# Universidade de Évora


Mestrado em Engenharia Informática

# OPTIMIZAÇÃO EVOLUTIVA DE EQUIPAMENTO DE PROCESSAMENTO DE POLÍMEROS

**MARÍLIA ALEXANDRA FERNANDES FELISMINO**
**(LICENCIADA)**

Dissertação realizada sob a orientação do Prof. Doutor Carlos Manuel Mira da Fonseca e do Prof. Doutor Paulo Miguel Torres Duarte Quaresma, para a obtenção do grau de Mestre em Engenharia Informática.

Évora, Setembro de 2006

*Esta dissertação não inclui as críticas e sugestões feitas pelo júri.*

# Universidade de Évora

Mestrado em Engenharia Informática

# OPTIMIZAÇÃO EVOLUTIVA DE EQUIPAMENTO DE PROCESSAMENTO DE POLÍMEROS



**MARÍLIA ALEXANDRA FERNANDES FELISMINO**
**(LICENCIADA)**

Dissertação realizada sob a orientação do Prof. Doutor Carlos Manuel Mira da Fonseca e do Prof. Doutor Paulo Miguel Torres Duarte Quaresma, para a obtenção do grau de Mestre em Engenharia Informática.

Évora, Setembro de 2006

*Esta dissertação não inclui as críticas e sugestões feitas pelo júri.*

# AGRADECIMENTOS

Esta tese foi para mim um grande desafio pois como venho de uma área diferente, de uma licenciatura em Matemática Aplicada, quer a parte curricular quer a dissertação teve para mim um grau de dificuldade maior do que para qualquer um dos meus colegas do mestrado.

Todo este trabalho não teria sido possível de realizar sem as aulas extra de informática leccionadas pelo meu namorado. Muito Obrigada!

No Algarve encontrei umas colegas extraordinárias às quais agradeço todo o apoio que me deram, quer ao nível da adaptação à cidade e à universidade, quer a nível dos conteúdos por mim estudados. Daqui vai o meu muito obrigada à Marisol Correia e à Cristina Vieira.

Quanto aos orientadores, agradeço a todos eles em especial ao professor Gaspar Cunha (coordenador do projecto na Universidade do Minho) que foi incansável no apoio prestado.

Por último, agradeço à minha família por todo o incentivo à continuação dos meus estudos.

# RESUMO

Nesta dissertação descreve-se a aplicação de algoritmos evolutivos ao processo de extrusão de polímeros, mais especificamente ao design de parafusos convencional e barreira. Optimizam-se parâmetros como a geometria e as condições operatórias de parafusos, com objectivos como a melhoria do consumo de energia e do grau da mistura.

Após um estudo ao espaço de pesquisa implícito no simulador de extrusão utilizado, comparou-se o desempenho de dois operadores de recombinação em domínios reais, tendo sido escolhido este último, sobre o qual se efectuaram igualmente estudos teóricos.

Para tornar a execução do algoritmo mais rápida desenvolveu-se uma versão que realiza a avaliação das soluções em vários computadores em paralelo.

Testou-se um grande número de hipóteses para a codificação do problema e escolheu-se aquela que era estatisticamente superior e que permitia a maior redução possível no espaço de pesquisa (por partilha, através de uma estrutura hierárquica, de variáveis comuns a ambos os tipos de parafusos).

# EVOLUTIONARY OPTIMIZATION OF POLYMER PROCESSING EQUIPMENT

## ABSTRACT

In this study, the application of evolutionary algorithms to the polymer extrusion process is described, more specifically to the design of conventional and barrier screws. The problem parameters like the geometry and the operating conditions are optimized, with objectives like the improvement of mechanical power consumption and mixing degree.

After studying the search space implicit in the extrusion simulator used, the performance of the SBX and PCX recombination operators for real parameters was compared, having the latter been chosen, and theoretical studies conducted on it.

In order to speed up the algorithm's run time, a new version that conducts the evaluation of solutions in parallel was developed.

A significant number of alternatives were tested for the problem's encoding and the one which was statistically better and allowed for the greatest reduction in the search space (by sharing, by a hierarchical structure, variables common to both screw types) was chosen.

# ÍNDICE GERAL


**1. INTRODUÇÃO** ..... **1**

1.1 Motivação ..... 1

1.2 Estrutura da Tese ..... 2

1.3 Objectivos e Principais Contribuições ..... 3

**2. O processo de extrusão de polímeros** ..... **5**

2.1 Introdução ..... 5

2.2 A extrusão de polímeros como um problema de optimização ..... 7

2.2.1. Abordagens anteriores ..... 9

2.3 Conclusão ..... 11

**3. Optimização Evolutiva** ..... **13**

3.1 Introdução ..... 13

3.2 Algoritmos Genéticos ..... 13

3.2.1. Algoritmo Genético Básico ..... 14

3.2.2. Componentes do Algoritmo Genético ..... 15

3.2.3. Operadores para Domínios Contínuos ..... 20

3.3 Algoritmos Genéticos Estruturados (*sGA*) ..... 26

3.3.1. Perspectiva Biológica ..... 27

3.3.2. Níveis *sGA* ..... 29

3.3.3. Cromossoma ..... 29

3.3.4. Diversidade genotípica ..... 30

3.4 Conclusão ..... 30

**4. Avaliação do Desempenho de Optimizadores** ..... **32**

4.1 Introdução ..... 32

4.2 Algoritmos Genéticos Multiobjectivo ..... 32

4.2.1. Problema Multiobjectivo ..... 33

4.3 Indicador de Desempenho e Índice de Hipervolumes ..... 35

Hipervolumes ..... 38

4.4 Conclusão ..... 39

**5. Optimização EVOLUTIVA De Extrusoras ..... 40**
5.1 Introdução ..... 40
5.2 Formulação do Problema ..... 40
5.3 Estudo das Soluções Obtidas pelo Simulador de Extrusão ..... 45
5.4 Estudos Efectuados para a Escolha dos Operadores do Algoritmo Genético ..... 48
5.4.1. Comparação dos Operadores SBX e PCX ..... 48
5.4.2. Estudo Teórico do Operador *Parent Centric Crossover* (*PCX*) ..... 53
5.5 Conclusão ..... 61
**6. Resultados Experimentais ..... 62**
6.1 Introdução ..... 62
6.2 Algoritmo Proposto *G4PCX* ..... 62
6.3 Estudo Efectuado à Paralelização do Algoritmo ..... 68
6.4 Estudo das Variáveis Comuns à geometria dos dois tipos de parafusos 73
6.4.1. Resultados Obtidos Usando uma Soma Pesada ..... 75
6.4.2. Resultados Obtidos usando Duas Somas Pesadas ..... 79
6.4.3. Interpretação dos Resultados Obtidos ..... 81
**7. Conclusões ..... 83**

# ÍNDICE DE FIGURAS

Figura 1 – Extrusora com um único parafuso convenvional e zonas funcionais correspondentes [3] ..... 6

Figura 2 – Tipos de parafusos convencional e barreira: CS – Conventional screw, MS – Maillefer Screw, BS – Barr Screw, DLS – Dray and Lawrence Screw and KS – Kim Screw. [3] ..... 7

Figura 3 - Exemplo de uma geometria do parafuso convencional [3] ..... 8

Figura 4 - Geometria básica da secção barreira [1] ..... 9

Figura 5 - Representação Binária ..... 16

Figura 6 - Representação Real ..... 16

Figura 7 - *Single Point Crossover* ..... 18

Figura 8 - *Multiple Point Crossover* ..... 19

Figura 9 - *One Bit Mutation* ..... 19

Figura 10 - Distribuição de probabilidade de soluções filhas calculadas pelo *SBX* [6]. ..... 22

Figura 11 - Várias distribuições consoante o índice n [7] ..... 24

Figura 12 - Distribuição de soluções filhas calculadas com três pais, numa abordagem *Parent-Centric* (Algoritmo *PCX*) [1] ..... 26

Figura 13 – DNA ..... 27

Figura 14 - Estrutura hierárquica do cromossoma ..... 28

Figura 15 - Codificação Estruturada ..... 28

Figura 16 - Exemplo de um cromossoma e de duas possíveis árvores associadas ..... 29

Figura 17 - Um exemplo de um problema com duas funções objectivos, ambas de minimização. A frente de Pareto está representada com a linha carregada. Nesta figura está também representado o índice de hipervolume do ponto X. 35

Figura 18 – Exemplo da aplicação do Testes Estatístico Kolmogorov-Smirnov para Comparação de duas funções comulativas [17] ..... 37

Figura 19 - Variáveis do parafuso convencional ..... 41

Figura 20 - Variáveis do parafuso barreira Maillefer ..... 41

Figura 21 - Cromossoma considerando todas as variáveis "comuns" aos dois parafusos, separadas. ..... 44

Figura 22- Árvore associada ao cromossoma, tendo todas as variáveis separadas ..... 44

Figura 23 - Distribuição dos resultados obtidos em 2000 execuções do programa de simulação do processo de extrusão ..... 46

Figura 24 - Evolução média do algoritmo 1 ..... 49

Figura 25 - Evolução média do algoritmo 2 ..... 49

Figura 26 – Funções de Aproveitamento Empíricas aplicadas a 10 execuções do PCX e a 10 execuções do SBX ..... 51

Figura 27 – Frequências Comulativas de pontos gerados pelo PCX (com os pais [3,3], [-2, 10], [-5, 0]) e por um gerador de números aleatórios segundo distribuição normal ..... 53

Figura 28- Frequências Cumulativas de pontos gerados pelo PCX (com os pais [0,2, 3]) num caso com o algoritmo standard e no outro através da fórmula para o cálculo explícito do desvio padrão ..... 60

Figura 29 - Comparação das Optimizações Com e Sem Paralelização ..... 70

Figura 30 - Uma possivel codificação do problema agregando, neste caso, todas as variáveis comuns a ambos os parafusos ..... 74

Figura 31 - Uma outra possível codificação do problema separando, neste caso, todas as variáveis comuns a ambos os parafusos ..... 74

Figura 32 - Gráfico representativo da Evolução de cada uma das 64 Codificações Estudadas ..... 76

Figura 33 - Exemplo de uma geometria do parafuso convencional [1]..... 81

Figura 34 - Diferentes tipos de parafusos, convencional e os de barreira [1]... 82

# ÍNDICE DE TABELAS

Tabela 1 - Operações necessárias para a implementação de um algoritmo genético básico. ..... 15

Tabela 2 - Etapas do Operador *SBX* ..... 21

Tabela 3 - Etapas da Mutação Polinomial ..... 23

Tabela 4 - Etapas do Operador *PCX* ..... 24

Tabela 5 - Etapas do modelo G3 ..... 25

Tabela 6 - Domínios das variáveis do problema ..... 42

Tabela 7 - Descrição e domínios dos objectivos a optimizar no problema. ..... 45

Tabela 8- Frequências relativas das soluções devolvidas pelo programa de simulação de extrusão ..... 46

Tabela 9 - Aplicação do Teste de Hipóteses Kolmogorov-Smirnov para comparação dos dois algoritmos em estudo ..... 52

Tabela 10 - Resultados da aplicação do testes estatistico Kolmogorov-Smirnov para verificar se a geração das soluções geradas pelo PCX segue ou não uma distribuição Normal ..... 54

Tabela 11 - Pesos utilizados na função soma pesada, para cada objectivo em estudo ..... 69

Tabela 12 - Resultados obtidos pela aplicação de teste estatístico *Kolmogorov-Smirnov* para comparação do desempenho dos algoritmos paralelizado e não paralelizado ..... 71

Tabela 13 - Resultados obtidos pela aplicação de teste estatístico *Wilkcoxon rank sum test (Mann-Whitney)* para verificação de qual algoritmo (paralelizado ou não)se adequa melhor ao problema, isto é, qual apresenta melhor desempenho ..... 72

Tabela 14 - Emparelhamento dos resultados obtidos na primeira execução do algoritmo, estando em estudo neste caso, a variável *L1*..... 77

Tabela 15 - Resultados obtidos pela aplicaçao do teste de Friedman com replicação dos dados das quatro execuções, para as seis variáveis em estudo. ..... 78

Tabela 16 - Resultados obtidos pela aplicação do teste de Wilcoxon (Mann-Withney) aos hipervolumes das soluções obtidas pela optimização de duas funções soma pesada ..... 80

# 1. INTRODUÇÃO

## 1.1 Motivação

Esta dissertação foi efectuada no âmbito da Bolsa de Investigação do Projecto POCTI/EME/48448/2002, da Fundação para a Ciência e Tecnologia, que é uma parceria entre Centro de Sistemas Inteligentes (*CSI*) da Universidade do Algarve e o Instituto de Polímeros e Compósitos da Universidade do Minho. O trabalho foi realizado no *CSI*, sob orientação do professor Carlos Fonseca da Universidade do Algarve e do professor Paulo Quaresma da Universidade de Évora. O objectivo do trabalho é a aplicação dos algoritmos evolutivos ao processo de extrusão de polímeros, mais especificamente ao design de parafusos convencional e barreira.

A extrusão de polímeros é uma das tecnologias mais relevantes no fabrico de peças de plástico. É usada geralmente para fabricar produtos tais como: tubagens, canos, produtos para indústrias automóveis e médicas, filamentos e fibras para têxteis, fios metálicos e cabos eléctricos, etc. O processo de extrusão consiste, basicamente, num parafuso tipo-Archimedes rodando a uma velocidade constante dentro de um cilindro aquecido. Normalmente, o polímero sob a forma de grãos sólidos é deitado no alimentador automático caindo através da gravidade para dentro do cilindro. De seguida, a acção da rotação do parafuso arrasta o polímero para a frente onde é progressivamente fundido e pressurizado. Esta pressurização força a que o polímero fundido flua através do cilindro para o molde que atribui a forma final ao produto. O desempenho deste processo é fortemente dependente das condições operatórias da máquina (velocidade de rotação e o perfil de temperaturas) e da geometria do sistema.

No sentido de ultrapassar algumas importantes limitações dos parafusos convencionais tipicamente usados, nomeadamente no que diz respeito eficiência na fusão, foram desenvolvidos os parafusos de barreira (*Maillefer Screw* (*MS*), *Barr Screw* (*BS*), *Dray and Lawrence Screw* (*DLS*) e *Kim Screw* (*KS*)). O objectivo é aumentar a zona de contacto entre os sólidos e as paredes metálicas da extrusora, e separar o fundido do sólido para assim melhorar o desempenho e a estabilidade do processo. No entanto, estes melhoramentos na eficiência da fusão e no processo de estabilidade estão fortemente dependentes do tipo de parafuso barreira e dos parâmetros geométricos usados, isto é, estão dependentes do design do parafuso.

O design de parafusos para a extrusão de polímeros pode ser vistos como um problema de optimização multiobjectivo envolvendo dois tipos de variáveis de decisão: i) o tipo de parafuso (discreta) e ii) os parâmetros geométricos correspondentes e as condições operatórias (contínuos). Esta optimização deve ser efectuada com o objectivo de maximizar o desempenho da extrusora, o qual é normalmente quantificado através da maximização do débito e do grau da mistura, e da minimização do comprimento do parafuso requerido para a fusão, da temperatura à saída do molde e do consumo de energia.

## 1.2 Estrutura da Tese

Com o objectivo de contextualizar a tese, é apresentado no Capítulo 2 uma descrição do processo de extrusão de polímeros bem como das técnicas de optimização já usadas neste problema. O Capítulo 3 apresenta certas características dos algoritmos evolutivos terminando com a descrição de um tipo de algoritmos aconselhados a problemas de engenharia. No Capítulo 4 introduz-se alguns conceitos de optimização multiobjectivo juntamente com técnicas de avaliação de optimizadores. O Capítulo 5 apresenta o estudo efectuado ao problema da extrusão, nomeadamente, a sua formulação e o

estudo dos operadores mais adequados. No Capítulo 6 é então proposto o algoritmo e as experiências para determinação da codificação mais adequada ao problema. Finalmente, o Capítulo 7 apresenta as conclusões finais do trabalho.

## 1.3 Objectivos e Principais Contribuições

O objectivo principal deste estudo é a implementação de um algoritmo evolutivo multiobjectivo para o desenho da geometria de parafusos para a extrusão de polímeros. Este é um processo industrial amplamente utilizado para o fabrico dos vulgares plásticos. A metodologia aqui apresentada descobre qual o tipo de parafuso mais adequado e respectivos parâmetros óptimos para um determinado tipo de polímero e determinadas condições operatórias, definidos antes da optimização. Este problema envolve múltiplos objectivos a optimizar: maximização da quantidade de output e do grau da mistura, e minimização do comprimento do parafuso requerido para a fusão, da temperatura à saída do molde e do consumo de energia.

As contribuições deste trabalho dividem-se em duas áreas, as relacionadas com o desenvolvimento da metodologia de optimização para este problema, e as relacionadas com a área da computação evolutiva em geral.

Entre as primeiras incluem-se:

- O estudo das características da *fitness landscape* implícita no simulador do processo de extrusão, e consequente escolha de um operador de recombinação mais adequado à pesquisa nesse espaço;
- Desenvolvimento de uma codificação hierárquica adequada ao problema em estudo que se mostrou estatisticamente como sendo aquela que permitia maior desempenho à optimização e que conseguia a maior redução do espaço de pesquisa.

Entre as contribuições para a área da computação evolutiva, encontram-se:

- O desenvolvimento de uma versão do algoritmo *G3PCX* [1] com processamento distribuído e optimização multiobjectivo;
- Estudo teórico do operador *PCX* [1] e consequente dedução da expressão matemática que governa o seu comportamento, tornando explícito o cálculo da distribuição normal subjacente ao funcionamento do algoritmo.

Um resumo do trabalho desenvolvido nesta tese foi apresentado na conferência *European Chapter on Combinatorial Optimization (ECCOXIX - http://paginas.fe.up.pt/~ecco/index.php.html )* realizada no Porto de 11 a 13 de Maio de 2006 [2].

# 2. O PROCESSO DE EXTRUSÃO DE POLÍMEROS

## 2.1 Introdução

O objectivo deste capítulo é de inicialmente, na secção 2.1, dar a conhecer o processo de extrusão de polímeros para, na secção 2.2, mostrar como este processo pode ser encarado como um problema de optimização multiobjectivo, e finalmente na secção 2.3, fazer o enquadramento deste trabalho com abordagens anteriormente efectuadas.

A extrusão de polímeros é uma das tecnologias mais relevantes no fabrico de peças de plástico. É usada geralmente para fabricar produtos tais como: tubagens, filamentos, fibras, revestimentos para cabos eléctricos, perfis e filmes. O processo de extrusão consiste basicamente num parafuso tipo-Archimedes rodando a uma velocidade constante dentro de um cilindro aquecido, como pode ser observado na Figura 1. Normalmente, o polímero sob a forma de grãos sólidos é colocado no alimentador automático caindo através da gravidade para dentro do cilindro. De seguida, a acção da rotação do parafuso arrasta o polímero para a frente onde é progressivamente fundido e pressurizado. Esta pressurização força a que o polímero fundido flua através do cilindro para a fieira que atribui a forma final ao produto.

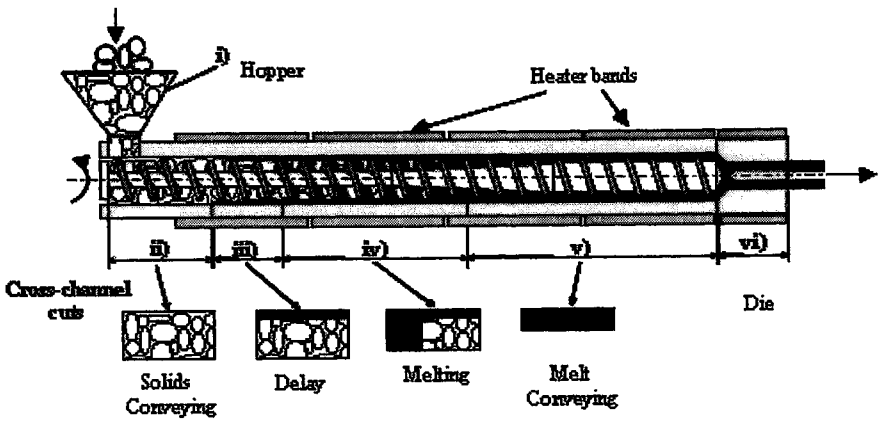



**Figura 1 – Extrusora com um único parafuso convencional e zonas funcionais correspondentes [3]**

O trabalho teórico e prático que tem vindo a ser desenvolvido nas últimas quatro décadas permite a compreensão dos fenómenos físicos que ocorrem dentro da extrusora. Tal como ilustra a Figura 1, o processo pode ser sub-dividido em seis zonas adjacentes e individuais: os grãos sólidos a serem transportados através da gravidade para o cilindro, o arraste dos sólidos devido ao atrito, o transporte dos grãos sólidos envoltos por polímeros fundidos, fusão de acordo com um mecanismo de fusão específico, transporte do fundido nas últimas secções do parafuso e pressurização do fundido para o molde. Assim, a modelação deste processo consiste em resolver as equações que regulam cada um destas etapas do processo, combinando com certas propriedades dos polímeros, com a geometria do parafuso e com as condições operatórias, juntamente com as condições limites relevantes. Com isto pode-se prever as respostas da extrusora em relação a alguns parâmetros importantes tais como o débito, a energia consumida, a temperatura do fundido à saída da fieira, o grau de mistura e a dissipação viscosa.

Com o objectivo de ultrapassar algumas das importantes limitações que os parafusos tipicamente têm, nomeadamente no que diz respeito à sua eficiência na fusão, foram desenvolvidos parafusos barreira (Figura 2 e Figura 4). O objectivo é o de aumentar a área de contacto entre os sólidos e as paredes metálicas do extrusora e separar o fundido do sólido para assim aumentar o

desempenho e a estabilidade. Foram desenvolvidos vários tipos de parafusos barreira, como pode ser visto na Figura 2 e Figura 4.

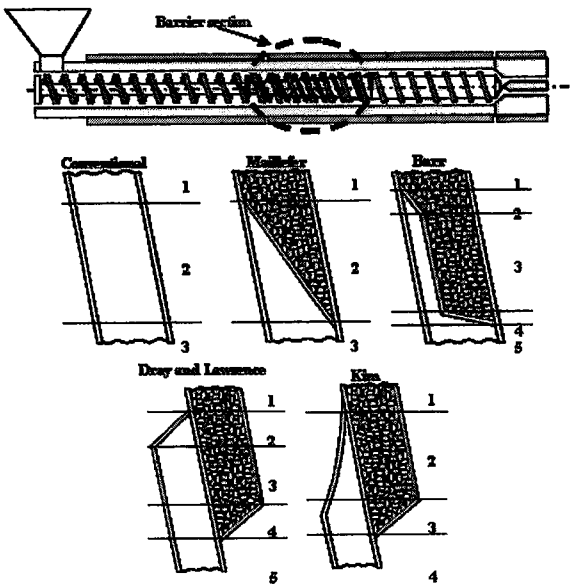


**Figura 2 – Tipos de parafusos convencional e barreira: CS – Conventional screw, MS – Maillefer Screw, BS – Barr Screw, DLS – Dray and Lawrence Screw and KS – Kim Screw. [3]**

No entanto, estes melhoramentos na eficiência da fusão e no processo de estabilidade, são fortemente dependentes do tipo de parafuso (convencional ou barreira) e dos parâmetros específicos da geometria usados. Por esse motivo, o principal objectivo deste trabalho é desenvolver uma metodologia geral de optimização que implemente automaticamente uma estratégia de design automático da geometria dos parafusos da extrusora.

## 2.2 A extrusão de polímeros como um problema de optimização

Antes de começar a manufacturar um novo produto, é frequentemente necessário definir o melhor parafuso e as condições de funcionamento que irão processar mais eficientemente o polímero seleccionado. O desempenho do processo é usualmente avaliado em termos do débito, da qualidade da mistura, do comprimento do parafuso requerido para a fusão, do consumo de energia,

do tempo médio de estada dentro da extrusora e do nível da dissipação viscosa. Para uma dada geometria do equipamento e matéria-prima a usar, as variáveis operacionais principais são a frequência de rotação do parafuso e o perfil de temperaturas no cilindro.

Existe um tipo de parafuso convencional e vários tipos de parafusos barreira que podem ser usados no processo de extrusão. O parafuso convencional tem três secções diferentes. Os parâmetros a optimizar são o tamanho das secções 1 e 2 (*L1 e L2*), a profundidade do canal das secções 1 e 3 (*D1 e D3*), o passo do parafuso (*P*) e a espessura do filete (*E*). Vejamos um exemplo de uma possível geometria do parafuso convencional (Figura 3).

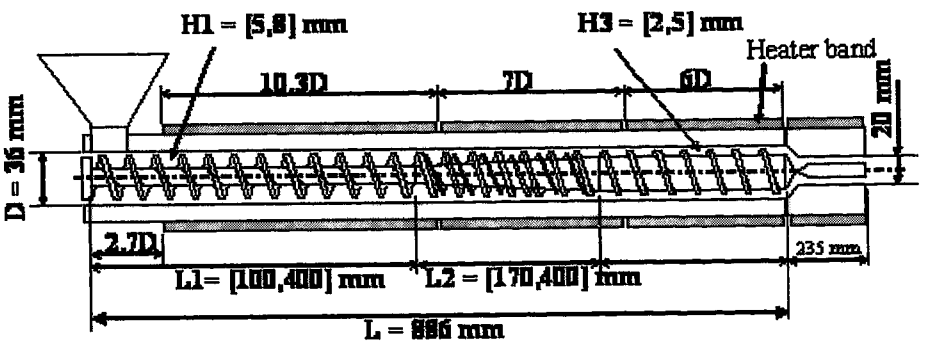



Figura 3 - Exemplo de uma geometria do parafuso convencional [3]

A geometria do parafuso barreira *Maillefer* é idêntica a este parafuso convencional, à excepção da existência de uma barreira que separa o sólido do fundido. Assim, o parafuso *Maillefer* tem dois parâmetros adicionais para optimizar, a largura da barreira (*Wf*) e a folga da barreira (*Hf*), isto é, a distância entre a barreira e o cilindro (Figura 4). Tal como referido anteriormente, existem outros tipos de parafusos barreira mas por uma questão de simplificação das experiências realizadas, optou-se por, neste trabalho, usar apenas os tipos de parafuso convencional e o de barreira Maillefer, pois será relativamente fácil a extensão para os outros tipos de parafusos.

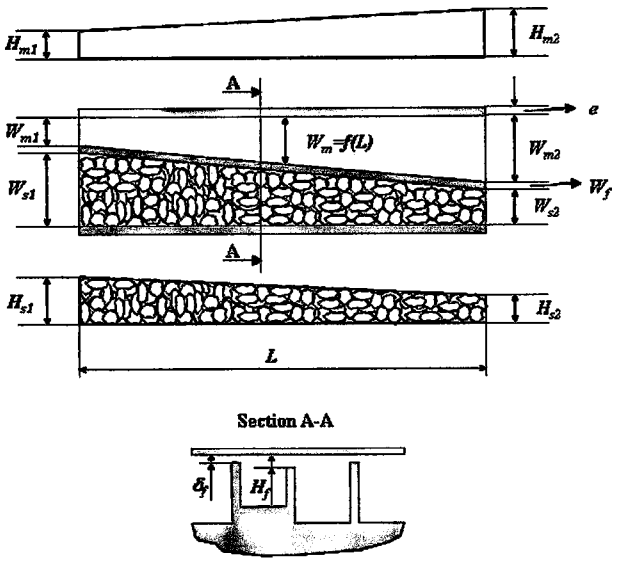



**Figura 4 - Geometria básica da secção barreira [1]**

## 2.2.1. Abordagens anteriores

Os parceiros deste projecto na Universidade do Minho já efectuaram alguns estudos no mesmo sentido dos desenvolvidos neste trabalho, usando um algoritmo multiobjectivo por eles criado, o *RPSGAe* [4]. Este algoritmo implementa uma técnica de *clustering* conhecida como *Complete-Linkage Method*, que compara a proximidade das soluções usando uma medida de distância entre elas e as agrega quando a distância entre elas não ultrapassa um valor pré-definido, essas soluções são agregadas. O *RPSGAe* segue os mesmos passos que um algoritmo genético tradicional à excepção de definir uma população externa (elitista) e de usar uma avaliação de fitness específica (Algoritmo 1). Começa por definir aleatoriamente uma população interna de tamanho *N* e cria uma população externa deixando-a vazia. As operações seguintes são aplicadas em cada geração:

- A população interna é avaliada através da rotina de modelação;
- A *fitness* é calculada usando a técnica de *clustering* (Algoritmo 2) especificada abaixo;
- Copia-se um número fixo de indivíduos para a população externa até esta estar cheia;

- Aplica-se novamente o Algoritmo 2 e os indivíduos da população externa são ordenados;
- Um número, pré-definido, de melhores indivíduos são incorporados na população interna, substituindo os indivíduos com menores valores de fitness;
- Posteriormente são aplicados os operadores de reprodução, de *crossover* e de mutação.

*Algoritmo 1 (RPSGAe):*

Inicializar aleatoriamente a população (interna)
Criar a população externa (vazia)
***While** not Stop-Condition do*
    Avaliar população interna
    Calcular a *Fitness* de todos os indivíduos usando Algoritmo 2
    Copiar os melhores indivíduos para a população externa
    *If* a população externa estiver cheia
        Aplicar Algoritmo 2 a esta população
        Copiar os melhores indivíduos para a população interna
    ***End if***
    Seleccionar os indivíduos para reprodução
    *Crossover*
    Mutação
***End while***

O Algoritmo 2 começa com a definição do número de *ranks*, *Nranks*, e coloca o *rank* de cada indivíduo *Rank[i],* a 0. Para cada *rank*, *r*, reduz-se a população a *NR* indivíduos (onde *NR* é o número de indivíduos de cada *rank*) usando a técnica de *clustering*. De seguida, é atribuído um *rank r* a esses *NR* indivíduos. O algoritmo termina quando é atingido o número pré-definido de *ranks*. Finalmente, calcula-se a fitness de cada indivíduo através da função linear de escalonamento:

$$F_i = 2 - SP + \frac{2(SP-1)(N_{ranks} + 1 - Rank[i])}{N_{ranks}} \quad (1)$$

Onde SP é a pressão selectiva ($1 \leq SP \leq 2$).

***Algoritmo 2 (Clustering):***

Definição de $N_{ranks}$

*Rank[i]*=0

r=1

**do {**

$NR = r(\frac{N}{N_{ranks}})$

Reduzir a população a *NR* indivíduos

*r=r+1*

**while** ($r<N_{ranks}$)

Calcular fitness

**End**

Nesse trabalho [4], a codificação implementada faz uso de um gene que serve de controlador das partes activas do cromossoma, o que faz com que tenha uma codificação de alguma forma estruturada mas não há aqui partilha de variáveis entre os parafusos, isto é, cada parafuso tem o seu conjunto de variáveis. Com este tipo de codificação surge o problema do tamanho do cromossoma ir crescendo à medida que se consideravam mais parafusos no processo de optimização.

## 2.3 Conclusão

O problema em estudo apresenta alguma complexidade, uma vez que estão envolvidos vários tipos de parafusos com um número razoável de variáveis e

com um elevado número de objectivos a optimizar, uns de maximização, outros de minimização, o que envolve a construção de um algoritmo bastante cuidado. No capítulo 3 encontram-se descritas as técnicas de optimização estudadas neste trabalho.

# 3. OPTIMIZAÇÃO EVOLUTIVA

## 3.1 Introdução

Os princípios defendidos por *Darwin* afirmavam que os seres vivos que não se adaptam ao seu ambiente têm menores probabilidades de sobreviver. Os mais aptos, pelo contrário, vivem e reproduzem-se mais passando para os filhos as suas boas características. Os algoritmos evolutivos são a versão computacional dos princípios da selecção natural. Existem quatro grandes áreas nos algoritmos evolutivos: algoritmos genéticos, programação genética, estratégias evolutivas e programação evolutiva. Os algoritmos genéticos foram os que adquiriram maior notoriedade e os escolhidos para usar neste trabalho.

## 3.2 Algoritmos Genéticos

*"Um algoritmo genético faz implicitamente o que é inviável explicitamente"*. Esta foi uma definição dada certa vez por *Jonh Holland*. É por este motivo que, nas últimas décadas, os algoritmos genéticos vêm sendo muito utilizados como métodos de busca e optimização em vários domínios. Outras razões para o seu sucesso relacionam-se com o carácter global da busca que eles realizam, com a facilidade da sua utilização e com a vasta aplicabilidade nos mais diferentes domínios [5].

Os algoritmos genéticos são algoritmos de pesquisa que reflectem de uma forma muito simplificada alguns processos da evolução natural, tal como, por exemplo, as redes neuronais artificiais se aproximam ao processamento

neuronal biológico. Normalmente, os engenheiros e os informáticos não se interessam tanto pelos fundamentos biológicos dos algoritmos genéticos como pela sua utilidade como ferramentas de análise.

Os algoritmos da computação evolutiva trabalham com **populações** de pontos, em vez de um único ponto. Cada "ponto" é um vector no hiperespaço representando uma solução (potencial ou candidata) do problema de optimização. A população é assim apenas um conjunto de vectores desse hiperespaço. Cada vector representa um **indivíduo** da população sendo também por vezes referido **cromossoma** devido à analogia com a representação genética dos organismos.

Devido ao facto dos números reais serem frequentemente codificados nos algoritmos genéticos como números binários, a dimensionalidade do vector do problema pode ser diferente da dimensionalidade do cromossoma, com codificação binária. Quando a codificação é real, o número de elementos em cada vector (indivíduo) iguala o número de parâmetros reais no problema de optimização. O **parâmetro (ou variável de decisão)** do vector corresponde geralmente a um parâmetro, ou dimensão, do vector numérico. Cada elemento pode ser codificado em qualquer número de bits, dependendo da representação de cada variável de decisão. O número total de bits define a dimensão do hiperespaço sob pesquisa.

### 3.2.1. Algoritmo Genético Básico

A sequência de operações necessárias para a implementação de um algoritmo genético básico é apresentada na Tabela 1.

Tabela 1 - Operações necessárias para a implementação de um algoritmo genético básico.

1. Inicialização da população.
2. Cálculo da fitness para cada indivíduo da população.
3. Reprodução dos indivíduos seleccionados para formar uma nova população.
4. Aplicar crossover e mutação à população.
5. Repete os passos 2, 3 e 4 até que alguma condição seja atingida

Em algumas implementações de algoritmos genéticos, outras operações para além do *crossover* e da mutação são aplicadas no passo 4. No entanto, o *crossover* é considerado por muitos, a operação essencial dos algoritmos genéticos.

### 3.2.2. Componentes do Algoritmo Genético

Para optimizar um determinado problema fazendo uso de um algoritmo genético, alguns componentes têm de ser inicialmente definidos, tais como:

- A representação genética das potências soluções do problema;
- A função de avaliação (Função de *Fitness*);
- O critério de selecção;
- Os operadores genéticos que irão afectar a geração dos filhos;
- Os valores dos parâmetros envolvidos no algoritmo genético (tais como, o tamanho da população, o critério de paragem, as probabilidades de mutação e de *crossover*,...)

## Representação Genética

Primeiro que tudo, tem de se definir a **representação genética** isto é, se o cromossoma irá ter uma representação binária ou real. Na codificação binária, cada variável de decisão é representada em mais do que um gene. Vejamos o seguinte exemplo apresentado na Figura 5.

| | $X_1$ | | | $X_2$ | | | $X_3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| **Genótipo** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| **Fenótipo** | 2 | | | 3 | | | 4 | | |

**Figura 5 - Representação Binária**

Este cromossoma está a representar as variáveis $X_1 = 2$, $X_2 = 3$ e $X_3 = 4$.

No caso deste trabalho, a representação escolhida é uma representação real pois estamos a trabalhar com variáveis contínuas, ficando cada variável de decisão a ser representada por um único gene, tal como mostra a Figura 6.

| $X_1$ | $X_2$ | $X_3$ | ... | $X_n$ |
|---|---|---|---|---|
| 3.2 | 2.5 | 5.4 | ... | 4.1 |

**Figura 6 - Representação Real**

Este cromossoma está a representar as variáveis $X_1 = 3.2$, $X_2 = 2.5$, $X_3 = 5.4$ e $X_n = 4.1$.

## Função de Fitness

Nos algoritmos genéticos, os indivíduos são avaliados de acordo com uma função objectivo que define o problema em estudo, fornecendo uma medida representativa da forma como os indivíduos se comportam no domínio do problema. Ao codificarem-se valores reais em cromossomas de representação real, o cromossoma (genótipo) é igual ao fenótipo, o que significa que basta

substituir os valores das variáveis de decisão na função objectivo dada para determinar o seu valor de aptidão. Quando se trata de uma representação binária, o cromossoma (genótipo) tem de ser transformado em fenótipo, e os valores do fenótipo substituídos na função objectivo dada para determinar o seu valor de aptidão.

O cálculo dos valores de fitness pode ser uma tarefa conceptualmente simples, podendo ter no entanto uma certa complexidade na sua implementação de forma a optimizar a eficiência do algoritmo genético na pesquisa do espaço do problema.

## Selecção de Indivíduos

Quanto aos operadores genéticos, um aspecto a definir é qual o tipo de **selecção** a usar. O objectivo principal da selecção é favorecer as melhores soluções de uma população de acordo com o seu valor de aptidão. Existem vários operadores de selecção propostos na literatura, pretendendo todos eles dar mais probabilidade de escolha a uma solução com maior valor de aptidão. A diferença entre os vários tipos de operadores reside na maneira como atribuem a probabilidade de um indivíduo ser escolhido para reprodução dada a sua fitness, relativamente aos restantes indivíduos da geração anterior. Neste trabalho, o tipo de selecção usado foi o *Tournament Selection*, que consiste na selecção aleatória de um determinado número T de indivíduos da população e a escolha do indivíduo nesse conjunto que apresenta melhor valor de fitness. Esse indivíduo ganha o torneio e é o seleccionado.

## Operador genético *Crossover*

Um outro operador genético a ter em conta é o ***crossover***. Este é considerado como o operador mais importante dos algoritmos genéticos pois permite propagar as características dos indivíduos considerados mais aptos, fazendo para isso uma troca de segmentos do cromossoma, entre os indivíduos progenitores que foram seleccionados para se reproduzirem. Existem diversas

formas de fazer este cruzamento, sendo as mais comuns, o *single point crossover* e o *multiple point crossover*.

Para o *single point crossover*, selecciona-se aleatoriamente apenas um único local entre quaisquer dois genes para separar a parte que entrará no novo cromossoma, tal como mostra a Figura 7. O novo indivíduo será constituído pela primeira parte do cromossoma de um pai juntamente com a segunda parte do cromossoma do segundo pai.

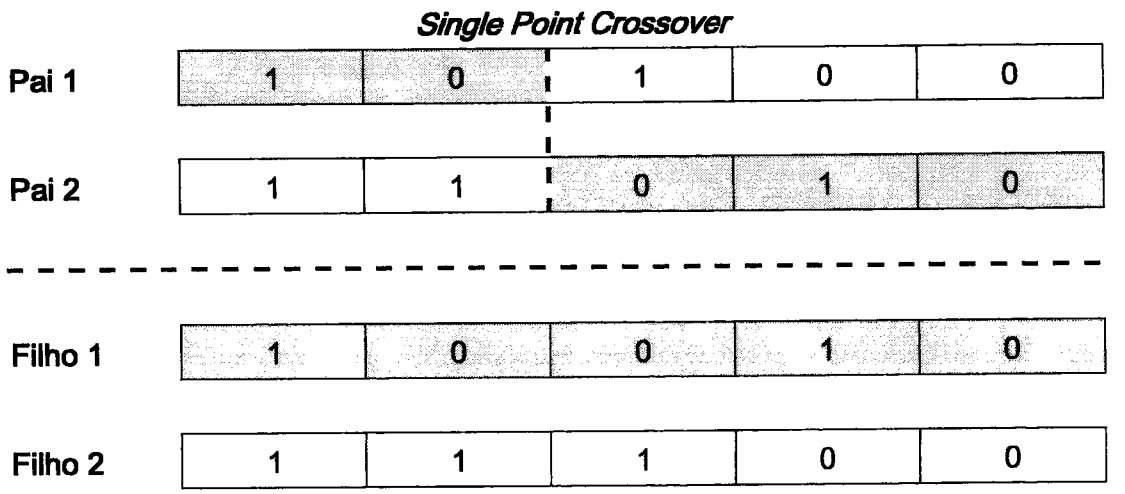
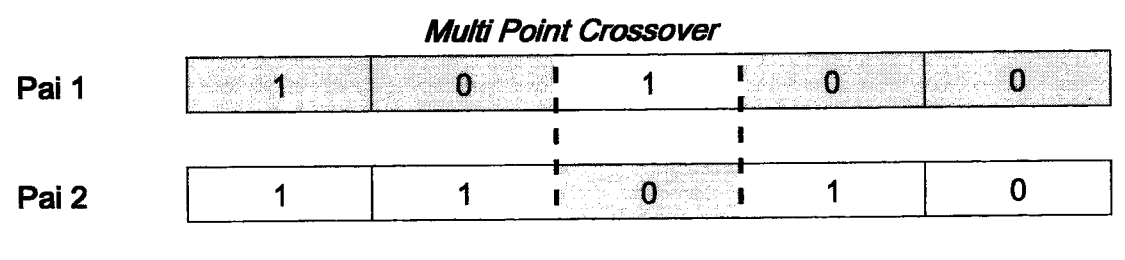



Figura 7 - *Single Point Crossover*

Para o caso do *Multiple Point Crossover*, selecciona-se aleatoriamente mais do que um local entre quaisquer dois genes para separar as partes que entrarão no novo cromossoma, tal como mosta a Figura 8. O cromossoma do filho ficará então constituído pela primeira parte do primeiro pai juntamente com a segunda parte do segundo pai, seguido pela terceira parte do primeiro, e assim sucessivamente.



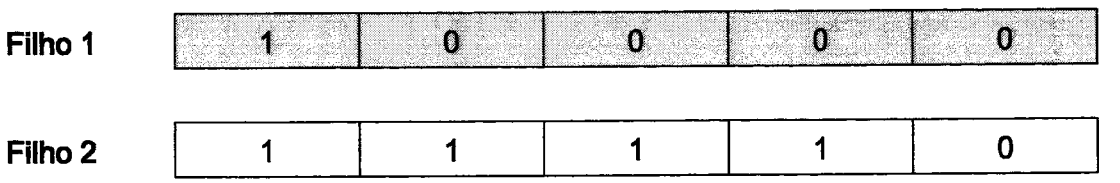



Figura 8 - *Multiple Point Crossover*

Para codificações reais são também utilizados outros tipos de crossover, tais como os usados neste trabalho, nomeadamente o ***Simulated Binary Crossover (SBX)*** [6] e o ***Parent Centric Crossover (PCX)*** [1], cujas descrições se encontram na secção 3.2.3.

## Operador Genético Mutação

A **mutação** é um operador que altera, com uma dada probabilidade, o valor de cada gene num cromossoma de um indivíduo. É um operador que permite a introdução e a manutenção de diversidade genética na população. Quando a representação utilizada é a binária, um bit sofre uma mutação quando é alterado de 0 para 1 ou de 1 para 0.

A figura 9 mostra o operador mutação aplicado ao segundo bit.

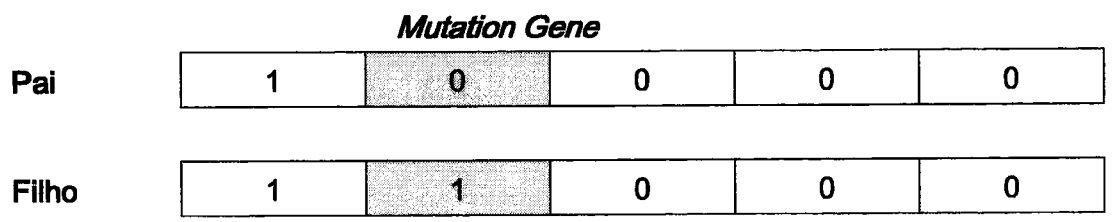



Figura 9 - *One Bit Mutation*

Este operador é aplicado depois do cruzamento e a probabilidade de mutação está tipicamente compreendida entre 0.001 e 0.1 em função do comprimento do cromossoma. Uma das suas principais funcionalidades é a de permitir que se chegue a um qualquer ponto do espaço de pesquisa ajudando assim o algoritmo a avançar quando este fica preso num óptimo local. De um modo

geral, é preciso ter cuidado com a probabilidade de mutação, pois uma taxa elevada pode tornar a busca essencialmente aleatória, além de aumentar a possibilidade de que uma boa solução seja destruída, e uma taxa muito reduzida pode promover uma estagnação num óptimo local havendo assim uma convergência antecipada, não tendo o algoritmo capacidade de explorar outras zonas do espaço de pesquisa.

Para codificações reais existem vários operadores de mutação. O que foi usado neste trabalho foi a **Mutação Polinomial** [7] cuja descrição é feita na secção 3.2.3.2.

### 3.2.3. Operadores para Domínios Contínuos

Como técnicas de optimização para o projecto, recorreu-se a dois algoritmos evolutivos: um algoritmo genético para optimização de funções de variáveis reais com a técnica de *crossover SBX* [6] com Mutação Polinomial [7] e ao *G3/PCX* [1], um algoritmo *stready-state* com elitismo. Como fonte de estudo usaram-se os artigos acima mencionados, juntamente com a implementação do algoritmo *G3/PCX* por parte dos autores [8].

No primeiro algoritmo estudado, implementaram-se as seguintes técnicas:

1. representação real;
2. sem elitismo, isto é, os melhores indivíduos não são copiados para a geração seguinte,
3. *tournament selection;*
4. *simulated binary crossover;*
5. *polynomial mutation.*

Descreve-se de seguida o funcionamento das técnicas 4) e 5).

## *Simulated Binary Crossover (SBX)*

O operador *SBX* [6] usa uma distribuição de probabilidade à volta dos dois pais para criar duas soluções filhas. Dois aspectos que permitem a auto adaptabilidade aos algoritmos genéticos reais com o *SBX* são o facto de ser mais provável criar soluções filhas próximas dos pais e o facto da distância entre as soluções filhas ser proporcional à distância entre as soluções pais.

Apresentam-se de seguida as etapas do operador *SBX*:

**Tabela 2 - Etapas do Operador *SBX***

1 – Escolher aleatóriamente $u$ do intervalo [0,1)

2 - Calcular $\beta_q$ usando a seguinte equação:

$$\beta_p = \begin{cases} 2u^{\frac{1}{\eta+1}} & se\ u \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta+1}} & se\ u > 0.5 \end{cases} \qquad (2)$$

3 – Calcular as soluções filhas através das seguintes equações:

$$x_i^{(1,t+1)} = 0.5\left[(1+\beta_q)x_i^{(1+t)} + (1-\beta_q)x_i^{(2,t)}\right] \qquad (3)$$

$$x_i^{(2,t+1)} = 0.5\left[(1-\beta_q)x_i^{(1+t)} + (1+\beta_q)x_i^{(2,t)}\right] \qquad (4)$$

Nota: Um valor razoável do $\eta$ encontra-se entre 2 e 5 [6].

A Figura 10 ilustra as distribuições de probabilidade de soluções filhas calculadas pelo *SBX*.

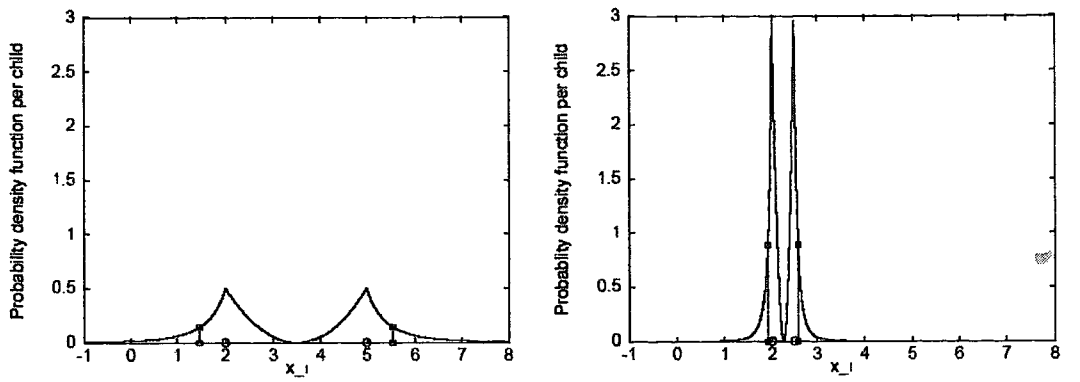



**Figura 10 - Distribuição de probabilidade de soluções filhas calculadas pelo *SBX*[6].**

Estas figuras mostram que se os valores dos pais estão afastados um do outro (primeiro caso), as soluções criadas podem surgir longe dos pais. No segundo caso, se os valores dos pais estão próximos um do outro, então as soluções filhas estarão mais provavelmente próximas dos pais.

## Mutação Polinomial

Este tipo de mutação aplica-se a variáveis contínuas com uma probabilidade baixa [7]. Para uma dada variável, o seu valor é alterado para um valor próximo usando uma distribuição de probabilidade polinomial tendo média no valor actual e variância como função de distribuição de índice *n*. Nesta mutação é aplicado um factor de perturbação definido da seguinte forma:

$$\delta = \frac{c - p}{\Delta_{max}} \tag{5}$$

onde $\Delta_{max}$ é uma quantidade fixa que representa a perturbação máxima permitida no valor do pai *p*, e *c* é o valor a mutar. Tal como o operador de

mutação, o valor a mutar é calculado com uma distribuição de probabilidade que depende do factor de perturbação $\delta$.

$$P(\delta) = 0.5(n+1)(1-|\delta|)^n \tag{6}$$

Esta distribuição de probabilidade é válida no intervalo (-1,1).

Apresentam-se de seguida, as etapas da Mutação Polinomial.

**Tabela 3 - Etapas da Mutação Polinomial**

1 - Escolher aleatoriamente $u$ do intervalo (0,1)

2 - Calcular o factor de perturbação $\bar{\delta}$ correspondente a $u$ (usando a distribuição de probabilidade acima) através da seguinte equação:

$$\bar{\delta} = \begin{cases} (2u)^{\frac{1}{n+1}} - 1, se\ u \le 0.5 \\ 1 - [2(1-u)]^{\frac{1}{n+1}}, se\ u \ge 0.5 \end{cases} \tag{7}$$

3 - Depois disso, o valor mutado é calculado da seguinte forma:

$$c = p + \bar{\delta}\Delta_{\max} \tag{8}$$

A Figura 11 mostra esta distribuição para vários valores do índice, $n$.

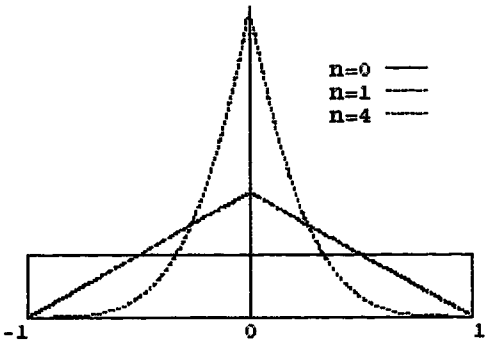



**Figura 11 - Várias distribuições consoante o índice n [7]**

## *Parent-Centric Recombination (PCX)*

A definição do *PCX* em [1] apresenta-nos o operador como capaz de produzir filhos na vizinhança de qualquer um dos pais sob consideração, não realizando portanto qualquer pressão selectiva. Pela introdução posterior do *Generalized Generation Gap Model* (*G3*), toma-se conhecimento que estes pais são escolhidos segundo uma certa lógica: o primeiro é sempre o melhor indivíduo da população e os restantes escolhidos aleatoriamente.

Apresentam-se de seguida as descrições do operador *PCX*, seguido do *Generalized Generation Gap Model* (*G3*).

**Tabela 4 - Etapas do Operador *PCX***

| |
|---|
| 1 – Calcula-se o vector médio $\vec{g}$ dos $u$ pais seleccionados (normalmente são 3 pais). |
| 2 – Para cada filho, é escolhido um pai $\vec{x}^{(p)}$ com igual probabilidade (na implementação usam sempre o melhor pai que coincide com o melhor pai da população –G3) |
| 3 – Calcula-se o vector direcção $\vec{d}^{(p)} = \vec{x}^{(p)} - \vec{g}\,.$ (9) |

4 – Depois disso, para cada um dos outros *(u-1)* pais calculam-se as distâncias perpendiculares $D_i$ à linha $\vec{d}^{(p)}$ e depois encontra-se a média $\overline{D}$ dessas distâncias.

5 – Os filhos são calculados da seguinte forma:

$$\vec{y} = \vec{x}_p + w_\zeta \mid \vec{d}^{(p)} \mid + \sum_{i=1;i\neq p}^{u} w_\eta \overline{D} \vec{e}^{(i)} \qquad (10)$$

onde $\vec{e}^{(i)}$ são as *(u-1)* bases ortogonais que abrangem o subespaço perpendicular a $\vec{d}^{(p)}$. Os parâmetros $w_\zeta$ e $w_\eta$ são variáveis normalmente distribuídas com média zero e com variância $\sigma_\zeta^{\,2}$ e $\sigma_\eta^{\,2}$.

O *G3* descreve-se da seguinte forma:

**Tabela 5 - Etapas do modelo G3**

1 – Da população *P(t)*, seleccionar o melhor pai e os restantes *u-1* seleccionam-se aleatoriamente.

2 – Gerar $\lambda$ filhos dos *u* pais escolhidos, através do *PCX*.

3 – Escolher dois pais aleatoriamente da população.

4 – De uma sub população que combina os dois pais seleccionados em (3) com os $\lambda$ filhos gerados em (2), escolher as duas melhores soluções e substituir os dois pais escolhidos em (2) por estas soluções.

Quer o *SBX*, que o *PCX*, são *Parent-Centric Approaches*, diferindo na distribuição de probabilidade usada.

**Figura 12 - Distribuição de soluções filhas calculadas com três pais, numa abordagem *Parent-Centric* (Algoritmo *PCX*) [1]**

No entanto, quando se aplica *G3* ao *PCX* já não se obtém uma distribuição desta forma, mas sim uma em que as soluções filhas rodeiam apenas o melhor pai.

## 3.3 Algoritmos Genéticos Estruturados (*sGA*)

Durante as últimas décadas, os algoritmos genéticos têm sido estudados por um grande número de investigadores de diversas áreas. Foram-lhes impostas alterações para resolver as dificuldades encontradas na resolução de diferentes problemas mas, apesar dessas modificações, os algoritmos genéticos tradicionais mantêm-se inadequados na resolução de muitos problemas práticos. Em 1992 [9], surge então um novo tipo de algoritmos genéticos, os algoritmos genéticos hierárquicos ou estruturados (*sGA*), que procuram emular a formulação da estrutura biológica do *DNA* de forma a que possa ser concebida uma estrutura genética hierárquica precisa para fins de engenharia. Este novo arranjo dos algoritmos genéticos tem um enorme potencial na modelação e design de sistemas e na resolução de problemas topológicos científicos e de engenharia.

### 3.3.1. Perspectiva Biológica

Na estrutura biológica do *DNA*, a estrutura genética do cromossoma é formada por um conjunto de variações nos genes que são organizadas de uma forma hierárquica. Uma das mais surpreendentes descobertas nos fundamentos da biologia molecular foi a da existência de genes activos (exões) e de genes inactivos (intrões).

A estrutura hierárquica do cromossoma no algoritmo genético pode ser vista como a do *DNA*, com as alterações dos genes estruturais do *DNA* a corresponderem aos genes paramétricos do algoritmo genético, e as sequências regulatórias do *DNA* a serem agora os genes de controlo do algoritmo genético (Figura 13). Para generalizar esta arquitectura, são introduzidos genes de controlo possivelmente com múltiplos níveis numa forma hierárquica (Figura 14). Assim, a activação dos genes paramétricos é gerida pelo valor do gene de controlo do primeiro nível que, por sua vez, é gerida pelo gene de controlo do segundo nível.

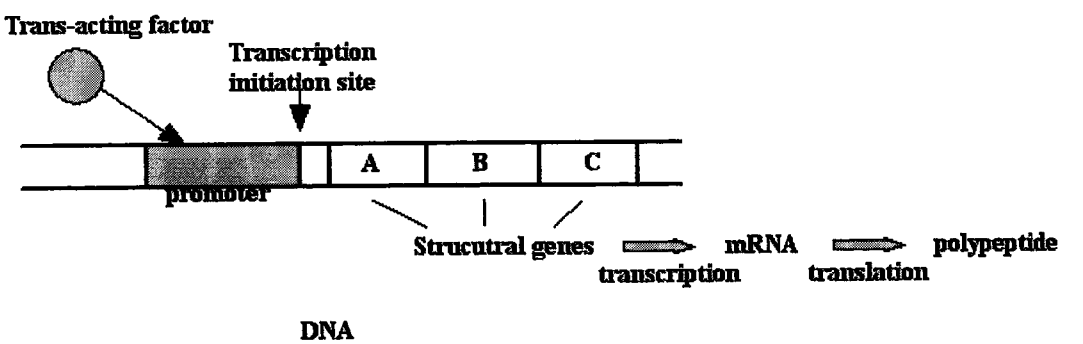



Figura 13 – DNA

Para indicar que um gene de controlo está activo, usa-se, por exemplo, o inteiro "1" (ou determinado intervalo de valores), se estiver inactivo usa-se, por exemplo, o "0" (ou outro intervalo específico).

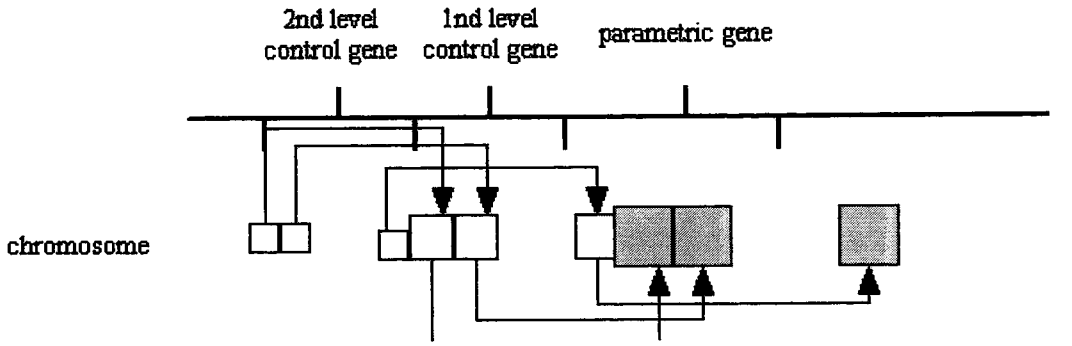



**Figura 14 - Estrutura hierárquica do cromossoma**

Quando o "1" está assinalado, os genes paramétricos associados do nível abaixo ficam activos. Note-se que mesmo quando o gene de controlo está a 0, os genes correspondentes do nível abaixo não desaparecem. Esta arquitectura hierárquica permite que o *sGA* guarde mais informação do que o algoritmo genético tradicional.

Vejamos agora alguns exemplos de codificação estruturada (Figura 15 e Figura 16).

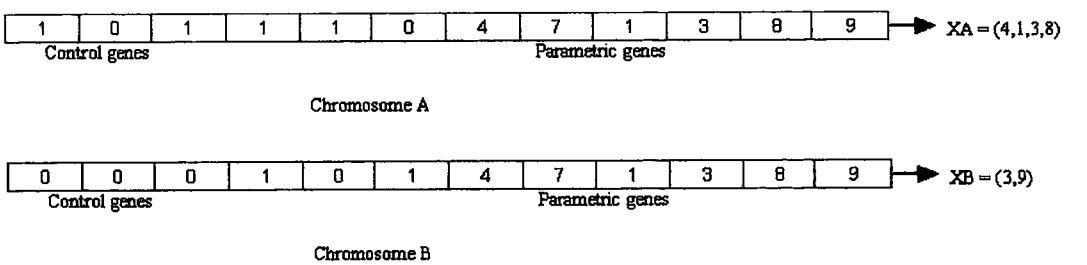



**Figura 15 - Codificação Estruturada**

Na Figura 15, o comprimento de XA é 4, e o comprimento de XB é 2, o que significa que são permitidos fenótipos com diferentes tamanhos dentro da mesma formulação do cromossoma. Assim, o *sGA* irá fazer a procura sob todos os tamanhos possíveis até que o objectivo final seja encontrado.

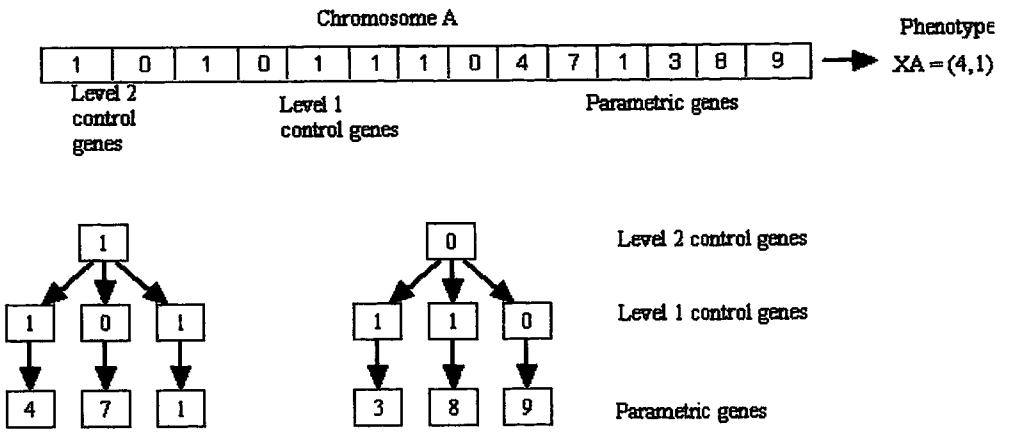



**Figura 16 - Exemplo de um cromossoma e de duas possíveis árvores associadas**

A Figura 16 apresenta um cromossoma estruturado juntamente com duas possíveis árvores com dois níveis de controlo. Como podemos verificar, a segunda árvore encontra-se inactiva pois o gene de controlo do segundo nível está a zero, enquanto que a primeira está activa, estando a codificar o fenótipo (4,1).

### 3.3.2. Níveis *sGA*

Os níveis do *sGA* dependem da complexidade do espaço de pesquisa, isto é, se o problema tem um espaço de pesquisa de um nível, então um *sGA* com dois níveis funciona eficientemente, onde os genes de alto nível podem activar o espaço de soluções alternativo. Quando o problema começa a ser mais complexo necessitará de mais níveis de controlo. Por exemplo, a Figura 16 apresenta dois níveis de controlo.

### 3.3.3. Cromossoma

Um cromossoma é representado por um conjunto de *substrings*, as quais, durante a reprodução, são modificadas pelos operadores genéticos.

Na descodificação para o fenótipo, o cromossoma é interpretado como uma estrutura genómica hierárquica do material genético. Apenas os genes actualmente activos no cromossoma contribuem para a fitness do fenótipo. Os genes passivos são aparentemente neutros e tratados como um material genético redundante durante o processo evolutivo.

### 3.3.4. Diversidade genotípica

Quando a população converge para o seu espaço fenótipo, a diversidade genotípica continua a existir e é a principal característica específica deste modelo. Na maioria dos modelos genéticos formais, a convergência do fenótipo implica a convergência do genótipo com o consequente empobrecimento do indivíduo da população.

## 3.4 Conclusão

Os algoritmos genéticos têm vindo a ser cada vez mais aplicados na resolução de problemas nos mais diversos domínios. No entanto, existem alguns problemas nos quais os algoritmo genéticos apresentam um fraco desempenho.

Para o estudo do problema de optimização de geometrias de extrusoras, vários algoritmos foram estudados nomeadamente, algoritmo genético geracional com *Simulated Binary Crossover* (*SBX*) e *Polinomial Mutation* (*PM*); *Generalized Generation Gap Model* (*G3*) *steady state* com *Parent-Centrix Crossover* (*PCX*); e posterior *Generalized Generation Gap Model* (*G3*) com *Parent-Centrix Crossover* (*PCX*), multiobjectivo, paralelizado e implementado várias codificações (*G4PCX*).

Para avaliar todos estes algoritmos com o objectivo de verificar quais os que se adequam melhor ao problema em estudo, usaram-se algumas técnicas de avaliação de desempenho de optimizadores, técnicas essas que se encontram descritas no capítulo seguinte.

# 4. AVALIAÇÃO DO DESEMPENHO DE OPTIMIZADORES

## 4.1 Introdução

Os problemas multiobjectivo abundam no mundo real, e já vários algoritmos evolutivos foram propostos para conseguirem boas soluções para estes problemas (tais como *MOGA* [10], *NSGA-II* [11], *SPEA-2* [12], etc). No entanto a questão de que métrica usar para comparar o desempenho dos optimizadores continua a ser uma questão de difícil solução. Uma métrica que tem sido muito adoptada é o hipervolume [13], que será descrita na secção 4.3. Outra forma de medir o desempenho de optimizadores multiobjectivo é através da função de aproveitamento que engloba quer a qualidade dos indivíduos não dominados no espaço dos objectivos quer a distribuição dessas soluções ao longo da fronteira de Pareto. Uma descrição mais pormenorizada encontra-se na secção 4.4 deste capítulo.

## 4.2 Algoritmos Genéticos Multiobjectivo

No mundo real encontramos inúmeros problemas multiobjectivo, isto é, que tenham dois ou mais objectivos que precisam de ser optimizados. Vejamos o exemplo de uma fábrica que quer produzir produtos de alta qualidade a baixos custos. Neste caso temos para optimizar dois objectivos, o primeiro pretende *maximizar a qualidade dos produtos* e o segundo pretende *minimizar os custos envolvidos*. Um outro exemplo igualmente comum é o de num automóvel, se pretender que tenha uma potência elevada mas que consuma o mínimo de

combustível. Os objectivos a optimizar neste caso seriam *maximizar a potência do automóvel* e *minimizar o consumo de combustível.*

### 4.2.1. Problema Multiobjectivo

Existem inúmeros problemas multiobjectivo que requerem a optimização simultânea de vários objectivos possivelmente contraditórios. Existem diversas abordagens que transformam as funções objectivo numa única função escalar, resolvendo o problema como um problema de optimização de um único objectivo. No entanto, existem outras abordagens que se concentram em encontrar o conjunto óptimo de *trade-offs*, o conhecido **Conjunto Óptimo de Pareto** [14].

Um conjunto óptimo de Pareto é definido sempre em função da forma como se relacionam cada duas soluções, isto é, se uma domina a outra.

Vejamos algumas definições relacionadas com os Conjuntos Óptimos de Pareto [14].

***Definição 1 (Problema Multiobjectivo):*** Consideremos sem perda de generalidade, um problema multiobjectivo de minimização com $m$ variávies de decisão (parâmetros) e com $n$ objectivos:

$$\begin{aligned} &\textit{Minimizar} \quad y = f(x) = (f1(x),...,f(n)) \\ &\textit{onde} \quad x = (x_1,...,x_m) \in X \\ &\qquad\quad y = (y_1,...,y_n) \in Y \end{aligned} \tag{11}$$

onde **x** é um vector de decisão, $X$ o espaço dos parâmetros, $y$ é o vector objectivo e $Y$ é o espaço dos objectivos.

***Definição 2 (Dominância de Pareto):***

Diz-se que um *vector de decisão* $a \in X$ diz-se que domina um *vector de decisão* $b \in X$ (também representado como $a \prec b$) sse ***f(a)*** é parcialmente menor do que ***f(b)***, isto é,

$$\begin{aligned} &\forall i \in \{1,...,n\}: \quad f_i(a) \leq f_i(b) \\ &\wedge \\ &\exists j \in \{1,...,n\}: \quad f_j(a) < f_i(b) \end{aligned} \tag{12}$$

***Definição 3 (Optimalidade de Pareto):***

Seja $a \in X$ um vector de decisão arbitrário.

1. Diz-se que o vector de decisão $a$ *é não dominado em relação a um conjunto* $X' \subseteq X$ sse não existe nenhum vector em $X'$ que domine $a$; Formalmente:

$$\nexists a' \in X' : a' \prec a \tag{13}$$

2. O vector de decisão $a$ é *Optimo de Pareto* sse $a$ não é dominado em relação a *X*.

***Definição 4 (Frente de Pareto):***

As soluções óptimas de Pareto são habitualmente referidas como soluções eficientes, não dominadas. O conjunto de todos os vectores não dominados é conhecido como o conjunto não dominado, ou como a fronteira de eficiência, do problema. Na Figura 17 está representada a *frente (ou fronteira) de Pareto* para um problema multiobjectivo (neste caso com dois objectivos para optimizar) com uma linha carregada.

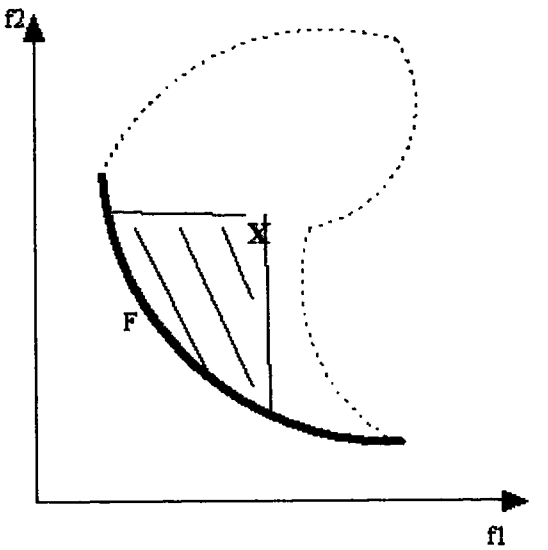



**Figura 17 - Um exemplo de um problema com duas funções objectivos, ambas de minimização. A frente de Pareto está representada com a linha carregada. Nesta figura está também representado o índice de hipervolume do ponto X.**

## 4.3 Indicador de Desempenho e Índice de Hipervolumes

O resultado de um optimizador multiobjectivo é um conjunto de soluções não dominadas. Se o optimizador for estocástico, as aproximações da fronteira de Pareto são aleatórias, logo é importante conhecer qual o seu desempenho. O desempenho de um optimizador multiobjectivo está directamente relacionado com a qualidade das aproximações à fronteira de Pareto. Combinando numa função objectivo de valores reais a qualidade das soluções dos indivíduos com a sua expansão ao longo da fronteira de Pareto, a função de aproveitamento consegue descrever um importante aspecto da distribuição das aproximações aleatórias do conjunto de Pareto, nomeadamente a sua localização. Para uma melhor compreensão destes objectivos, vejamos as seguintes definições apresentadas em Fonseca et al. [16]:

***Definição 1 (Conjunto Aleatório de Pontos Não-Dominados):***

Um conjunto de pontos aleatório $\Theta = \{\theta_1,...,\theta_M \in \Re^d : P(\theta_i \leq \theta_j) = 0, i \neq j\}$, onde quer o número de elementos M quer os próprios elementos $\theta_i$ são aleatórios e $P(0 \leq M < \infty) = 1$, é designado um *conjunto aleatório de pontos não-dominados (Random non-dominated point set (RNP-set)).*

As aproximações aleatórias ao conjunto de Pareto produzidas por optimizadores multiobjectivo estocásticos num problema de *d-objectivos* são conjuntos *RNP* em $\Re^d$.

***Definição 2 (Attained Set ):***

O conjunto aleatório

$$\begin{aligned} Z &= \{z \in \Re^d \mid \theta_1 \leq z \vee \theta_2 \leq z \vee ... \vee \theta_M \leq z\} \\ &= \{z \in \Re^d \mid \Theta \trianglelefteq z\} \end{aligned} \qquad (14)$$

é o conjunto de todas as metas $z \in \Re^d$ alcançados pelo conjunto *RNP* $\Theta$ .

As distribuições de $\Theta$ e $Z$ são equivalentes, pois a caracterização da distribuição de $\Theta$ estabelece automaticamente a distribuição de $Z$, e vice-versa.

***Definição 3 (Indicador de Aproveitamento):***

Seja $I\{.\} = I_{\{.\}}(z)$ a função de indicador. Então, a variável aleatória $b_X(z) = I\{\Theta \trianglelefteq z\}$é denominada como *Indicador de aproveitamento* de $\Theta$ no objectivo $z \in \Re^d$.

***Definição 4 (Função de Aproveitamento):***

A função de aproveitamento de $\Theta = \{\theta_1,...,\theta_M\}$ é definida por

$$\begin{aligned}\alpha_\theta(z) &= P(\theta_1 \leq z \vee \theta_2 \leq z \vee ... \vee \theta_M \leq z) \\ &= P(\theta \lhd\!= z) \\ &= P(\textit{optimizador atingir o objectivo } z \textit{ numa só execução})\end{aligned}$$

Pode ser estimada através da Função de Aproveitamento Empírica (*EAF*)

$$\alpha_r(z) = \frac{1}{r}\sum_{i=1}^{r} I\{\theta_i \rhd\!= z\}, \tag{15}$$

Onde *r* é o número de execuções e *I* é o indicador da função.

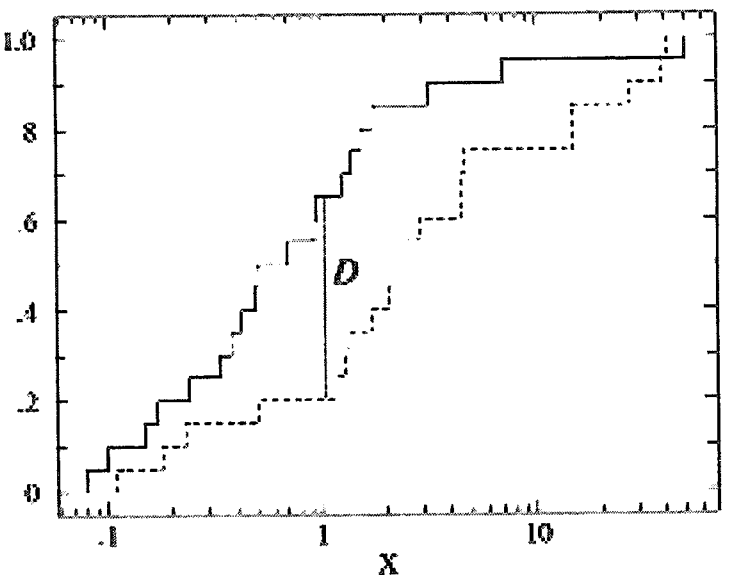



**Figura 18 – Exemplo da aplicação do Testes Estatístico Kolmogorov-Smirnov para Comparação de duas funções comulativas [17]**

Usando as *EAF* podemos aplicar um teste estatístico análogo ao teste de *Kolmogorov-Smirnov* para comparação de diversos algoritmos. Este teste calcula a distância máxima entre duas *EAFs* (*D* na Figura 18). Consideremos o caso da comparação de dois algoritmos. A hipótese nula do teste afirma que as funções de aproveitamento dos dois algoritmos são semelhantes, enquanto

que a hipótese alternativa afirma que as distribuições diferem de algum modo. Se a hipótese nula for rejeitada é possível investigar quais as diferenças que existem entre as duas funções de aproveitamento, especialmente usando um ambiente gráfico interactivo. Em teoria, os métodos para o cálculo da função de aproveitamento funcionam para qualquer número de dimensões, mas actualmente o seu conteúdo para mais do que três objectivos tende a tornar-se inaceitavelmente lento.

## Hipervolumes

O indicador de hipervolume [13] [28] da região dominada por um conjunto de soluções no espaço dos objectivos é uma medida muito usada para determinar para avaliar a qualidade da aproximação à fronteira de Pareto porque combina num simples escalar quer a proximidade das soluções ao conjunto óptimo, quer a sua distribuição ao longo da frente. Para além do indicador de hipervolume possuir boas propriedades matemáticas que outras medidas não possuem, também tem outras não ideais: requer a definição de um ponto de referência (por exemplo o ponto X na Figura 17) no qual os cálculos se irão basear, e é sensível à escala relativa dos objectivos e à presença ou ausência de pontos nos extremos da fronteira.

O *HSO* (*Hypervolume by Slicing* Objectives) é um algoritmo desenvolvido por While et al [15] que é considerado como um dos mais rápidos para o cálculo exacto do hipervolume. Dados m pontos não dominadados em *d* objectivos, o algoritmo *HSO* baseia-se na ideia de processar um objectivo de cada vez. Após a ordenação, a complexidade do algoritmo é de O(n^(d-1)).

O indicador de hipervolume usado neste trabalho é uma melhoria deste *HSO* em três aspectos, nomeadamente o uso de um algoritmo para o caso tridimensional, o conjunto de pontos não dominados é guardado numa estrutura de dados dedicada mantendo a complexidade do espaço linear e, nesta estrutura de dados também é guardado o estado de dominância actual

de cada ponto, assim como os valores intermédios dos hipervolumes, permitindo que ramos da árvore de recursão possam ser podados sempre que os resultados calculados anteriormente possam ser reutilizados. O cálculo deste indicador está a começar a despertar interesse.

## 4.4 Conclusão

Neste capítulo concluiu-se a revisão teórica deste trabalho, com uma breve descrição dos algoritmos genéticos multiobjectivo complementada com a apresentação das técnicas de avaliação de desempenho destes algoritmos usadas em algumas experiências descritas nos capítulos que se seguem.

# 5. OPTIMIZAÇÃO EVOLUTIVA DE EXTRUSORAS

## 5.1 Introdução

Uma vez descrita a componente teórica envolvida no presente trabalho, neste capítulo apresentam-se as experiências e estudos efectuados quer ao problema de extrusão quer aos algoritmos genéticos seleccionados para este trabalho. A formulação do problema surge na próxima secção, apresentando-se na secção 5.3 um estudo às avaliações devolvidas pelo programa de simulação do processo de extrusão. Finalmente, na secção 5.4 apresentam-se os estudos efectuados aos algoritmos genéticos, o primeiro geracional com *Simulated Binary Crossover* (*SBX*) e com a *Polinomial Mutation* (*PM*), e o segundo *steady-state* implementando o *Generalized Generation Gap Model* (*G3*) com *Parent Centric Crossover* (*PCX*), envolvendo comparações de desempenho e estudo teórico do operador *PCX*.

## 5.2 Formulação do Problema

O problema em estudo envolve a optimização da geometria de dois tipos de parafusos simultaneamente, o parafuso tipo convencional e o parafuso tipo barreira *Maillefer*, descritos mais pormenorizadamente no capítulo 2.

O parafuso convencional é descrito pelas seguintes variáveis:

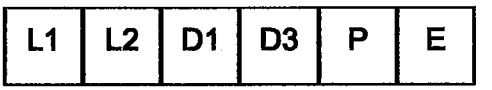



**Figura 19 - Variáveis do parafuso convencional**

Por sua vez, o parafuso de Barreira Maillefer é descrito pelas seguintes variáveis:

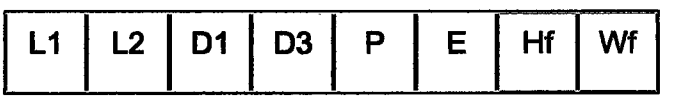



**Figura 20 - Variáveis do parafuso barreira Maillefer**

Repare-se que existem seis variáveis comuns (representadas com um sombreado). No entanto, estas variáveis, embora aparentemente comuns, podem ter comportamentos diferentes nos diferentes parafusos, portanto uma das questões a estudar neste projecto será o significado de cada uma das variáveis em cada um dos casos.

Para tal, na codificação do problema, o cromossoma terá de representar todas as variáveis:

- variáveis do parafuso convencional
  - *L1*
  - *L2*
  - *D1*
  - *D3*
  - *P*
  - *E*
- variáveis do parafuso de barreira Maillefer
  - *L1*
  - *L2*
  - *D1*
  - *D3*
  - *P*
  - *E*

- *Hf*
- *Wf*

Para além destas variáveis, é necessário que o cromossoma represente também as condições operatórias do problema, acrescentando-lhe assim as seguintes variáveis:

- Velocidade de Rotação
- Perfil de Temperaturas
  - *T1*
  - *T2*
  - *T3*

Para que a optimização dos dois parafusos possa decorrer simultaneamente acrescentou-se uma variável extra que serve de gene de controlo, atribuindo assim uma estrutura hierárquica ao cromossoma.

Os valores das condições operatórias foram fixados em 50 rpm, 170°C, 170°C, 170°C respectivamente, uma vez que já foram estudados anteriormente [18]. Quanto aos valores das outras variáveis, variam nos domínios apresentados na Tabela 6.

**Tabela 6 - Domínios das variáveis do problema**

| | Designação da Variável | Representação da Variável | Limite Inferior | Limite Superior |
|---|---|---|---|---|
| Condições Operatórias | Velocidade de Rotação (rpm) | *VR* | Fixo em 50 | |
| | Temperatura 1 (°C) | *T1* | Fixo em 170 | |
| | Temperatura 2 (°C) | *T2* | Fixo em 170 | |
| | Temperatura 3 (°C) | *T3* | Fixo em 170 | |
| Parafuso Convencional | Comprimento da 1ª Secção do Parafuso (mm) | *L1* | 100 | 400 |
| | Comprimento da 2ª Secção do | *L2* | 170 | 400 |

| | | | | |
|---|---|---|---|---|
| | Parafuso (mm) | | | |
| | Diâmetro da 1ª Secção do Parafuso (mm) | *D1* | 20 | 26 |
| | Diâmetro da 3ª Secção do Parafuso (mm) | *D3* | 26 | 32 |
| | Passo do Filete do Parafuso (mm) | *P* | 30 | 42 |
| | Expessura do Filete do Parafuso (mm) | *E* | 3 | 4 |
| Parafuso de Barreira Maillefer | Comprimento da 1ª Secção do Parafuso (mm) | *L1* | 100 | 400 |
| | Comprimento da 2ª Secção do Parafuso (mm) | *L2* | 170 | 400 |
| | Diâmetro da 1ª Secção do Parafuso (mm) | *D1* | 20 | 26 |
| | Diâmetro da 3ª Secção do Parafuso (mm) | *D3* | 26 | 32 |
| | Passo do Filete do Parafuso (mm) | *P* | 30 | 42 |
| | Expessura do Filete do Parafuso (mm) | *E* | 3 | 4 |
| | Altura da Zona de Barreira (mm) | *Hf* | 0.5 | 1 |
| | Largura da Zona de Barreira (mm) | *Wf* | 3 | 4 |

| Gene de Controlo | Tipo de Parafuso | *TP* | 0 | 1 |
|---|---|---|---|---|

Considerando a codificação em que todas as variáveis estão separadas (explicação das codificações na secção 6.4), o cromossoma fica então com a forma:

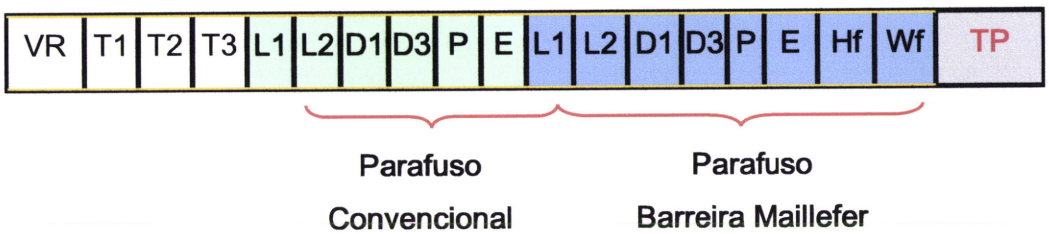



**Figura 21 - Cromossoma considerando todas as variáveis "comuns" aos dois parafusos, separadas.**

Traduzindo em forma de árvore:

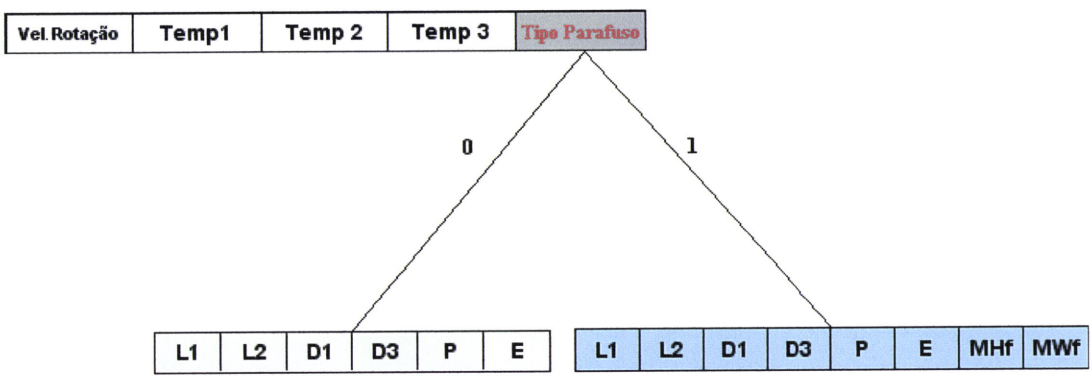



**Figura 22- Árvore associada ao cromossoma, tendo todas as variáveis separadas**

Quanto aos múltiplos objectivos a optimizar, existem oito critérios avaliados mas apenas cinco são considerados para optimização (por indicação do coordenador do projecto). A sua descrição e respectivos domínios apresentam-se na tabela seguinte:

**Tabela 7 - Descrição e domínios dos objectivos a optimizar no problema.**

| Objectivo | Designação do Critério | Critério | Limite Inferior | Limite Superior |
|---|---|---|---|---|
| **Maximização** | Débito (kg/h) | *C1* | 1 | 20 |
| **Minimização** | Comprimento do Parafuso Requerido para a Fusão (m) | *C2* | 0.2 | 0.9 |
| **Minimização** | Temperatura do Fundido (ºC) | *C3* | 150 | 210 |
| **Minimização** | Consumo de Energia Mecânica (W) | *C4* | 0 | 9200 |
| **Maximização** | Qualidade da Mistura; WATS | *C5* | 0 | 1300 |
| | Dissipação Viscosa; T/Tc | *C6* | 0 | 2 |
| | Dissipação Viscosa; Tmax/Tc | *C7* | 0 | 3 |
| | Taxa de Deformação Máxima; 'Taxa*L' | *C8* | 0 | 2000 |

## 5.3 Estudo das Soluções Obtidas pelo Simulador de Extrusão

Ao analisar os resultados obtidos nas primeiras experiências realizadas (descritas na secção seguinte), notou-se que muitas das soluções devolvidas apresentavam erros ou estavam fora dos limites das variáveis do problema. Então, efectuou-se um estudo sobre as soluções obtidas pelo programa que simula o processo de extrusão, com o objectivo de qualificar essas soluções

nos algoritmos de optimização implementados. Para tal, começou-se por gerar 2000 cromossomas aleatoriamente dentro dos limites das variáveis. Seguidamente, essas soluções foram avaliadas pelo programa de simulação de extrusão e os resultados obtidos foram estudados através das frequências relativas apresentadas na Tabela 8 e na Figura 23.

| Com Erro | | Sem Erro | |
|---|---|---|---|
| **Tipo** | **31,50%** | **68,50%** | |
| **300** | 0,10% | 42,43% | **DentroLimites** |
| **401** | 0,20% | 25,87% | **ForaC5** |
| **407** | 0,20% | 0,15% | **ForaC4** |
| **408** | 0,05% | 0,05% | **ForaC1,C4,C5** |
| **701** | 9,89% | | |
| **801** | 0,25% | | |
| **2000** | 0,05% | | |
| **2001** | 5,54% | | |
| **5000** | 15,23% | | |

**Tabela 8- Frequências relativas das soluções devolvidas pelo programa de simulação de extrusão**

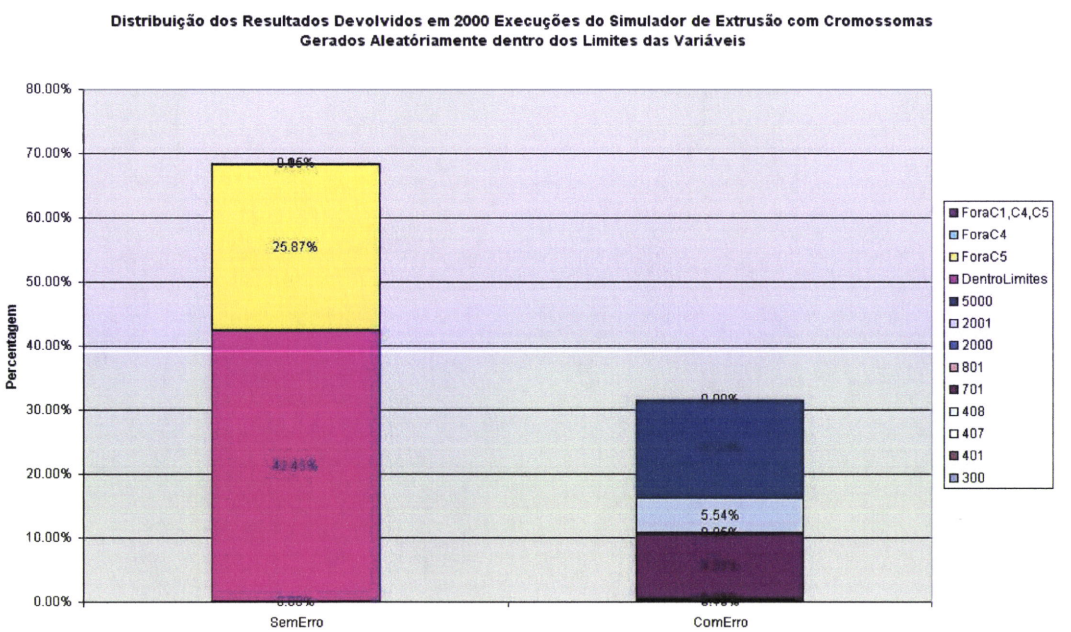



**Figura 23 - Distribuição dos resultados obtidos em 2000 execuções do programa de simulação do processo de extrusão**

Como se pode observar quer na tabela quer no gráfico anteriores, 31,5% das soluções apresentam erros de algum dos tipos e 25,87% das soluções têm os seus valores fora dos limites *C5*.

Dada a elevada percentagem de erros e de soluções fora de limites adaptaram-se os algoritmos estudados de forma a eliminar da população as soluções que apresentavam algum tipo de erro. Essa eliminação consiste em atribuir a essa solução com erro um valor de fitness de 1000000 (este valor é positivo pois as funções são de, ou foram transformadas para, minimização). Em relação às soluções que apresentavam valores fora de limites do critério *C5*, atribuiu-se uma penalização a cada uma dessas soluções. Esta penalização consiste na distância euclideana da solução ao limite de *C5* que lhe está mais próximo. Exemplo:

- Sabendo que o intervalo de valores válidos do critério *C5* são (0.0, 1300.0), consideremos uma solução *a* = -10.0 e uma solução *b* = 1320.0. A penalização atribuída à solução *a* é 0-(-10.0) = 10.0 enquanto que o da solução *b* é (1320.0 – 1300.0) = 20.0

Aquando da comparação de duas soluções podem surgir as seguintes situações (tendo em conta que o critério a optimizar é o *C1*):

1. Se ambas as soluções estão dentro dos limites de *C5*, escolhe-se a que tem melhor valor de *C1*;
2. Se uma solução está dentro dos limites de *C5* e outra está fora, então escolhe-se a solução que está dentro dos limites;
3. Se ambas as soluções estão fora dos limites de *C5*, escolhe-se a que tem menor penalização.

Se comparássemos as duas soluções dadas acima (*a* e *b*) a que "venceria" era a *a* pois é a que tem menor penalização.

## 5.4 Estudos Efectuados para a Escolha dos Operadores do Algoritmo Genético

Analisadas que estão as soluções devolvidas pelo simulador de extrusão, vamos então estudar os algoritmos genéticos com variações dos operadores, para ver quais as combinações que se adequam melhor ao problema.

### 5.4.1. Comparação dos Operadores SBX e PCX

Esta secção apresenta um estudo efectuado ao desempenho dos dois algoritmos genéticos considerados, o Algoritmo 1, geracional que implementa o crossover *Simulated Binary Crossover* (*SBX*), a mutação *Polinomial Mutation* e o esquema de selecção *Tournament Selection*, e o Algoritmo 2, algoritmo genético *steady-state Generalized Generation Gap (G3) Model* com *Parent Centric Recombination* (*PCX*), cujas descrições se encontram no capítulo 3 deste trabalho. Para comparar o desempenho destes dois algoritmos, começou-se por estudar as evoluções dos valores de *fitness* (neste caso apenas foi considerado o critério C1, isto é, a maximização do débito) dos dois algoritmos, representando as suas médias nos gráficos da secção seguinte.

#### Gráficos Ilustrativos da Evolução Efectuada por Cada um dos Algoritmos

Apresentam-se de seguida dois gráficos (Figura 24 e Figura 25) que ilustram a evolução dos valores de fitness dos dois algoritmos. Estes valores obtiveram-se através de dez execuções de cada um dos algoritmos com uma população de 100 indivíduos e evoluindo 15 gerações (1500 avaliações). No primeiro algoritmo a probabilidade de crossover usada foi de 0.5 e a de mutação foi de 1/tamanho do cromossoma, enquanto no segundo as taxas utilizadas (não se

pode falar em taxas de crossover ou de mutação separadas pois o algoritmo envolve os dois num só) no algoritmo foram as adoptadas pelos autores [1].

A linha azul representa a fitness média alcançada em cada momento da evolução (número de avaliações efectuadas).

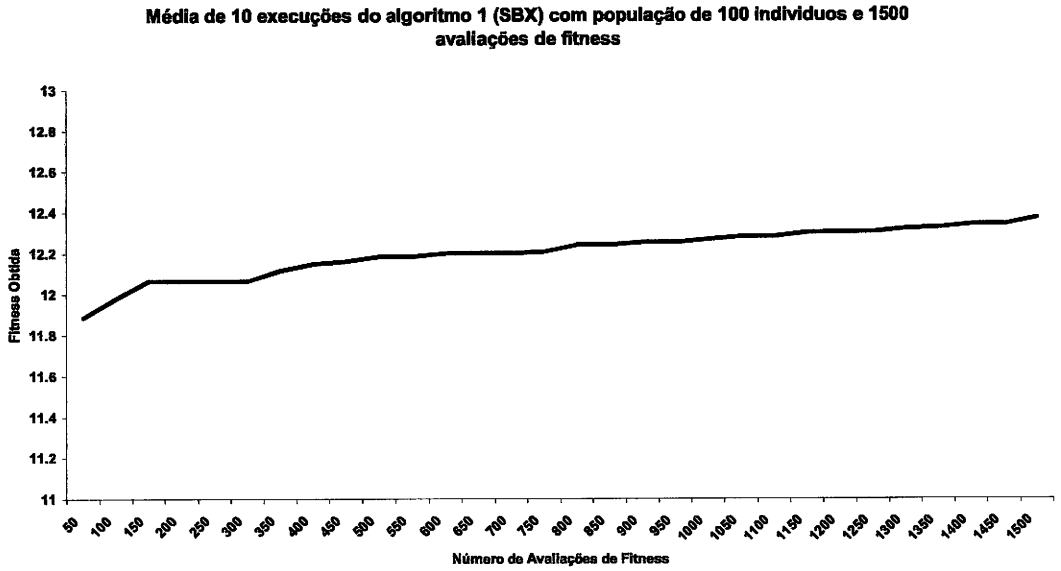



Figura 24 - Evolução média do algoritmo 1

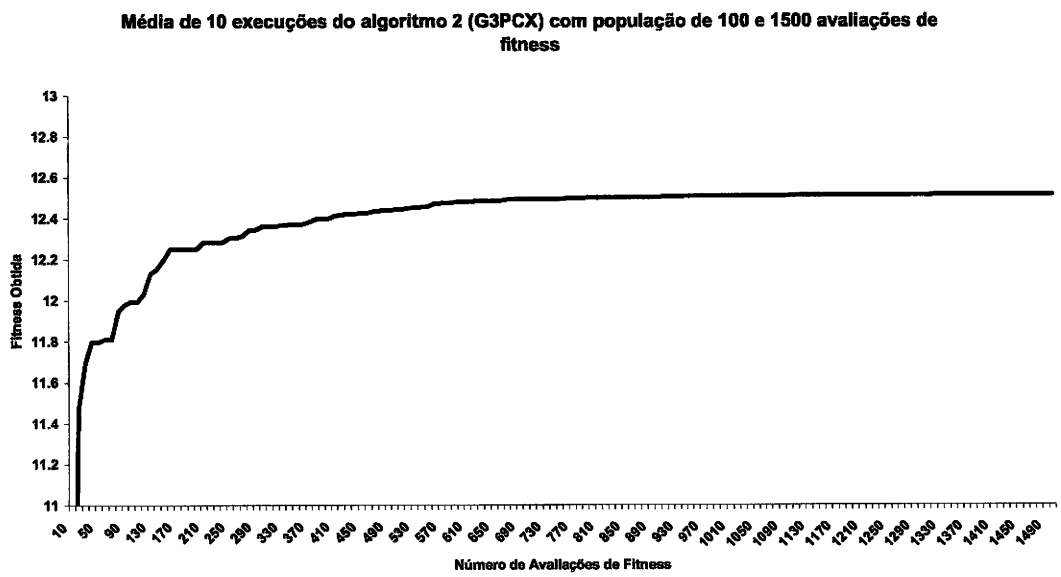



Figura 25 - Evolução média do algoritmo 2

Ao analisarmos os dois gráficos (Figura 25 e Figura 26), reparamos que o primeiro algoritmo atinge o seu máximo (aproximadamente 12.35) já no final da sua evolução (aproximadamente na avaliação 1500, enquanto que o algoritmo 2 aproximadamente na evolução 800, atinge o seu valor máximo (aproximadamente 12.5), exigindo este uma menor computação. À partida, este algoritmo 2 parece comportar-se melhor no problema, mas teremos de o comprovar com testes estatísticos que se encontram na secção seguinte. A técnica estatística de comparação de desempenho dos algoritmos aplicada denomina-se *empirical attainment function* [17]. Esta técnica mede estatisticamente o desempenho de optimizadores estocásticos, englobando quer a qualidade das soluções individuais no espaço dos objectivos, quer a sua dispersão na superfície de *trade-offs*. Note-se que o tempo aqui funciona como um objectivo. Uma descrição mais detalhada desta técnica encontra-se no capítulo 4 deste trabalho.

## Comparação do Desempenho dos Algoritmos

Para a aplicação da *empirical attainment function,* usou-se uma ferramenta desenvolvida por Fonseca et al [19] disponível em http://www.tik.ee.ethz.ch/pisa/. Dos resultados obtidos, surge o gráfico seguinte (considerando 20 execuções de cada algoritmo) (Figura 26):

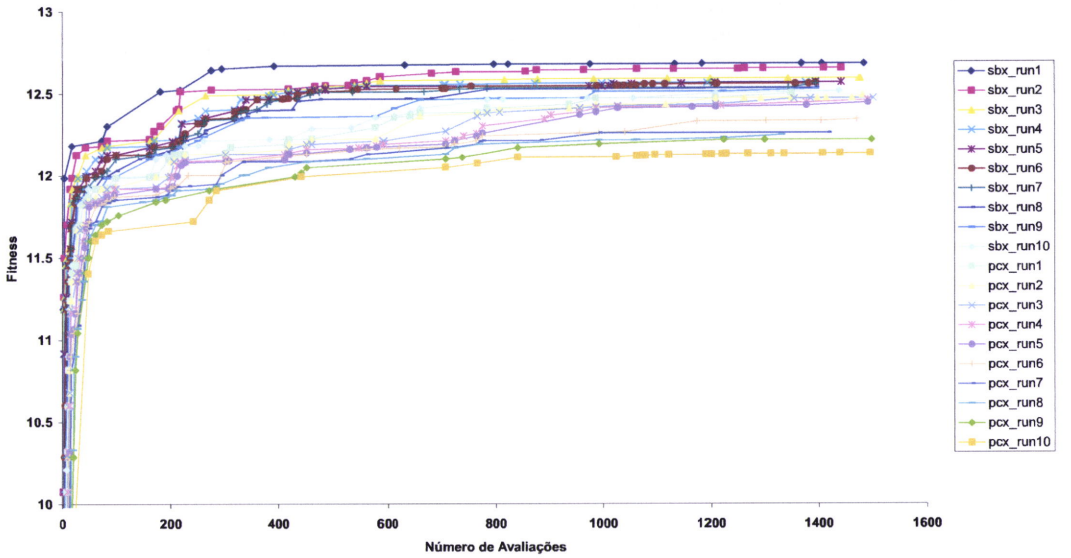



**Figura 26 – Funções de Aproveitamento Empíricas aplicadas a 10 execuções do PCX e a 10 execuções do SBX**

Mais uma vez, pela simples observação do gráfico não conseguimos ter certezas se algum algoritmo é preferível ao outro. Para compreender se algum dos algoritmos apresenta resultados melhores do que os apresentados pelo outro, a biblioteca também disponibiliza código de um teste de hipóteses semelhante ao de *Kolmogorov Smirnov*. As hipóteses a serem testadas são as seguintes:

$$H0 : A_{A\lg 1}(d) = A_{A\lg 2}(d)$$
$$H1 : A_{A\lg 1}(d) \neq A_{A\lg 2}(d) \qquad (16)$$

Significando a hipótese nula que a evolução dos valores devolvidos pelos dois algoritmos é semelhante, isto é, os algoritmos apresentam um desempenho semelhante, enquanto que a hipótese alternativa indica que os algoritmos apresentam desempenhos diferentes. Vejamos, na Tabela 9, os resultados produzidos pelo teste.

**Tabela 9 - Aplicação do Teste de Hipóteses Kolmogorov-Smirnov para comparação dos dois algoritmos em estudo**

```
Using default random seed
Read 40 rows, 2842 columns
Assuming 2 sets of 20 runs each

FIRST-ORDER EAF KS-LIKE TWO—SAMPLE TWO-SIDED TEST
==========================================================

Test statistic = 16/20
               = 0.8

Using 10000 random permutations to estimate null distribution
Please be patient... done
Critical value = 11/20
               = 0.55
p-value = 0

Decision: REJECT the null hypothesis
```

Como o resultado é **rejeitar a hipótese nula**, significa que dois algoritmos não provêm da mesma distribuição apresentando diferenças significativas, podendo nós afirmar com rigor estatístico, que um algoritmo apresenta desempenho diferente do outro. Se a esta conclusão juntarmos a análise aos gráficos das Figuras 24 e 25, concluímos que o algoritmo 2, o algoritmo genético *steady-state* que implementa o *Generalized Generation Gap Model* (*G3*) com *Parent Centric Crossover* (*PCX*), é o que melhor se adequa ao problema. Para além desta razão, este algoritmo é também o mais recente e apresenta um maior conjunto de características possíveis de exploração. Por todos estes importantes aspectos, este Algoritmo 2 foi o escolhido para os estudos que se seguiram neste trabalho.

## 5.4.2. Estudo Teórico do Operador *Parent Centric Crossover* (*PCX*)

À medida que se iam realizando experiências usando o *PCX* detectou-se que a distribuição de valores dos filhos gerados apresentava uma característica bastante interessante que se tentou comprovar. Decidiu-se então aprofundar o estudo dessa distribuição.

De seguida, apresenta-se um gráfico (Figura 27) que representa as frequências cumulativas de pontos gerados pelo *PCX* e de pontos gerados por um gerador de números aleatórios seguindo uma distribuição normal, com a média no melhor pai e com o desvio padrão dos pontos devolvidos pelo *PCX*.

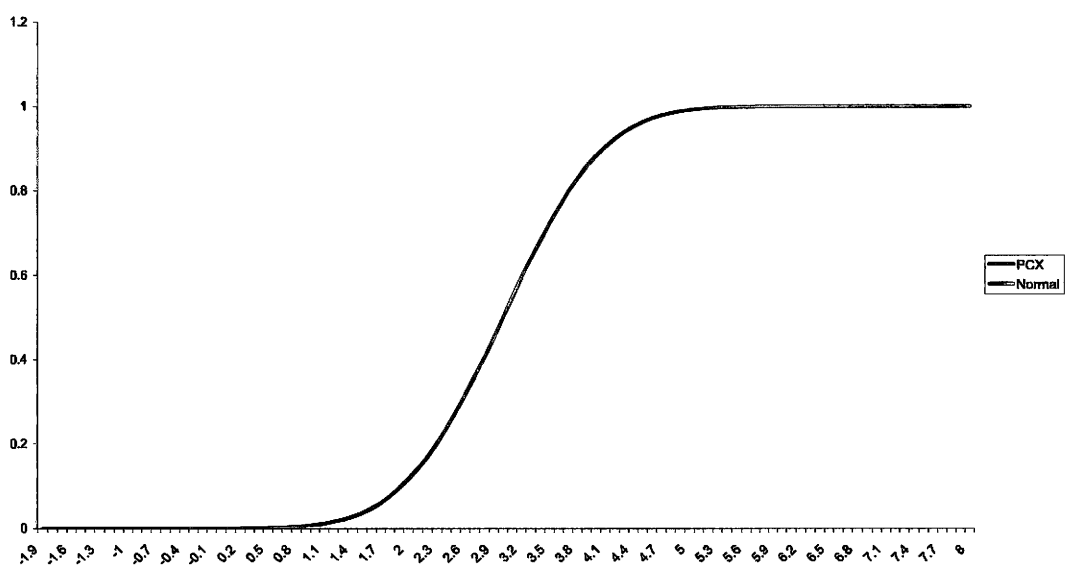



**Figura 27 – Frequências Comulativas de pontos gerados pelo PCX (com os pais [3,3], [-2, 10], [-5, 0]) e por um gerador de números aleatórios segundo distribuição normal**

Como se pode observar pelo gráfico, as duas curvas são praticamente coincidentes o que levou ainda mais a suspeitar da hipótese do *PCX* gerar os filhos segundo uma distribuição normal. Mas, para confirmar estatisticamente esta hipótese, fez-se um teste de *Kolmogorov-Smirnov* comparando a distribuição normal e a distribuição dos valores dos filhos gerados pelo *PCX*, cujos resultados são apresentados na tabela 10. As hipóteses são as seguintes:

$$H0: F_{PCX} = F_{Normal}$$
$$H1: F_{PCX} \neq F_{Normal} \quad (17)$$

**Tabela 10 - Resultados da aplicação do testes estatistico Kolmogorov-Smirnov para verificar se a geração das soluções geradas pelo PCX segue ou não uma distribuição Normal**

**KOLMOGOROV-SMIRNOV GOODNESS-OF-FIT TEST**

**NULL HYPOTHESIS H0:** DISTRIBUTION FITS THE DATA

**ALTERNATE HYPOTHESIS HA:** DISTRIBUTION DOES NOT FIT THE DATA

**DISTRIBUTION:** NORMAL

**NUMBER OF OBSERVATIONS** = 10000

TEST:

KOLMOGOROV-SMIRNOV TEST STATISTIC = 0,0086

| ALPHA LEVEL | CUTOFF | CONCLUSION |
|---|---|---|
| 10% | 0.03858 | ACCEPT H0 |
| 5% | 0.04301 | ACCEPT H0 |
| 1% | 0.05155 | ACCEPT H0 |

| | | | | |
|---|---|---|---|---|
| N | 10000 | 10000 | D(cal)=MAX(ABS(Diff)) | 0,0086 N.S. (P>0.05 |
| N(effective) | 5000 | Approx.P(D>=D(cal)) two-sided | 0,8519301582 | |

**Caution:** the P(D>=D(cal)) is an approximate value, which becomes asymptotically accurate as N becomes large.

Como se pode observar na Tabela 10, quer para um nível de significância de 1, de 5 ou de 10%, o teste **não rejeita a hipótese nula**, o que significa que não há evidência estatística de que as amostras sigam distribuições diferentes. Temos então a demonstração de que o *PCX* gera os filhos segundo uma distribuição normal centrada no melhor pai (ou aquele que é apresentado como tal). Os outros "pais" mostram o estado de convergência da população e usam-se para determinar o desvio padrão da distribuição normal. Esta característica, também observada (com algumas diferenças) em *Particle Swarm Optimization* (*PSO*) [20], confere ao *PCX* a capacidade, também presente em differential evolution [21], de ajustar dinâmicamente durante a execução a exploration/exploitation

por consenso: "*When a particle and its best neighbour have found their previous best solutions in locations in the problem space that are near one another, the amplitude of the particle's oscillation will be smaller; naturally, if their previous successes are very far apart, the oscillations will range over a wider area.*" [22]

Falta então determinar qual a expressão para o cálculo explícito do desvio padrão da distribuição normal gerada pelo operador *PCX*.

## Expressão para Cálculo Explícito do Desvio Padrão da Distribuição Normal Gerada pelo Operador *PCX*

Para determinar a expressão para o cálculo explícito do desvio padrão da distribuição normal gerada pelo operador *PCX*, recorreu-se a duas propriedades do desvio padrão de uma distribuição normal [23]. São elas:

***Propriedade 1:***

Seja *X* uma variável normal $X \sim N(\mu,\sigma)$. A variável normal *Y = aX+b* irá ser

$$Y = aX + b \sim N(a\mu + b, a\sigma), \tag{18}$$

sendo *a* e *b* constantes quaisquer.

***Propriedade 2:***

Se $X \sim N(\mu_x,\sigma_x)$ e $Y \sim N(\mu_Y,\sigma_Y)$ forem independentes, então

$$X + Y \sim N(\mu_X + \mu_Y, \sqrt{\sigma_X^2 + \sigma_Y^2}) \tag{19}$$

A forma de calcular cada *locus* (cada gene do cromossoma) dos filhos gerados através do *PCX* é dada pela seguinte fórmula, dada na descrição do PCX na secção 3.2.3:

$$offs(l) = x_l^{(p)} + d_l^p w_\zeta + (\overline{D} w_\eta)_l - \frac{\left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \times d_l^p}{\left\| \vec{d}^p \right\|^2} \tag{20}$$

De seguida apresentam-se algumas indicações do significado destas variáveis apresentadas na fórmula (20).

- **Centróide**, vector médio dos $\mu$ pais:

$$\vec{g} = \frac{\sum_{i=1}^{\mu} \vec{x}^{(i)}}{\mu} \tag{21}$$

- **Vector Direcção**:

$$\vec{d}^{(p)} = \vec{x}^{(p)} - \vec{g} \tag{22}$$

Onde $\vec{x}^{(p)}$ representa o pai escolhido. No modelo G3 este é o cromossoma da população com melhor fitness.

- "Thereafter, from each of the other ($\mu$-1) parents **perpendicular distances** $D_i$ to the line $\vec{d}^{(p)}$ are computed and their average $\overline{D}$ is found" [1].

$$\overline{D} = \frac{\sum_{i=1, i \neq p}^{\mu} D_i}{\mu - 1} \tag{23}$$

Onde

$$D_i = \left\| \vec{h}^{(i)} \right\| \sqrt{1.0 - \left( \frac{\left\langle \vec{h}^{(i)}, \vec{d}^{(p)} \right\rangle}{\left\| \vec{h}^{(i)} \right\| \left\| \vec{d}^{(p)} \right\|} \right)^2} \tag{24}$$

E

$$\vec{h}^{(i)} = \vec{x}^{(p)} - \vec{x}^{(i)} \tag{25}$$

Um dado locus ($l$) de um filho será então obtido gerando um valor aleatório normalmente distribuído, de média $\vec{x}_l^{(p)}$ e desvio padrão:

$$\sigma_l = \sqrt{(d_l\sigma_\eta)^2 + (\overline{D}\sigma_\zeta)^2 + \left(\frac{-d_l^{(p)}\sqrt{\sum_{j=1}^{n}(d_j^{(p)}\overline{D}\sigma_\eta)^2}}{\left\|\vec{d}^{(p)}\right\|^2}\right)^2} \qquad (26)$$

**Dedução da Fórmula do Desvio Padrão:**

Consideremos separadamente cada uma das parcelas da fórmula (20), referindo-as como *a), b), c) e d)* respectivamente.

*a)*

$$= x_l^{(p)} \qquad (27)$$

*b)*

$$= d_l^p w_\zeta, \qquad (28)$$

onde $d_l^p$ é uma constante e $w_\zeta$ é uma variável normal $w_\zeta \sim N(0,\sigma_\zeta)$ [3]

Então, pela Propriedade 1

$$Y_b = d_l^p w_\zeta \sim N(0, d_l^p \sigma_\zeta) \qquad (29)$$

*c)*

$$= (\overline{D}w_\eta)_l, \qquad (30)$$

Onde $\overline{D}$ é uma constante e $w_\eta$ é uma variável normal $w_\eta \sim N(0,\sigma_\eta)$ [5]

Então, pela Propriedade 1

$$Y_c = (\overline{D}w_\eta)_l \sim N(0, \overline{D}\sigma_\eta) \qquad (31)$$

*d)*

$$= \frac{\left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \times d_l^{(p)}}{\left\| \vec{d}^{(p)} \right\|^2} = \left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \times \frac{d_l^{(p)}}{\left\| \vec{d}^{(p)} \right\|^2} \quad (32)$$

Ora, $(-\frac{d_l^{(p)}}{\left\| \vec{d}^{(p)} \right\|})$ é uma constante. Precisamos então de estudar o produto interno

$\left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle$. (33)

Mas, $\left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle = \left( \overline{D} w_{\eta 1} \times \frac{d_1^{(p)}}{\left\| \vec{d}^{(p)} \right\|} + \ldots + \overline{D} w_{\eta n} \times \frac{d_i^{(p)}}{\left\| \vec{d}^{(p)} \right\|} \right)$, (34)

O que significa que

$\overline{D} \times \frac{d_i^{(p)}}{\left\| \vec{d}^{(p)} \right\|}$, com $i = 1,\ldots,n$ (35)

(sendo $n$ o tamanho do cromossoma) é constante logo, cada uma destas parcelas pode ser descrita como

$$Y_{dj} = \overline{D} w_{\eta j} \times \frac{d_{ji}^{(p)}}{\left\| \vec{d}^{(p)} \right\|} \sim N(0, \overline{D}\sigma_\eta \frac{d_j^{(p)}}{\left\| \vec{d}^{(p)} \right\|}) \quad (36)$$

Portanto, pela Propriedade 2, podemos afirmar que

$$Y_d = \left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \sim N(0, \sqrt{\frac{\sum_{j=0}^{n} (\overline{D}\sigma_\eta d_j^{(p)})^2}{\left\| \vec{d}^{(p)} \right\|^2}}) \quad (37)$$

Para terminar esta parte *d)*, falta apenas considerar a constante que multiplica o produto interno nesta parcela, isto é,

$$(-\frac{d_l^{(p)}}{\left\| \vec{d}^{(p)} \right\|}) \left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle. \quad (38)$$

Então, aplicando a Propriedade 1 resulta então na forma final desta parte:

$$Y_d =$$

$$= \left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \times \left(-\frac{d_l^{(p)}}{\left\|\vec{d}^{(p)}\right\|}\right) \sim N\left(0, -\frac{d_l^{(p)}}{\left\|\vec{d}^{(p)}\right\|} \times \sqrt{\frac{\sum_{j=0}^{n} (\overline{D}\sigma_\eta d_j^{(p)})^2}{\left\|\vec{d}^{(p)}\right\|^2}}\right) = N\left(0, -\frac{d_l^{(p)} \sqrt{\sum_{j=0}^{n} (\overline{D}\sigma_\eta d_j^{(p)})^2}}{\left\|\vec{d}^{(p)}\right\|^2}\right)$$

(39)

Juntando agora (29) + (31) + (39) e aplicando a Propriedade 2 resulta:

$$Y_{b+c+d} =$$

$$= d_l^p w_\zeta + (\overline{D}w_\eta)_l - \frac{\left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \times d_l^p}{\left\|\vec{d}^{\,p}\right\|^2} \sim N\left(0, \sqrt{(d_l^p \sigma_\zeta)^2 + (\overline{D}\sigma_\eta)^2 + \left(-\frac{d_l^{(p)} \sqrt{\sum_{j=1}^{n} (\overline{D}\sigma_\eta d_j^{(p)})^2}}{\left\|\vec{d}^{(p)}\right\|^2}\right)^2}\right)$$

(40)

Falta apenas somar a parte *a)* (a constante (27)) e, aplicando a Propriedade 1, resulta então em

$$Y_{a+b+c+d} =$$

$$= x_l^{(p)} + d_l^p w_\zeta + (\overline{D}w_\eta)_l - \frac{\left\langle \vec{w}_\eta \overline{D}, \vec{d}^{(p)} \right\rangle \times d_l^p}{\left\|\vec{d}^{\,p}\right\|^2} \sim N\left(x_l^{(p)}, \sqrt{(d_l^p \sigma_\zeta)^2 + (\overline{D}\sigma_\eta)^2 + \left(-\frac{d_l^{(p)} \sqrt{\sum_{j=0}^{n} (\overline{D}\sigma_\eta d_j^{(p)})^2}}{\left\|\vec{d}^{(p)}\right\|^2}\right)^2}\right)$$

(41)

Fica então provado que a geração de filhos através do operador PCX segue uma distribuição normal com **média** $x_l^{(p)}$ e com **desvio padrão**

$$\sqrt{(d_l^p \sigma_\zeta)^2 + (\overline{D}\sigma_\eta)^2 + \left(-\frac{d_l^{(p)} \sqrt{\sum_{j=1}^{n} (\overline{D}\sigma_\eta d_j^{(p)})^2}}{\left\|\vec{d}^{(p)}\right\|^2}\right)^2}. \qquad (42)$$

Apresenta-se de seguida um gráfico com as frequências cumulativas de pontos gerados pelo PCX, num caso com o código standard, no outro com a fórmula para o cálculo explícito do desvio padrão.

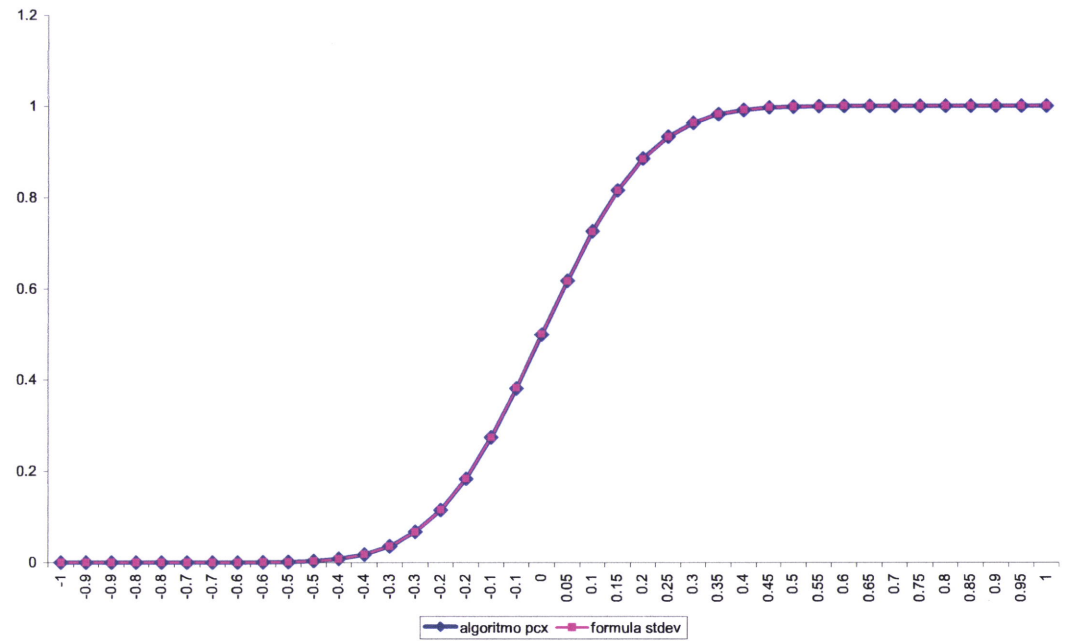



**Figura 28- Frequências Cumulativas de pontos gerados pelo PCX (com os pais [0,2, 3]) num caso com o algoritmo standard e no outro através da fórmula para o cálculo explícito do desvio padrão**

Como se pode verificar pela Figura 28, as linhas encontram-se sobrepostas, o que ilustra a validade da fórmula para o cálculo explícito do desvio padrão da normal que define o funcionamento do operador PCX.

É de salientar o facto deste resultado nunca ter sido apresentado explicitamente em nenhum artigo, fazendo dele um resultado novo.

## 5.5 Conclusão

Tal como referido anteriormente, para optimizar um problema através de um algoritmo genético, é necessário ter primeiro conhecimento acerca do problema em si e do optimizador a usar, de forma a que quer a formulação do problema, e a respectiva codificação, quer os operadores, sejam os mais adequados possível ao problema em estudo. Neste sentido, depois de realizarmos estes estudos, temos já um bom conhecimento para podermos então construir um algoritmo que inclua todos os parâmetros estudados. O algoritmo proposto e respectivas experiências encontram-se descritos no capítulo 6.

# 6. RESULTADOS EXPERIMENTAIS

## 6.1 Introdução

Estudados que estão o problema e os operadores genéticos em causa, neste capítulo vamos então descrever o algoritmo implementado e respectivos resultados obtidos nas diversas experiências realizadas. Para tal, a secção 6.2 apresenta a descrição do algoritmo proposto, algoritmo este que é uma paralelização do algoritmo *steady-state G3* estudado anteriormente, apresenta também uma codificação estruturada hierárquica e realiza uma optimização multiobjectivo. Posto isto, a secção 6.3 descreve um conjunto de resultados obtidos através das experiências realizadas com o objectivo de comprovar que se pode paralelizar o algoritmo genético sem que as soluções devolvidas sofram significativas alterações. Na secção seguinte encontra-se o estudo estatístico efectuado às relações entre as variáveis dos dois parafusos em estudo, com o intuito de ver o que é mais vantajoso, agregar ou separar as variáveis semelhantes.

## 6.2 Algoritmo Proposto *G4PCX*

O algoritmo proposto é essencialmente uma paralelização do *G3* com *PCX* descrito anteriormente. Este algoritmo, denominado por "*G4PCX*", está paralelizado, é hierárquico e multiobjectivo.

A paralelização foi feita através do *PVM* (*Parallel Virtual Machine*) [24], servindo esta para distribuir o processamento por todos os processadores

disponíveis reduzindo o tempo de execução. Esta técnica também serviu como elo de ligação entre o algoritmo genético, desenvolvido na linguagem de programação *Python*, com o simulador do processo de extrusão (desenvolvido pelos parceiros do projecto da universidade do Minho) implementado na linguagem .C. Quanto à implementação deste algoritmo, recorreu-se ao uso de *buffers* quer para os cromossomas, quer para as *fitness*. Ou seja, manteve-se a lógica da geração dos filhos e da sua inserção na população, mas deixou de ser um procedimento rígido (escolha dos pais, seguido da geração dos filhos, seguido da actualização da população), uma vez que o estado dos *buffers* e dos *slaves* do *PVM* que avaliam a *fitness*, passam a ditar quando se lê e quando se actualiza a população.

A forma de funcionamento do *G3* com *PCX* na geração de novos filhos consiste em escolher o melhor indivíduo da população e dois ou mais (dependendo do valor do parâmetro que indica o número de pais usados – habitualmente são 3) escolhidos aleatoriamente. O objectivo da escolha destes últimos é para dar uma noção do estado de convergência da população para assim explicitamente definir o desvio padrão da normal à volta do melhor pai que guiará a geração dos filhos. Quando passamos para multiobjectivo, tudo funciona da mesma forma exceptuando o facto de que deixará de haver um único indivíduo como o melhor da população, mas vários pois estamos a optimizar em relação a duas *weighted sums*. Para o determinar, calcula-se a *frente de Pareto* usando o algoritmo descrito em [25] e, como todos os indivíduos dessa frente são igualmente bons, retira-se então um aleatoriamente para cada chamada ao *PCX*, mantendo os outros pais a serem escolhidos aleatoriamente da restante população.

Como técnica multiobjectivo, adoptou-se a função de agregação (uma ou duas dependendo da experiência a realizar) que se resume a uma soma pesada dos valores dos diferentes objectivos sendo o resultado desta o parâmetro a optimizar.

Apresenta-se de seguida o pseudo código da função principal do algoritmo implementado seguido da respectiva interpretação:

## Pseudo Código do *G4PCX*

```
1     def pvm_checkSlaves(blocking,tid,msg_tag)
2     def pvm_Send_job(tid,buff_chrom,unsent_chroms,slave_jobs)
3     def pvm_Get_and_Send_job(tid,buff_chrom,buff_fit,unsent_chroms,slave_jobs,jobs_done)
4     def fill_unsent_buff(buff_chrom, free_pos, unsent_chroms)
5     def move_jobs_done_to_pop(buff_chrom, buff_fit, jobs_done, free_pos)


6     unsent_chroms = []
7     max_unsent_len = nr_slaves
8     slave_jobs = {}
9     jobs_done = []
10    max_nr_offspring_allowed_waiting = nr_slaves
11    buff_len = max_nr_offspring_allowed_waiting + max_unsent_len + nr_slaves
12    free_pos = [1, 2, ..., buff_len]
13    buff_chrom = numarray.zeros((buff_len,chrom_len), numarray.Float64)
14    buff_fit = numarray.zeros((buff_len,nr_objectives_with_wsums), numarray.Float64)
15    fitness_evals = 0
16    fit_evals_considered_in_pop_updates = 0
```

```
17    Ending = False

18    while fit_evals_considered_in_pop_updates < max_fit_evals:
19          if unsent_chroms == [] and not Ending:

20                fill_unsent_buff()

21          if len(jobs_done) == max_nr_offspring_allowed_waiting:
22                move_jobs_done_to_pop()
23                if fit_evals_considered_in_pop_updates == max_fit_evals:
24                      break


25          if ((len(unsent_chroms) == max_unsent_len or Ending) and
26              len(jobs_done) < nr_offs_needed_for_move):

27                tid, msg_tag = pvm_checkSlaves('Blocking')
28                if msg_tag == MSG_GENOME_RES and tid > 0:
29                      pvm_Get_and_Send_job(tid,buff_chrom,buff_fit,unsent_chroms,slave_jobs,jobs_done)
30                elif msg_tag == MSG_READY:
31                      pvm_Send_job(tid)


32          else:
33                tid, msg_tag = pvm_checkSlaves('Non-Blocking')
34                if msg_tag == MSG_GENOME_RES and tid > 0:
35                      pvm_Get_and_Send_job(tid)
36                elif msg_tag == MSG_READY:
37                      pvm_Send_job(tid)
38                else:
39                      if jobs_done está em piores condiçoes que unsent_chroms:
40                            move_jobs_done_to_pop(buff_chrom, buff_fit, jobs_done, free_pos)
41                      elif not Ending:
42                            fill_unsent_buff()
```

Inicialmente (linhas 1 a 5) apresentam-se os cabeçalhos de algumas funções necessárias para o *loop* principal, nomeadamente, a função *pvm_checkSlaves* que verifica se algum slave enviou mensagem (linha 1), a função *pvm_Send_job* que envia um cromossoma para o slave indicado por *tid* para avaliação (linha 2), a função *pvm_Get_and_Send_job* que recebe de um *slave* um cromossoma avaliado com os respectivos valores de *fitness* e envia um novo cromossoma a esse *slave* para avaliação (linha 3), a função *fill_unsent_buff* que gera novos cromossomas através do *PCX* (ou aleatoriamente durante a fase de inicialização) e coloca-os no *buffer* (linha 4), finalmente a função *move_jobs_done_to_pop* que actualiza a população principal do algoritmo, movendo para lá alguns indivíduos do *buffer* já avaliados, ocorrendo a decisão sobre a inserção de um indivíduo na população segundo a técnica *G3* (linha 5).

De seguida (linhas 6 a 17) encontra-se a inicialização das variáveis. Na linha 6 inicializa-se *unsent_croms* que é uma lista de índices em *buffer* que correspondem por avaliar, isto é, que ainda não foram enviados a nenhum *slave*, seguido na linha 7 pela inicialização de *max_unsent_len* que representa um inteiro que diz quantos cromossomas desejamos manter no máximo em espera por enviar. Na linha 8 está definido o dicionário *slave_jobs* que associa *TasksIDs* (os índices das tarefas) de *slaves* ao índice nos *buffers* do cromossoma e da *fitness* da tarefa atribuída a esse *slave*. É também criada uma lista *jobs_done* de índices (linha 9) nos *buffers* prontos para o *trial* (definida no G3) de inserção na população principal. O inteiro *max_nr_offspring_allowed_waiting* (linha 10) indica qual o tamanho máximo permitido à lista *jobs_done*. Na linha 11 está definida a variável *buff_len* que calcula o tamanho do *buffer*, seguido da lista *free_pos* que indica os índices que estão livres nos buffers (linha 12). Nas linhas 13 e 14 encontram-se então definidos s *arrays* (*buffers*) que guardam os cromossomas e os valores de *fitness*, respectivamente. Finalmente, inicializa-se a variável *fitness_evals* (indica o número de avaliações efectuadas até ao momento) a zero (linha 15), a variável *fit_evals_considered_in_pop_updates* (indica o número de avaliações

consideradas na actualização da população) a zero, e a variável *Ending* a *FALSE* (linha 16).

O *loop* principal encontra-se da linha 18 à 41. Enquanto o número de avaliações consideradas na actualização da população for inferior ao número máximo de avaliações permitido (linha 18) vai verificar o estado dos *buffers* e lida com eles. Se a lista dos cromossomas por avaliar *unsent_chroms* ainda está vazia e se o *loop* ainda não terminou, vai à população principal gerar filhos usando o *PCX*. Se a lista dos cromossomas avaliados estiver cheia (linha 21) faz o *trial* para inserção de indivíduos na população (linha 22) e verifica se já foi atingido o número máximo de avaliações de *fitness* (linha 23) para terminar em caso afirmativo (linha 24). Se não há trabalho para fazer (linha 25) espera por respostas dos slaves (linha 27). Se receber entretanto a mensagem de envio de cromossomas avaliados, guarda-o e envia um outro por avaliar (linha 29), se a mensagem for de que está livre, então envia-lhe um novo cromossoma para avaliação (llinha 31). Se existe algum trabalho para fazer (linha 32), primeiro verifica se algum *slave* está livre para lhe enviar trabalho (linhas 33 à 36), se não está nenhum livre (linha 37), trata do *buffer* em piores condições, tendo em conta que queremos manter o tamanho do *jobs_done* minimizado e o tamanho do *unsent_croms* maximizado ao longo da execução (linhas 38 à 41).

## 6.3 Estudo Efectuado à Paralelização do Algoritmo

Tal como descrito na secção 5.2 do anterior capítulo, o problema da extrusão de polímeros é complexo de optimizar pois envolve um cromossoma com 18 variáveis (ou menos, consoante o número de variáveis partilhadas entre os parafusos) e cinco critérios a optimizar, sendo dois deles a maximizar e três a minimizar. Cada avaliação demora em média 6 segundos (num processador Intel Pentium 4, 3.2 GHz, com 1GB de RAM). Ora, numa optimização que envolva 30 gerações com uma população de 70 indivíduos, são necessárias 2100 avaliações de *fitness*, logo uma execução do algoritmo demoraria 3 horas

e 3 minutos. Se quisermos realizar, por exemplo, 20 execuções do algoritmo, necessitamos de esperar aproximadamente 3 dias, o que é muito tempo. Então a solução seria paralelizar o algoritmo para que o processamento envolvido fosse distribuído pelos oito computadores do *cluster* disponível.

Para testarmos se a paralelização afecta ou não a qualidade das soluções, executámos o algoritmo 21 vezes sem paralelização e outras 21 fazendo a distribuição do processamento pelo *cluster*. A optimização foi realizada tendo em conta um único objectivo, uma *weighted sum* que devolve um valor real que engloba os seis objectivos, usando os pesos mais adequados ao problema, estudados anteriormente em [18].

$$G = \sum_{i=1}^{n} w_i f_i \tag{43}$$

A Tabela 11 mostra os pesos usados.

**Tabela 11 - Pesos utilizados na função soma pesada, para cada objectivo em estudo**

| Objectivo | Peso |
|---|---|
| **Débito (kg/h)** | 0.5 |
| **Comprimento do Parafuso Necessário para a Fusão (m)** | 0.5/4 |
| **Temperatura do Fundido (ºC)** | 0.5/4 |
| **Consumo de Energia (W)** | 0.5/4 |
| **Qualidade da Mistura** | 0.5/4 |

O gráfico seguinte mostra duas linhas, uma correspondente aos valores médios devolvidos pela soma pesada paralelizando o algoritmo, e a outra os valores médios da soma pesada mas não tendo o algoritmo paralelizado.

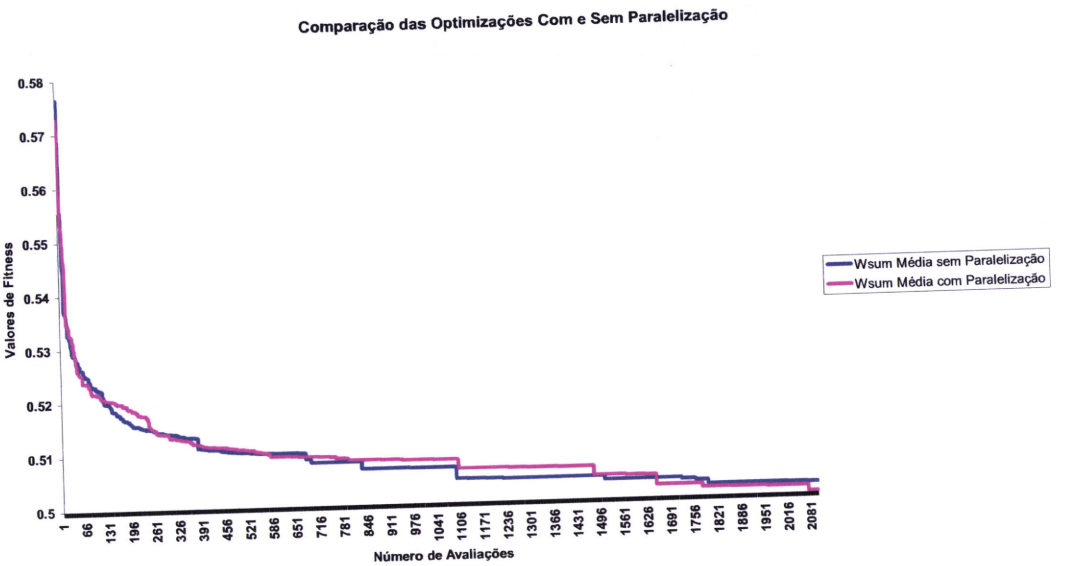



**Figura 29 - Comparação das Optimizações Com e Sem Paralelização**

Por observação do gráfico (Figura 29) podemos constatar que existem umas pequenas diferenças quando se optimiza paralelizando o algoritmo ou não, pois as linhas estão sempre muito próximas ao longo da evolução. Mas para verificar estatisticamente se os resultados apresentam ou não diferenças significativas, aplicámos, mais uma vez, um teste de hipóteses, o teste de *Kolmogorov-Smirnov*, afirmando na hipótese nula que não existem diferenças significativas entre as soluções devolvidas pelo algoritmo paralelizado e pelo não paralelizado, afirmando a outra hipótese o contrário, isto é que existem diferenças.

$$H0: F_{Sem\_Paralelização}(g) = F_{Com\_Paralelização}(g)$$
$$H1: F_{Sem\_Paralelização}(g) \neq F_{Com\_Paralelização}(g) \quad (44)$$

Vejamos os resultados devolvidos pelo teste de *Kolmogorov-Smirnov*, devolvidos pelo programa de estatística R [26].

Tabela 12 - Resultados obtidos pela aplicação de teste estatístico *Kolmogorov-Smirnov* para comparação do desempenho dos algoritmos paralelizado e não paralelizado

**Two-sample Kolmogorov-Smirnov test**

```
> ks.test(sem_paral, com_paral)

        Two-sample Kolmogorov-Smirnov test

data:  sem_paral and com_paral
D = 0.1938, p-value < 2.2e-16
alternative hypothesis: two.sided
```

Como podemos verificar, o valor p é de aproximadamente zero, logo **rejeitamos a hipótese nula**. Podemos afirmar assim que as soluções resultantes do algoritmo paralelizado são diferentes das resultantes do algoritmo não paralelizado, o que pode significar uma interferência positiva ou negativa da paralelização no desempenho do algoritmo.

Para verificar qual é então o efeito da paralelização efectuou-se um outro teste estatístico, desta vez o *Wilcoxon Rank Sum Test* (*Mann-Whitney*) considerando desta vez as seguintes hipóteses:

$$\begin{aligned} H0 &: F_{Sem\_Paralelização}(g) < F_{Com\_Paralelização}(g) \\ H1 &: F_{Sem\_Paralelização}(g) > F_{Com\_Paralelização}(g) \end{aligned} \tag{45}$$

Vejamos então os resultados obtidos através do programa R [27]:

Tabela 13 - Resultados obtidos pela aplicação de teste estatístico *Wilkcoxon rank sum test (Mann-Whitney)* para verificação de qual algoritmo (paralelizado ou não)se adequa melhor ao problema, isto é, qual apresenta melhor desempenho

**Wilcoxon rank sum test with continuity correction**

```
> wilcox.test(sem_paral,com_paral, alternative = "l")

        Wilcoxon rank sum test with continuity correction

data:  sem_paral and com_paral
W = 2055024, p-value = 6.743e-05
alternative hypothesis: true mu is less than 0
```

Como se pode verificar, o valor de p é bastante pequeno o que faz, mais uma vez, com que se **rejeite a hipótese nula**, o que neste caso significa então que os valores obtidos paralelizando o algoritmo são estatisticamente inferiores aos obtidos sem paralelização. Sendo este um problema de minimização, podemos então concluir que paralelizando o algoritmo obtêm-se melhores resultados.

Uma possível justificação para este facto será que, como o funcionamento do **G4** não segue uma estrutura rígida, isto é, de gerar filhos seguido da sua avaliação seguido por sua vez pela inserção na população, mas rege-se sim mediante o estado dos *buffers*, introduzindo-lhe um certo atraso, o que faz com que a geração dos filhos possa ser feita com pais de algumas iterações anteriores. A vantagem deste processo é que assim o algoritmo não converge tão rapidamente o que faz com que tenha uma melhor capacidade de exploração da *fitness landscape*, adaptando-se melhor ao problema.

## 6.4 Estudo das Variáveis Comuns à geometria dos dois tipos de parafusos

Depois de estudada a paralelização do algoritmo começámos então as experiências com o objectivo principal de verificar quais das variáveis idênticas deveriam ou não ser partilhadas por ambos os parafusos. Para tal, implementámos uma codificação hierárquica denominada, por Dasgupta em 1991, de *Structured Genetic Algoritm* (*sGA*) [9] e já descrita no capítulo 3 deste trabalho. Este algoritmo fornece um mecanismo pelo qual se pode dar ao cromossoma a capacidade de codificar simultaneamente mais de um ponto do espaço de estados através da repetição de genes no cromossoma, e adição de variáveis de controlo que determinam quais as secções que estão activas (e determinam a descodificação do cromossoma), e quais as inactivas. Usando esta técnica a selecção operará sobre os valores que estão a ser expressos num determinado momento no cromossoma (parte activa) mas a recombinação vai também operar sobre os valores que não estão a ser expressos, o que faz com que se transporte uma memória de áreas anteriormente exploradas e às quais o cromossoma pode facilmente regressar pela simples troca dos valores das variáveis de controlo. Esta técnica aumentará a capacidade de *exploration* do espaço de estados (exploração não restringida de novos pontos) mas com o custo de, no mesmo número de avaliações, se realizar menos *exploitation* (exploração do conhecimento já adquirido para a geração dos novos pontos a visitar).

Então a questão que se coloca é, mais uma vez, **o que é mais vantajoso: separar ou agregar as variáveis idênticas no problema em estudo? E será que devemos separar ou agregar todas em conjunto? Ou, por outro lado, só algumas é que deverão ser separadas ou ficar agregadas?**

Para tentar responder a esta pergunta identificaram-se todas as 2^6 = 64 possibilidades de agregação ou separação das variáveis. Vejamos dois exemplos destas codificações. O primeiro (Figura 30) representa o caso em que todas as variáveis comuns a ambos os parafusos estão agregadas, isto é,

se por exemplo a variável *L1* tem o valor 280.2, quando o cromossoma for avaliado, quer esteja na altura activo os genes do parafuso convencional (caso em que a variável de controlo, tipo de parafuso, se encontra a 0) quer sejam os genes do parafuso barreira (caso em que a variável de controlo, tipo de parafuso, se encontra a 1), a variável *L1* será sempre 280.2. O que pode acontecer é para o caso de um parafuso pode ser um bom valor enquanto que para o caso do outro parafuso, pode ser um mau valor. Este facto também poderá acontecer para as restantes variáveis comuns a ambos os parafusos. Na outra codificação extrema, isto é, em que todas as variáveis estão separadas (Figura 31) este facto já não acontece, uma vez que cada uma destas variáveis estará duplicada, uma correspondente ao parafuso convencional e outra pertencente ao parafuso barreira *Maillefer.* Assim, o algoritmo durante o processo de optimização já poderá alterar o valor de cada variável tendo em conta apenas o melhor para cada tipo de parafuso. Por exemplo, a variável *L1* poderá ter o valor de 280.2 para o parafuso convencional enquanto que a variável *L1* do parafuso Maillefer pode ter o valor de 320.1.

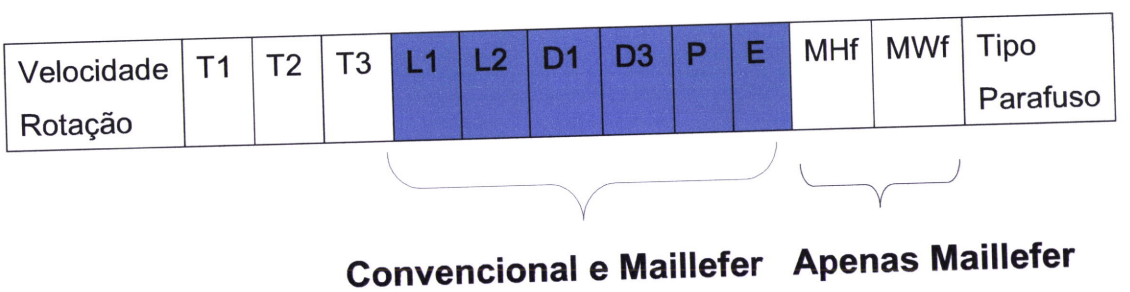



**Figura 30 - Uma possivel codificação do problema agregando, neste caso, todas as variáveis comuns a ambos os parafusos**

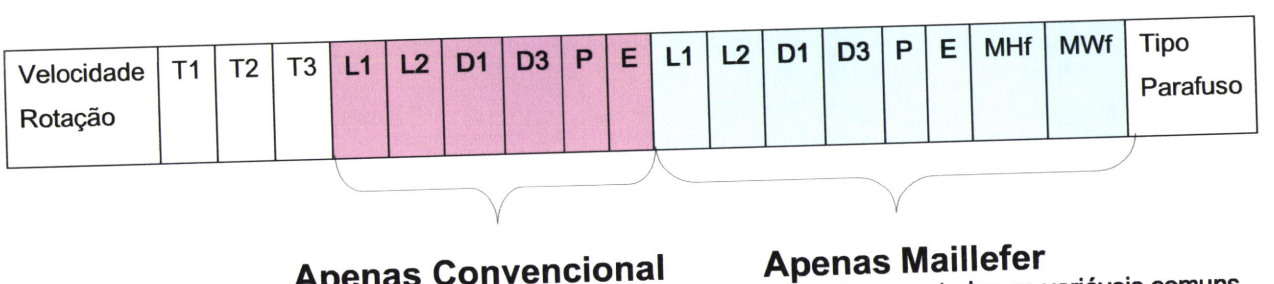



**Figura 31 - Uma outra possível codificação do problema separando, neste caso, todas as variáveis comuns a ambos os parafusos**

Depois de identificadas as 64 codificações, realizaram-se então as experiências com todas elas na tentativa de identificar estatisticamente, qual a codificação que melhor se adequa ao problema.

### 6.4.1. Resultados Obtidos Usando uma Soma Pesada

Depois da definição do algoritmo com todos os parâmetros e técnicas envolvidas realizaram-se as primeiras experiências. Executou-se o algoritmo com uma população de 70 indivíduos e com 2100 avaliações de *fitness* (equivalente a 30 gerações, valor suficiente para obter bons resultados através do PCX), tendo cada uma dessas execuções considerado as $2^6$=64 codificações possíveis. Em todas elas o objectivo a optimizar foi o de minimização da soma pesada (*weighted sum*)

$$G = \sum_{i=1}^{n} w_i f_i \qquad (46)$$

com os pesos já referidos anteriormente, ou seja, w1 = 0.5, w2 = w3 = w4 = w5 = 0.125.

O gráfico que se segue mostra os resultados obtidos ao longo das 2100 avaliações de *fitness*.

Pela simples observação do gráfico (Figura 32) não conseguimos detectar nenhuma codificação que se destaque pela positiva de todas as outras. Portanto aplicaram-se alguns testes estatísticos para estudar estas 64 codificações possíveis.

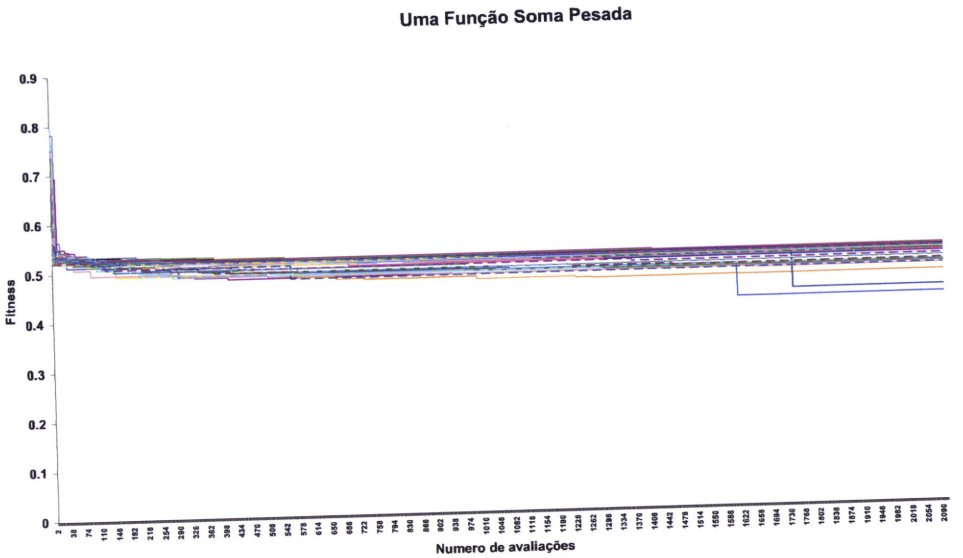



Figura 32 - Gráfico representativo da Evolução de cada uma das 64 Codificações Estudadas

Correu-se o algoritmo mais três vezes e, dos resultados devolvidos por cada execução, formaram-se dois grupos de 32 codificações cada, para o emparelhamento dos dados. Esse emparelhamento foi efectuado da seguinte forma: para cada codificação verifica-se se a variável em estudo está a ser partilhada por ambos os parafusos. (Para uma melhor compreensão, considere-se por exemplo, o estudo da variável *L1*.) Se não estiver partilhada coloca-se na coluna da esquerda, colocando na célula à direita, a codificação idêntica mas em que a variável *L1* está partilhada. Vejamos a primeira linha da tabela 13. À direita da codificação [*D3*, *P*] encontra-se a correspondente [*L1*, *D3*, *P*].

A tabela que se segue mostra os resultados obtidos na primeira execução do algoritmo, apresentados já na forma emparelhada correspondente ao estudo da variável *L1*.

Tabela 14 - Emparelhamento dos resultados obtidos na primeira execução do algoritmo, estando em estudo neste caso, a variável *L1*.

| L1 não Partilhada | L1 Partilhada | Wsum Sem Var Part | Wsum Com Var Part |
|---|---|---|---|
| [D3, P] | [L1, D3, P] | 0.500199445 | 0.52229345 |
| [L2] | [L1, L2] | 0.50695425 | 0.521168279 |
| [L2, E] | [L1, L2, E] | 0.508729208 | 0.524855767 |
| [L2, D3, E] | [L1, L2, D3, E] | 0.521597602 | 0.52161046 |
| [L2, D3] | [L1, L2, D3] | 0.469745103 | 0.521329438 |
| [D1, D3] | [L1, D1, D3] | 0.522540739 | 0.525532204 |
| [D1, D3, E] | [L1, D1, D3, E] | 0.476925353 | 0.517015901 |
| [L2, D1, D3, P] | [L1, L2, D1, D3, P] | 0.510254275 | 0.527658458 |
| [L2, D1, D3, P, E] | [L1, L2, D1, D3, P, E] | 0.522712531 | 0.494294772 |
| [D1, E] | [L1, D1, E] | 0.518559318 | 0.526679559 |
| [D1] | [L1, D1] | 0.525519035 | 0.526415741 |
| [L2, D1, P, E] | [L1, L2, D1, P, E] | 0.519941015 | 0.51965424 |
| [L2, D1, P] | [L1, L2, D1, P] | 0.518779795 | 0.520506389 |
| [D3, P, E] | [L1, D3, P, E] | 0.519609248 | 0.520686439 |
| [P, E] | [L1, P, E] | 0.522962296 | 0.520156158 |
| [P] | [L1, P] | 0.522373001 | 0.519905045 |
| [D3] | [L1, D3] | 0.519581967 | 0.522290648 |
| [D3, E] | [L1, D3, E] | 0.509274613 | 0.47812834 |
| [L2, D1, D3] | [L1, L2, D1, D3] | 0.493221045 | 0.521641353 |
| [L2, D1, D3, E] | [L1, L2, D1, D3, E] | 0.485690284 | 0.520159412 |
| [D1, D3, P] | [L1, D1, D3, P] | 0.526483385 | 0.508040306 |
| [D1, D3, P, E] | [L1, D1, D3, P, E] | 0.502329015 | 0.521310579 |
| [L2, D1, E] | [L1, L2, D1, E] | 0.522808519 | 0.507366815 |
| [L2, D1] | [L1, L2, D1] | 0.511178118 | 0.521202024 |
| [D1, P, E] | [L1, D1, P, E] | 0.521430979 | 0.519450232 |
| [D1, P] | [L1, D1, P] | 0.500284966 | 0.521139658 |
| [L2, P, E] | [L1, L2, P, E] | 0.522377772 | 0.523095846 |
| [L2, P] | [L1, L2, P] | 0.518617951 | 0.516370034 |
| [L2, D3, P, E] | [L1, L2, D3, P, E] | 0.468414108 | 0.528677738 |
| [L2, D3, P] | [L1, L2, D3, P] | 0.520831115 | 0.507914343 |
| [] | [L1] | 0.355877209 | 0.520053866 |
| [E] | [L1, E] | 0.520893953 | 0.52301967 |

O mesmo raciocínio foi considerado para o estudo de todas as outras variáveis, para cada uma restantes execuções.

Depois de aplicar os testes dos sinais, de *wilcoxon* para amostras emparelhadas e de *Friedman*, a cada variável de cada execução sem aqui obter resultados conclusivos. Como individualmente não se conseguiu obter resultados conclusivos, estudaram-se os resultados das quatro execuções em

conjunto, aplicando então o teste de *Friedman* com replicação aos dados, por variável.

As hipóteses são as seguintes:

$$H0 : F_{Gsep}(g) = F_{Gagg}(g)$$
$$H1 : F_{Gsep}(g) \neq F_{Gagg}(g) \quad (47)$$

Os resultados obtidos, desta vez usando o programa de estatísitica *Kyplot* [27], encontram-se na seguinte tabela:

**Tabela 15 - Resultados obtidos pela aplicaçao do teste de Friedman com replicação dos dados das quatro execuções, para as seis variáveis em estudo.**

| Friedman Statistic | Chi^2 | Df | P-value |
|---|---|---|---|
| L1 | 5.5104166 | 1 | 0.01890354 |
| L2 | 0.04166667 | 1 | 0.83825649 |
| D1 | 0.2109375 | 1 | 0.64603359 |
| D3 | 0.00260417 | 1 | 0.95930079 |
| P | 0.2109375 | 1 | 0.64603359 |
| E | 0.2109375 | 1 | 0.64603359 |

Como podemos observar nesta tabela, apenas um valor de p é inferior ao nível de significância (5%). Então, como resposta aos testes de hipóteses, apenas o valor correspondente ao *L1* faz com que se **rejeite a hipótese nula**, podendo nós então concluir que o algoritmo com a variável *L1* agregada tem um melhor desempenho, não se verificando diferenças significativas nos resultados da agregação ou não agregação das outras variáveis. Por esta razão e por uma questão de simplificação, nas experiências seguintes optou-se por agregar sempre todas as variáveis.

### 6.4.2. Resultados Obtidos usando Duas Somas Pesadas

Depois de concluído que optimizando com um objectivo (uma *weighted sum*) o preferível é agregar a variável *L1* sendo para todas as outras indiferente, colocou-se a questão: Porque é que agregando o resultado é melhor? Com um objectivo o melhor parafuso é dum tipo, com dois objectivos pode ser necessário manter os dois tipos de parafusos na população, portanto o resultado pode ser diferente. Então, o passo a seguir é estudar o problema optimizando dois objectivos, isto é duas somas pesadas, de forma a verificar se continua a ser preferível agregar a variável L1 ou não. As funções objectivo são as seguintes:

$$G_1 = f_1 \qquad (48)$$
$$G_2 = \sum_{i=2}^{5} w_i f_i$$

A primeira apenas considera o débito produzido, sendo a segunda uma soma pesada dos restantes critérios.

Uma vez que agora já estamos a envolver uma optimização multiobjectivo, o resultado final já não é um único valor, mas sim uma *fronteira de Pareto*. No entanto, para poder avaliar estatisticamente a qualidade dos resultados obtidos é mais simples ter uma medida única da qualidade da solução (e não um conjunto delas). Para tal efectuou-se o cálculo do indicador de hipervolume das soluções da frente de Pareto produzida. Quanto maior for este valor do hipervolume, melhor será a qualidade da solução.

Nesta experiência executaram-se 20 vezes o algoritmo, 10 vezes com *L1* agregado, 10 vezes com o *L1* separado, mantendo as restantes variáveis

agregadas (pois já vimos anteriormente que a agregação ou não destas variáveis não apresentava diferenças significativas nos resultados).

As hipóteses para os testes serão novamente:

$$H0: F_{Gsep}(g) = F_{Gagg}(g)$$
$$H1: F_{Gsep}(g) \neq F_{Gagg}(g) \quad (49)$$

Nesta experiência já não estamos a trabalhar com dados emparelhados mas sim com repetição de codificações. Então o teste de hipóteses já terá de ser para amostras independentes e não para dados emparelhados. Usou-se então o **teste de Wilcoxon (Mann-Withney)** para dados não emparelhados.

A tabela 16 apresenta os resultados obtidos pela aplicação deste teste:

**Tabela 16 - Resultados obtidos pela aplicação do teste de Wilcoxon (Mann-Withney) aos hipervolumes das soluções obtidas pela optimização de duas funções soma pesada**

**Wilcoxon rank sum test with continuity correction**

```
> wilcox.test(L1_nao_part,L1_part)

        Wilcoxon rank sum test with continuity correction

data:  L1_nao_part and L1_part
W = 13, p-value = 0.01689
alternative hypothesis: true mu is not equal to 0
```

Como podemos observar, o p-value neste caso é de 0.01689, que é menor do que o nível de significância (5%), pelo que se **rejeita a hipótese nula.** Podemos então concluir que se mantém a característica de que **agregando a variável L1 se obtêm melhores resultados do que separando.**

## 6.4.3. Interpretação dos Resultados Obtidos

Como se viu na descrição das últimas experiências, o *L1* deverá ser agregado. Vejamos agora fisicamente o porquê deste resultado.

Comecemos por observar a estrutura da extrusora:

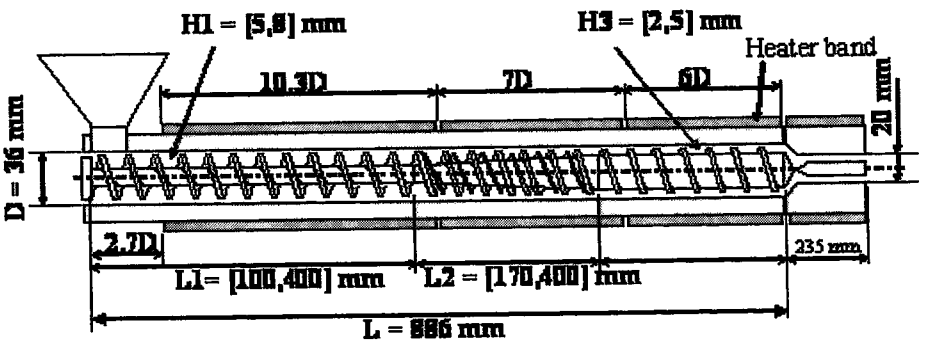



Figura 33 - Exemplo de uma geometria do parafuso convencional [1]

O polímero fundido é transportado da tramonha para a fieira, logo tudo o que acontece na primeira secção influencia as seguintes, mas nem a segunda nem a terceira secção têm uma forte influência na primeira.

Relembremos a estrutura dos diferentes parafusos através da Figura 34:

O facto das diferenças entre os dois tipos de parafusos apenas ocorrerem depois da primeira secção, faz com que a primeira secção seja idêntica nos dois tipos de parafusos. Portanto, tem lógica que a variável que está directamente dependente desta secção, no caso a *L1*, seja a mesma para os dois tipos de parafusos e portanto deva ser agregada.

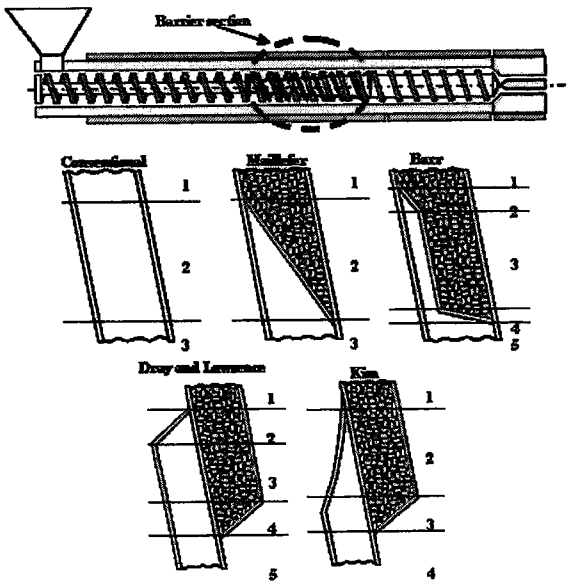


**Figura 34 - Diferentes tipos de parafusos, convencional e os de barreira [1].**

Podemos assim afirmar que os resultados experimentais obtidos neste trabalho têm um significado físico lógico.

# 7. CONCLUSÕES

O objectivo desta tese foi o desenvolvimento de uma metodologia de optimização de equipamento de extrusão de polímeros, que permita a optimização simultânea da geometria dos vários tipos de parafusos existentes, usando para isso algoritmos genéticos.

Por ser um problema com um número razoavelmente elevado de variáveis, logo um grande espaço de pesquisa, e por a função ser avaliada por simulação, decidiu-se optimizar este problema com algoritmos genéticos. Fez-se, para isso, uma pesquisa no domínio dos algoritmos genéticos, pelas melhores variantes, operadores, parametrizações e codificações para este problema. Efectuaram-se estudos ao desempenho de implementações dessas variantes neste problema e concluiu-se que o *G3PCX*, juntamente com um cromossoma de estrutura hierárquica, é o mais adequado.

A aplicação deste algoritmo requer um elevado tempo de processamento pois a avaliação da *fitness* de um indivíduo é feita através do programa de simulação do processo de extrusão (disponibilizado pelos colegas da Universidade do Minho) que demora em média 6 segundos na apresentação do resultado. Por esta razão, desenvolveu-se uma versão paralelizada do algoritmo *G3PCX*, capaz de distribuir o processamento pelas oito máquinas que constituem o *cluster* disponível no *CSI*. Mostrou-se ainda, estatisticamente, que este novo algoritmo (*G4PCX)* apresenta um melhor desempenho para este problema do que o anterior (*G3PCX),* e conjectura-se que apresentará igualmente melhores resultados em problemas de maior complexidade, em virtude de realizar maior *exploration* do espaço de pesquisa.

Realizaram-se estudos teóricos ao operador de recombinação *PCX* (*Parent Centric Crossover*) nos quais se mostrou matematicamente de que forma são produzidas as soluções, dado um conjunto de pais: as variáveis nos cromossomas dos filhos são valores retirados de uma distribuição normal, centrada no melhor pai, e com desvio padrão determinado a partir das restantes pais, os quais fazem uma amostragem do estado de convergência da população.

Quanto à estrutura hierárquica do cromossoma, efectuaram-se estudos no sentido de mostrar estatisticamente qual ou quais das variáveis comuns deveriam, ou não, ser partilhadas pelos vários tipos de parafusos, chegando-se à conclusão de que a partilha da variável *L1* (comprimento da primeira secção do parafuso) provoca realmente um melhor desempenho do algoritmo. As restantes variáveis não mostraram diferenças significativas entre as duas situações.

Apresentado que está um algoritmo genético adequado ao problema, implementado com operadores recentes e aqui intensivamente estudados, com um bom tempo de execução devido à paralelização, e com um cromossoma com uma estrutura hierárquica que se verificou estatisticamente ser a melhor, dou por concluídas todas as tarefas do plano da tese inicialmente entregue.

# BIBLIOGRAFIA


[1] – Deb, K.; Anand, A., and Joshi, D.. *A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization.* KanGAL Report No. 2002003. (April, 2002)

[2] – Felismino, M; Gaspar-Cunha, A; Fonseca, *C.. Application of a Structured Genetic Algorithm to the Design of Screws for Polymer Extrusion.* Proceedings of ECCO XIX, Porto. (2006)

[3] – Gaspar-Cunha, A; Covas, José A.: *A Real-World Test Problem for EMO Algorithms.* EMO 2003, 752-766. Springer

[4] – Gaspar-Cunha, A. *Modelling and Optimisation of Single Screw Extrusion*, Ph. D. Thesis, University of Minho, Guimares, Portugal (2000)

[5] – Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

[6] – Deb, K., Beyer, *H. Self-Adaptive Genetic Algorithms with Simulated Binary Crossover.* Technical Report Nº CI-61/99

[7] – Deb, K.; Goyal, M. *A combined genetic adaptive search (geneAS) for engineering design.* Computer Science and Informatics, 26(4): 30-45. (1996)

[8] – *G3PCX* software (http://www.iitk.ac.in/kangal/soft.htm)

[9] – Dasgupta, D.; McGregor, D. *sGA: A structured genetic algorithm.* Technical Report no. IKBS-8-92, University of Strathclyde.

[10] – Fonseca, C; Fleming, P. *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization.* Genetic Algorithms: Proceedings of the Fifth International Conference (1993)

[11] – Deb, K; Pratap, A.; Agarwal, S.; Meyarivan, T.. *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE Transactions on Evolutionary Computation, 6(2), 2002.

[12] - Zitzler, E., M. Laumanns, and L. Thiele. *Improving the strength pareto evolutionary algorithm.* Technical Report 103, Computer Engineering and Communications Networks Lab (TIK), Swiss Federal Institute of Technology (ETH),Zurich, Gloriastrasse 35, CH-8092. (2001)

[13] – Emmerich, M.; Beume, N.; Naujoks, B.. *An EMO algorithm using the hypervolume Measure as Selection Criterion.* International Conference on Evolutionary Multi-Criterion Optimization. Pp 62-76. (2005)

[14] – Fonseca, C. *Evolutionary Multi-Criterion Optimisation.* Proceedings of the "Advanced School and Workshop on Soft Computing and Complex Systems", Coimbra, June 23 to 27, 2003.

[15] – While, L; et al. *A Faster Algorithm for Calculating Hypervolume.* IEEE Transactions on Evolutionary Computation. Vol. 10. Nº1. (2006)

[16] – da Fonseca, V.; Fonseca, C; Hall, A.. *Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function.* Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization. (2001)

[17] – Fonseca, C.; et al. *A tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers.* Proceedings of the Conference on Evolutionary Multi-Criterion Optimization. (2005)

[18] - Gaspar-Cunha, A.;Covas, J.. *The use of Evolutionary Algorithms to Solve Practical Problems in Polymer Extrusion.* Applications of Multi-Objective Evolutionary Algorithms, C.A. Coello Coello, G.B. Lamont (Eds.), in press, World Scientific. (2004)

[19] – Bleuler, S.; et al. *PISA - A Platform and Programming Language Independent Interface for Search Algorithms.* Proceedings of the Conference on Evolutionary Multi-Criterion Optimization. (2003)

[20] – Kennedy, J. *Bare bones particle swarms.* Proceedings of the IEEE Swarm Intelligence Symposium 2003, Indianapolis, Indiana, USA. (2003)

[21] – Price, Kenneth V. *An introduction to Differential Evolution.* In: David Corne, Marco Dorigo and Fred Glover (editors) (1999). New Ideas in Optimization. McGraw-Hill. pp 79-108. (1999)

[22] – Kennedy, J. *Probability and Dynamics in the Particle Swarm.* Proceedings of the 2004 IEEE Congress on Evolutionary Computation. IEEE Press, Portland, Oregon, USA. (2004)

[23] – DeGroot, M.; Schervish, M.. *Probability and Statistic.* Third Edition. Addison Wesley (2002)

[24] – Geist, A.; Beguelin, A.; Dongarra, J.; Jiang, W.; Manchek, R.; Sunderam, V. *PVM – Parallel Virtual Machine.* MIT Press, Cambridge, MA. (1994)

[25] - Bentley, J.; Clarkson, K.; Levine, D.. *Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls.* Proceedings of the First ACM-SIAM Sympos. on Discrete Algorithms, pp 179-187.(1990)

[26] - R software package (www.r-project.org )

[27] – Kyplot sofware (http://www.kyenslab.com/en/ )

[28] – Zitzler, E., et al. *Performance Assessment of Multiobjective Optimizers: An analysis and Review.* IEEE Transactions on Evolutionary Computation, vol.7, no2, pp. 117-132. (2003)