina digital realize uma “sequência” de operações. O sistema sequencial mais simples que existe é, provavelmente, o *flip-flop*, um mecanismo que representa um dígito binário ou *bit*. Os sistemas sequenciais são muitas vezes designados por máquinas de estado. Desta maneira é possível projectar o comportamento geral de um sistema, e até mesmo testá-lo, em simulação, sem ser necessário considerar todos os detalhes das funções lógicas.

Os sistemas sequenciais dividem-se ainda em duas sub-categorias. Os sistemas sequenciais síncronos, que mudam de estado com um sinal de relógio, e os sistemas sequenciais assíncronos, que mudam de estado quando as entradas mudam. Os sistemas sequenciais síncronos são constituídos por circuitos assíncronos, tais como os *flip-flops*, que transitam de estado apenas quando muda o sinal de relógio e que têm margens de tempo cuidadosamente desenhadas.

A forma usual de implementar uma máquina de estados sequencial síncrona é dividi-la em lógica combinatória e num conjunto de *flip-flops* designado por registo de estados. A cada sinal de relógio, o registo de estados captura a realimentação gerada pela lógica combinatória anterior e volta a alimentar, a parte combinatória da máquina de estados. A velocidade do relógio é estabelecida com base nos cálculos do dispositivo lógico que consome mais tempo da lógica combinatória.

O registo de estados é apenas a representação de um número binário. Se os estados na máquina de estados são numerados, a função lógica é apenas lógica simples que produz o número do próximo estado.

Em comparação, os sistemas assíncronos são bastante difíceis de projectar devido a ter que se considerar todos os estados possíveis em todos os tempos. O

método normalmente usado passa pela construção de uma tabela com informação sobre os tempos mínimo e máximo que cada estado pode durar. De seguida, ajusta-se o circuito para minimizar o número desses estados e forçar o circuito a, periodicamente, esperar que todas as suas partes entrem num estado compatível (auto re-sincronização). Sem haver este tipo de atenção no projecto destes sistemas é muito fácil produzir acidentalmente lógica assíncrona instável, isto é, o sistema que apresenta resultados imprevisíveis devido aos repetidos atrasos causados por pequenas variações de valores dos componentes electrónicos.

Hoje em dia a maioria das máquinas digitais são síncronas devido à maior facilidade de concepção e verificação destas máquinas em relação às assíncronas. No entanto, a lógica assíncrona, quando bem projectada, é considerada superior uma vez que a velocidade não está condicionada a impulsos de relógio. Pelo contrário, a lógica assíncrona é executada à máxima velocidade que as taxas de propagação das portas lógicas permitem, dependendo apenas da sua construção. A rapidez de um circuito assíncrono depende apenas da velocidade dos circuitos lógicos que o constituem. Generalizando, pode-se dizer que grande parte dos sistemas digitais são máquinas de fluxo usualmente projectadas usando lógica de transferência por registos síncrona através de linguagens de programação especializadas tais como o VHDL¹ ou o Verilog².

Na lógica de transferência por registos, os números binários (isto é, os dados) são armazenados em grupos de *flip-flops* designados por registos de deslocamento. As saídas de cada registo formam um pacote de linhas designadas por barramento (*bus*) que transferem esses dados para os módulos

¹ "VHSIC *Hardware Description Language*" (Linguagem de descrição de *hardware*) é uma linguagem usada para facilitar o projecto/concepção de circuitos digitais em FPGAs e ASICs. Um VHSIC é um tipo de circuito digital. A sigla deriva do nome em inglês, "Very-High-Speed Integrated Circuit" (circuito integrado de velocidade muito alta).

² Verilog é uma linguagem de descrição de *hardware*.

seguintes de cálculo. Estes módulos são constituídos por lógica combinatória que têm, da mesma forma, barramentos de saída que podem estar ligados às entradas de vários registos. Por vezes os registos têm um multiplexador na entrada para poderem ir armazenando os dados provenientes de qualquer um dos vários barramentos existentes no circuito. Em alternativa, as saídas dos vários dispositivos podem estar ligadas a um barramento através de *buffers*³ que têm a capacidade de desligar as saídas e todos os dispositivos à excepção de um. Uma máquina sequencial de estados controla cada registo de deslocamento, ou seja, o instante quando cada registo aceita novos dados para as suas entradas.

Nos anos 80 alguns investigadores descobriram que quase todas as máquinas de fluxo síncronas podiam ser convertidas em desenhos assíncronos usando lógica de sincronização do tipo *first-in-first-out*. Neste esquema, a máquina digital é caracterizada por um conjunto de fluxos de dados. Em cada etapa da transferência, o circuito assíncrono de sincronização determina quando é que as saídas dessa etapa são válidas e gera um sinal que comunica aos estágios seguintes para as considerarem e armazenarem. Verificou-se que eram apenas necessários alguns destes circuitos de sincronização.

O exemplo mais comum de uma máquina de lógica por registo de deslocamento de uso geral é o computador. Basicamente é uma calculadora binária automática. A unidade de controlo de um computador é usualmente designada por micro-programa executado por um micro-sequenciador. Cada entrada de tabela ou palavra (*word*) do micro-programa comanda o estado de cada *bit* que controla o computador. O sequenciador conta e por sua vez o contador endereça a memória ou a máquina de lógica combinatória que contém o micro-programa. Os *bits* do micro-programa controlam a unidade lógica e

³ Circuito isolador, frequentemente um amplificador, usado para minimizar a influência de um circuito noutra.

aritmética incluindo o próprio micro-sequenciador. Nesta perspectiva, a tarefa complexa de concepção do controlo de um computador fica reduzida à tarefa, bem mais simples, de programar uma colecção relativamente independente de máquinas lógicas.

A arquitectura de computadores é uma actividade especializada de engenharia dedicada ao estudo da estrutura e organização das várias partes que constituem um computador, nomeadamente os registos, lógica de cálculo, barramentos e outras partes do computador.

2. Álgebra de Boole

Em 1854, o matemático britânico George Boole (1815 - 1864), através da obra intitulada *An Investigation of the Laws of Thought*, apresentou um sistema matemático de análise lógica conhecido como álgebra de Boole. A figura A.2 ilustra a capa deste livro.

No início da era da electrónica, todos os problemas eram resolvidos por sistemas analógicos. Apenas em 1938, o engenheiro americano Claude Elwood Shannon utilizou as teorias da álgebra de Boole para a solução de problemas de circuitos de telefonia com relés, tendo publicado um trabalho denominado *Symbolic Analysis of Relay and Switching*, praticamente introduzindo na área tecnológica o campo da electrónica digital. Este ramo da electrónica emprega nos seus sistemas um pequeno grupo de circuitos básicos padronizados e conhecidos como portas lógicas.

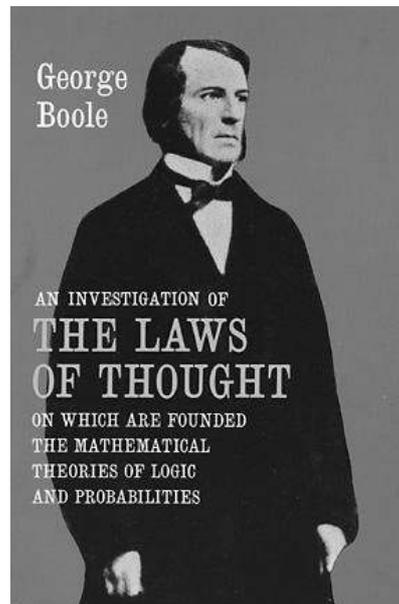


Figura A. 2 - Capa do livro "An Investigation of the Laws of Thought " de George Boole

A álgebra de Boole trata de um tipo de álgebra que, baseando-se na teoria dos conjuntos, aplica-se a sistemas matemáticos que só consideram dois elementos possíveis (Hassoun e Sasao, 2002): 0 e 1. Desta definição decorre a possibilidade de ser aplicada à análise e desenho de circuitos digitais usando a seguinte convenção:

- Presença de tensão = 1;
- Ausência de tensão = 0.

Em álgebra de Boole as variáveis que aparecem nas equações são usualmente representadas através de letras maiúsculas ou minúsculas, preferindo-se as primeiras letras do alfabeto (Cuesta *et al*, 1994).

3. Portas Lógicas

Portas lógicas são dispositivos, ou circuitos lógicos (figura A.3), que operam um ou mais sinais lógicos de entrada para produzir uma, e somente uma, saída, dependente da função implementada no circuito. As portas lógicas são geralmente usadas em circuitos electrónicos que empregam sinais do tipo: presença de sinal, ou "1"; e ausência de sinal, ou "0". As situações "Verdadeira" e "Falsa" são estudadas na Lógica Matemática ou na Lógica de Boole que dá origem ao nome destas portas. O comportamento das portas lógicas é conhecido pela tabela verdade que apresenta os estados lógicos das entradas e das saídas (Taub, 1982).

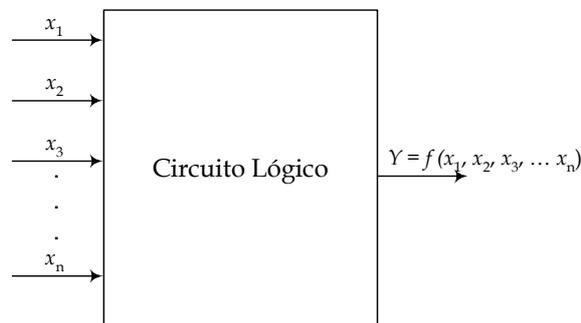


Figura A. 3 - Representação genérica de um circuito lógico

3.1. Portas lógicas AND, OR, XOR e NOT

A porta lógica AND combina dois ou mais sinais de entrada de forma equivalente a um circuito em série, para produzir um único sinal de saída. A porta AND produz uma saída 1, se todos os sinais de entrada forem 1. Se um

dos sinais de entrada for em 0, então a porta AND produz um sinal de saída igual a 0.

A porta lógica OR combina dois ou mais sinais de entrada de forma equivalente a um circuito em paralelo, para produzir um único sinal de saída. Ela produz uma saída 1, se qualquer um dos sinais de entrada for igual a 1. Por outro lado, a porta OR produzirá um sinal de saída igual a 0 apenas se todos os sinais de entrada forem 0.

A porta lógica XOR compara os *bits*: ela produz saída 0 quando todos os *bits* de entrada são iguais e saída 1 quando pelo menos um dos *bits* de entrada é diferente dos demais.

A porta lógica NOT inverte o sinal de entrada (executa a NEGAÇÃO do sinal de entrada), ou seja, se o sinal de entrada for 0 ela produz uma saída 1 e se a entrada for 1 ela produz uma saída 0.

A figura A.4 ilustra as tabelas de verdade (ou de funcionamento) das quatro portas lógicas descritas.

<i>A</i>	<i>B</i>	<i>Y</i>	<i>A</i>	<i>B</i>	<i>Y</i>	<i>A</i>	<i>B</i>	<i>Y</i>	<i>A</i>	<i>Y</i>
0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	1	1	1	0
1	0	0	1	0	1	1	0	1		
1	1	1	1	1	1	1	1	0		
	<i>a)</i>			<i>b)</i>			<i>c)</i>			<i>d)</i>

Figura A. 4 - Tabelas de verdade: a) AND b) OR c) XOR d) NOT

A porta lógica NAND equivale a uma porta AND seguida por uma porta NOT, isto é, ela produz uma saída que é a negação da saída produzida pela porta AND. A porta NOR equivale a uma porta OR seguida por uma porta NOT, isto é, ela produz uma saída que é a negação da saída produzida pela

porta OR. A porta lógica XNOR equivale a uma porta XOR seguida por uma porta NOT, isto é, ela produz uma saída que é a negação da saída produzida pela porta XOR.

As tabelas de verdade destas portas lógicas são apresentadas na figura A.5 e na tabela A.1 podem ser visualizados os símbolos gráficos das portas lógicas descritas assim como as respectivas equações booleanas.

<i>A</i>	<i>B</i>		<i>Y</i>
0	0		1
0	1		1
1	0		1
1	1		0

a)

<i>A</i>	<i>B</i>		<i>Y</i>
0	0		1
0	1		0
1	0		0
1	1		0

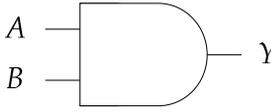
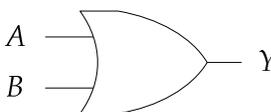
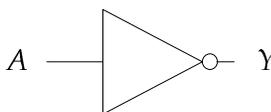
b)

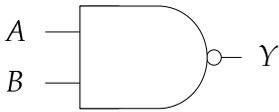
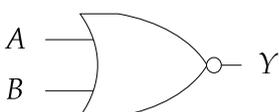
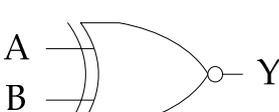
<i>A</i>	<i>B</i>		<i>Y</i>
0	0		1
0	1		0
1	0		0
1	1		1

c)

Figura A. 5 - Tabelas de verdade: a) NAND b) NOR c) XNOR

Tabela A. 1- Portas Lógicas

Função Lógica Básica	Símbolo Gráfico	Equação Booleana
AND		$Y = A \cdot B$
OR		$Y = A + B$
XOR		$Y = A \oplus B$
NOT		$Y = \bar{A}$

NAND		$Y = \overline{A \cdot B}$
NOR		$Y = \overline{A + B}$
XNOR		$Y = \overline{A + B}$

4. Circuitos Combinatórios

Os circuitos digitais combinacionais, também designados por circuitos combinatórios, são circuitos que satisfazem a condição de as suas saídas serem exclusivamente função das suas entradas, sem intervenção do último valor em que se encontram essas saídas. Por outras palavras, os circuitos combinacionais, não memorizam o estado anterior das suas saídas.

Um circuito combinacional é constituído por um conjunto de portas lógicas que determinam os valores das saídas directamente a partir dos valores actuais das entradas. Pode-se dizer que um circuito combinacional realiza uma operação de processamento de informação, a qual pode ser especificada através de um conjunto de equações Booleanas (Sasao, 1999). Cada combinação de valores de entrada pode ser vista como uma informação diferente e cada conjunto de valores de saída representa o resultado da operação. Estes circuitos são criados implementando a respectiva equação Booleana de funcionamento com portas lógicas e cumprem múltiplas funções dentro dos circuitos digitais.

A figura A.6 mostra o diagrama de blocos genérico de um circuito combinacional.

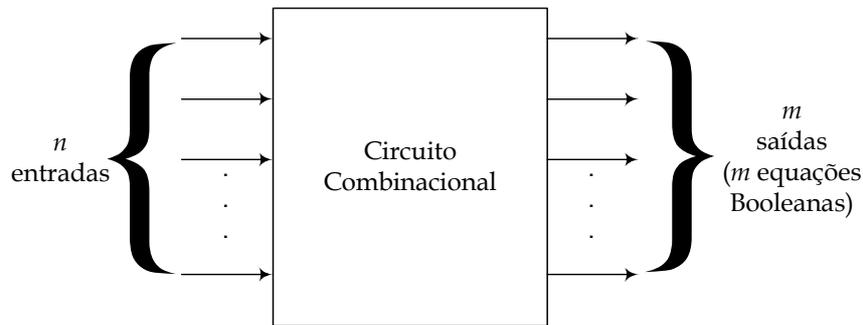


Figura A. 6 - Diagrama genérico de um circuito combinacional.

Os circuitos combinacionais classificam-se segundo a função que desempenham dentro dos sistemas digitais, nos grupos seguintes:

Circuitos de Comunicação:

Estes circuitos são utilizados para transmitir informação através de uma linha de comunicação, para codificar, decodificar ou alterar a estrutura da informação a ser transmitida. Neste grupo destacam-se os codificadores (*encoder*), decodificadores, multiplexadores (*multiplexer*) e demultiplexadores (*demultiplexer*).

Circuitos Aritméticos:

Estes circuitos são responsáveis pela realização das operações aritméticas com os dados binários que processam. Exemplos destes circuitos são os somadores (*adders*), os substractores e os comparadores.

4.1. Circuitos de Comunicação

Esta secção descreve os circuitos combinatórios de comunicação.

4.1.1. Codificadores

Estes circuitos possuem n saídas e um número de entradas menor ou igual a 2^n (figura A.7) cuja estrutura é tal que, quando é activada uma das entradas (por selecção de um nível lógico determinado, 0 ou 1), na saída surge a representação binária (ou o seu complemento) do número decimal atribuído a essa entrada.

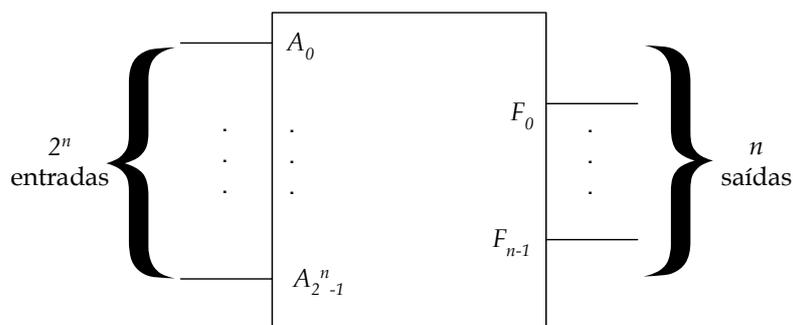


Figura A. 7 - Codificador binário 2^n -para- n .

A aplicação mais usual dos codificadores consiste em converter qualquer informação digitalizada das entradas dos sistemas digitais no respectivo, ou equivalente, valor binário natural, ou em qualquer outro código binário.

A figura A.8 apresenta a tabela de verdade do circuito codificador binário 4-para-2 e a figura A.9 ilustra o respectivo esquema eléctrico.

A_3	A_2	A_1	A_0	F_1	F_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Figura A. 8 - Tabela de verdade do codificador binário 4-para-2.

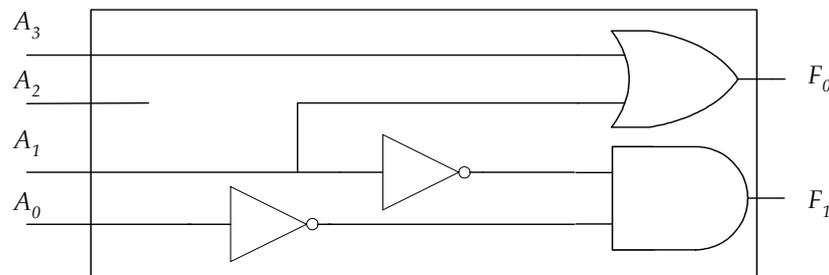


Figura A. 9 - Esquemático do circuito codificador binário 4-para-2.

4.1.2. Descodificadores

São circuitos combinacionais que possuem n entradas e um número de saídas menor ou igual a 2^n (figura A.10). Funcionam de modo a que, quando aparece uma combinação binária nas suas entradas, apenas é activada uma das saídas. Normalmente a saída activada fica no nível lógico baixo (0), enquanto as restantes permanecem no nível lógico alto (1) (Brown, 2003).

No entanto, nem todos os descodificadores possuem a mesma atribuição de estados lógicos, e existem muitos que são activos ao nível lógico alto (1).

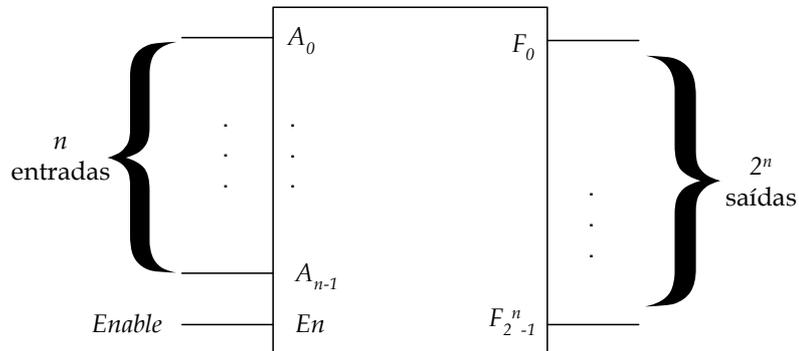


Figura A. 10 - Descodificador binário n -para- 2^n .

Os descodificadores são utilizados em sistemas digitais na conversão de informação binária noutra tipo de informação digitalizada (mas não binária) utilizada em dispositivos como, por exemplo, os visualizadores alfanuméricos. A figura A.11 apresenta a tabela de verdade do circuito descodificador binário 2-para-4.

En	A_1	A_0	F_0	F_1	F_2	F_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	0	0	0	0	0	1
0	x	x	0	0	0	0

Figura A. 11 - Tabela de verdade do descodificador binário 2-para-4.

4.1.3. Multiplexadores

Os multiplexadores são circuitos combinacionais que possuem N entradas de informação, n entradas de selecção ou controlo, uma entrada de autorização (*enable*) e uma saída de informação (figura A.12).

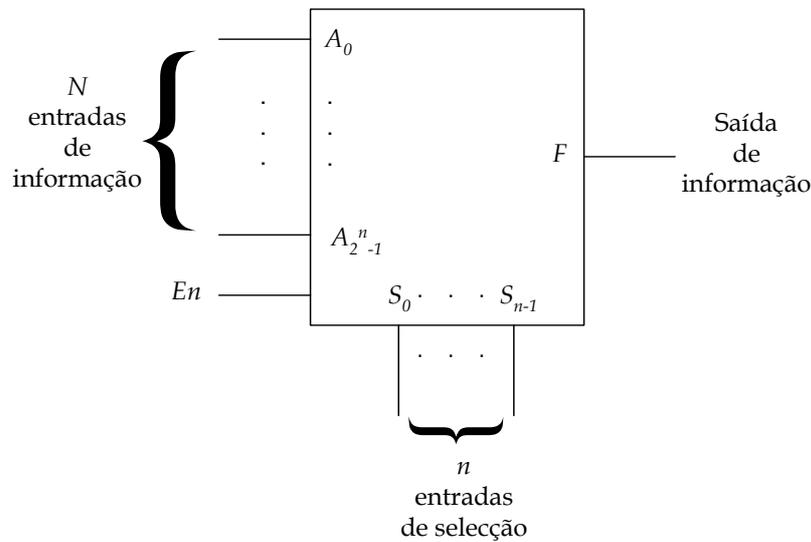


Figura A. 12 - Circuito multiplexador

As N entradas de informação estão relacionadas com as n linhas de selecção pela seguinte equação:

$$N = 2^n \quad (\text{A.1})$$

São circuitos que seleccionam uma entre 2^n entradas A_0, \dots, A_{2^n-1} ligando-a a uma saída comum. Os multiplexadores possuem n entradas de selecção ou controlo, de S_0, \dots a S_n , que são interpretadas como a representação binária do inteiro i , para seleccionar a entrada A_i . Pode-se dizer que são dispositivos que codificam as informações de duas ou mais fontes de dados num único canal. Eles são utilizados em situações onde o custo de implementação de canais separados para cada fonte de dados é maior que o custo e a inconveniência de utilizar as funções de multiplexação/demultiplexação.

Os circuitos multiplexadores são muito usados nos computadores dado que permitem que diferentes unidades utilizem o mesmo dispositivo. Por exemplo, um mesmo controlador pode controlar vários discos. As figuras A.13 e A.14

ilustram, respectivamente a tabela de verdade e o esquemático do circuito multiplexador 4-para-1.

En	S_1	S_0	F
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	0	0	A_3
1	x	x	0

Figura A. 13 - Tabela de verdade do multiplexador 4-para-1.

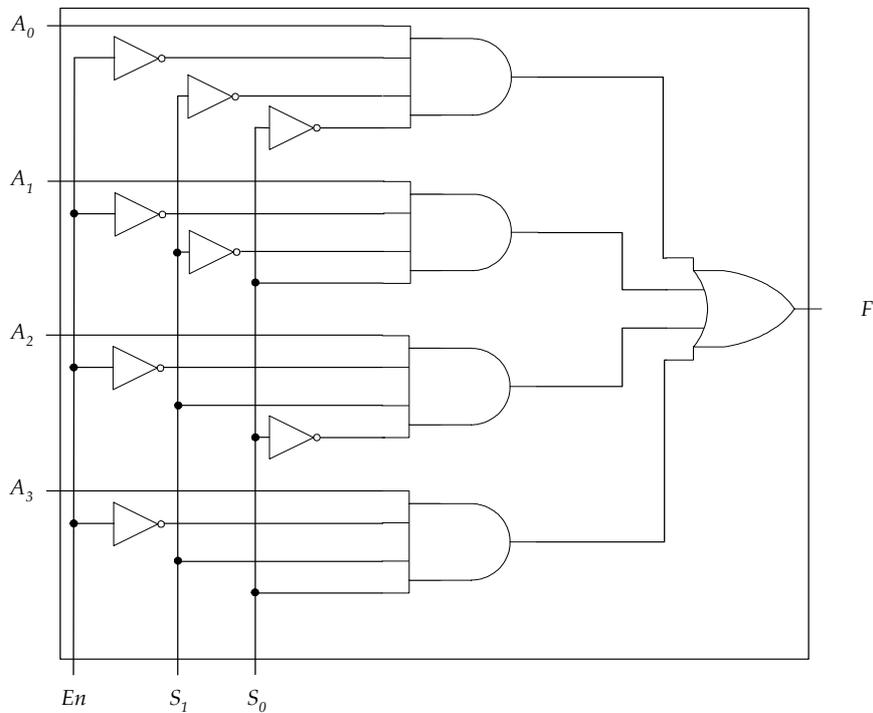


Figura A. 14 - Esquemático do circuito multiplexador 4-para-1.

4.1.4. Demultiplexador

O demultiplexador executa a operação inversa do multiplexador, isto é, toma

uma única entrada que distribui para uma das diversas saídas. São circuitos combinacionais que possuem uma entrada de informação, n entradas de selecção ou controlo, uma entrada de autorização (*enable*) e N saídas de informação (figura A.16). A figura A.16 apresenta a tabela de verdade do circuito demultiplexador 1-para-4.

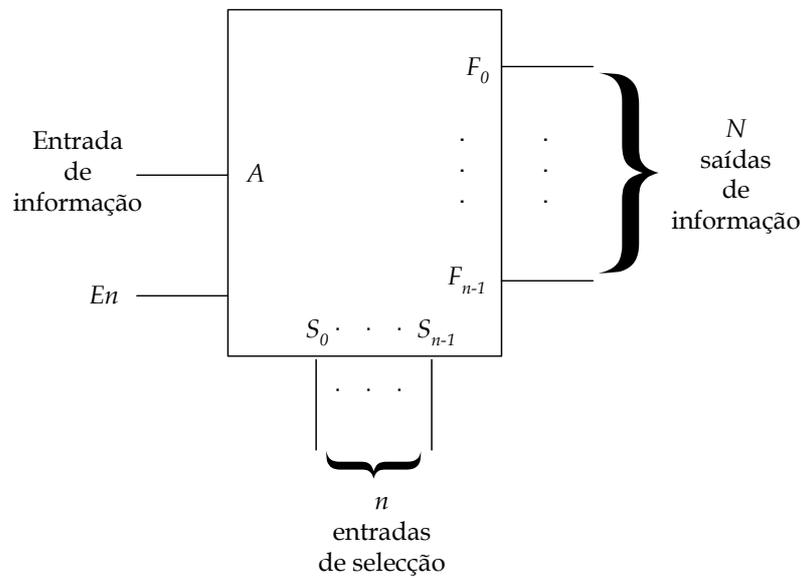


Figura A. 15 - Circuito demultiplexador.

En	S_1	S_0	F_0	F_1	F_2	F_3
0	0	0	A	0	0	0
0	0	1	0	A	0	0
0	1	0	0	0	A	0
0	0	0	0	0	0	A
1	x	x	0	0	0	0

Figura A. 16 - Tabela de verdade do demultiplexador 1-para-4.

4.2. Circuitos aritméticos

Esta secção descreve os circuitos combinatórios aritméticos.

4.2.1. Semi-somadores

O semi-somador é um circuito digital que faz a soma de dois dígitos binários presentes nas entradas, fornecendo nas saídas o resultado da soma e o respectivo transporte ou *carry out* (figura A.17).

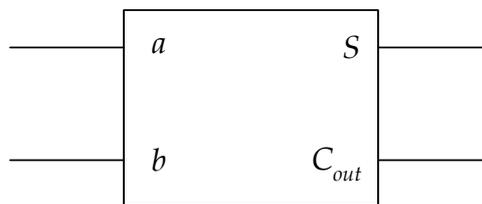


Figura A. 17 - Circuito semi-somador.

A tabela de verdade que representa o funcionamento deste circuito é:

<i>a</i>	<i>b</i>	<i>S</i>	<i>C_{out}</i>
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figura A. 18 - Tabela de verdade do semi-somador.

A partir da tabela de verdade podem deduzir-se as equações lógicas do circuito:

$$S = \bar{a} b + a \bar{b}, C_{out} = a b$$

A figura A.19 apresenta a implementação do respectivo circuito com portas lógicas.

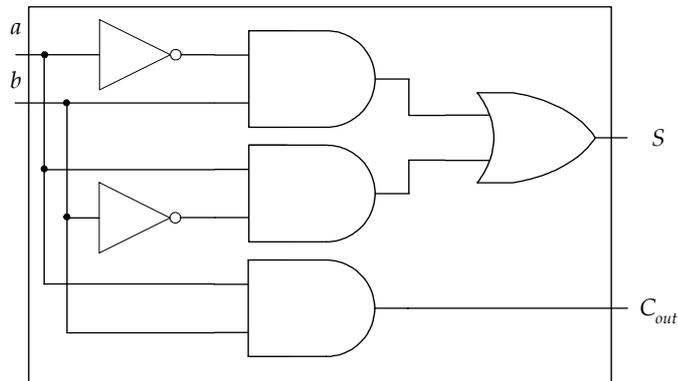


Figura A. 19 - Esquemático do circuito semi-somador.

4.2.2. Somadores

Um circuito somador é um circuito aritmético que efectua a soma binária dos dois dígitos de entrada com o transporte de entrada procedente do andar anterior. O circuito é constituído pelas mesmas saídas S e C que o semi-somador e uma entrada mais que este. Este circuito é ilustrado na figura A.20.

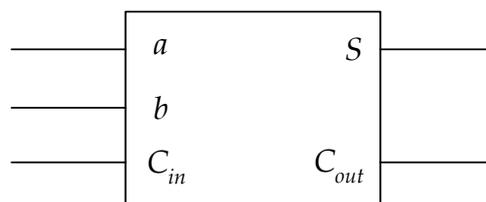


Figura A. 20 - Circuito somador.

A tabela de verdade correspondente a este circuito encontra-se representada na figura A.21.

C_{in}	a	a	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figura A. 21 - Tabela de verdade do somador completo.

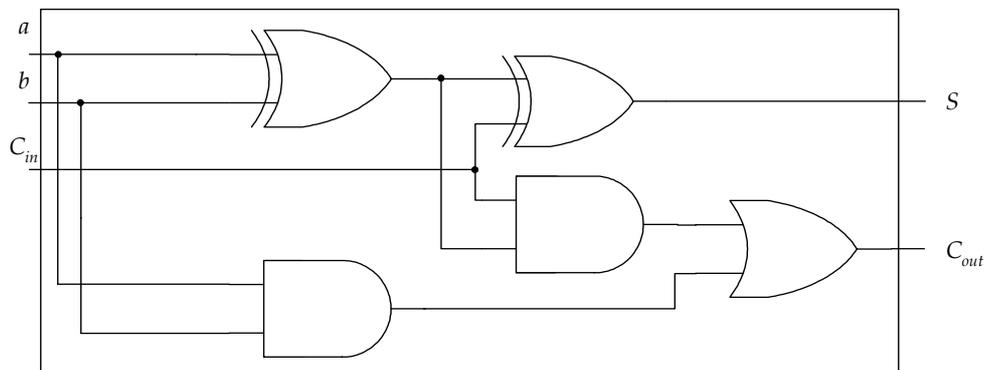


Figura A. 22 - Esquemático do circuito somador completo.

4.2.3. Subtratores

A estrutura dos circuitos subtratores é muito similar à dos somadores, com a diferença de calcularem a subtração binária dos dígitos de entrada, e de os transportes, de saída e de entrada, receberem o nome de empréstimo (*borrow*) (figura A.23).

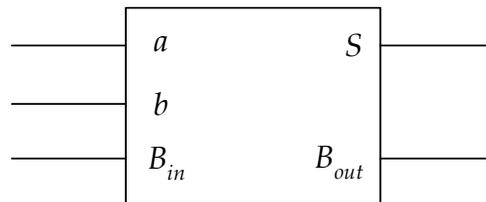


Figura A. 23 - Circuito subtrator.

B_{in}	a	a	S	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Figura A. 24 - Tabela de verdade do subtrator.

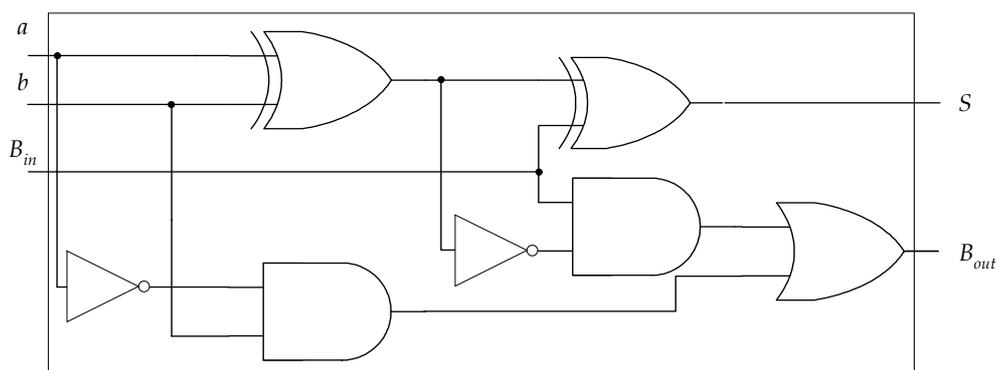


Figura A. 25 - Esquemático do circuito subtrator.

4.2.4. Comparadores

Os circuitos comparadores são circuitos combinacionais que indicam se dois números A e B de n bits são iguais ou se são diferentes e, neste caso, qual a relação de desigualdade entre eles. Além disso, podem dispor de um conjunto de entradas de acoplamento em cascata para que possam comparar palavras com um número de bits maior que as permitidas por um único dos comparadores utilizados.

A figura A.26 mostra o diagrama esquemático de um comparador de quatro bits.

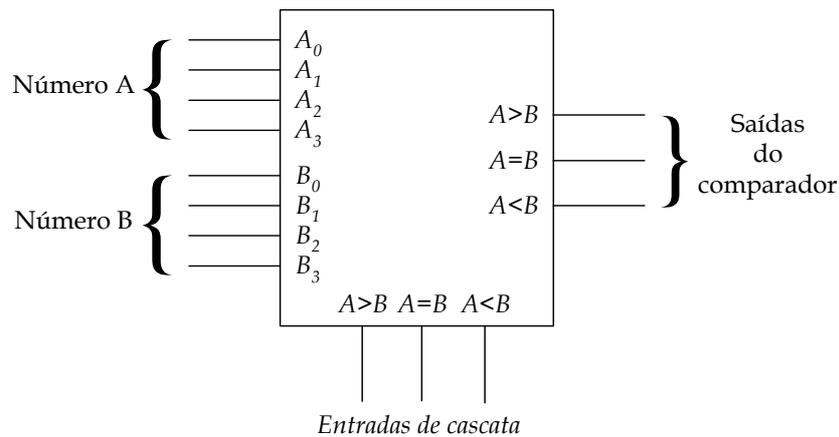


Figura A. 26 - Esquema de um comparador de quatro bits.

Referências

- Boole, G. An Investigation of the Laws of Thought. Dover; 1st American edition, 1958.
- Brown, S. e Vranesic, Z. Fundamentals of Digital Logic with Verilog Design. McGraw-Hill, New York, ISBN 0-07-282315-1, 2003.
- Cuesta, L., Padilla, A. G. e Remiro, F. Electrónica Digital. McGraw-Hill, Lisboa, ISBN 972-9241-64-3, 1994.
- Hassoun, S. e Sasao, T. Logic Synthesis and Verification. Kluwer Academic Publishers, USA, ISBN 0-7923-7606-4, 2002.
- Mano, M. M. e Kime, C. R. Logic and Computer Design Fundamentals. Prentice Hall, New Jersey, ISBN 0-13-031486-2, 2001.
- Sasao, T. Switching Theory for Logic Synthesis. Kluwer Academic Publishers, Massachusetts, ISBN 0-7923-84-56-3, 1999.
- Taub, H. e Schilling, D. Eletrônica Digital. McGraw Hill, São Paulo, 1982.