Universidade Federal de Pernambuco

Graduação em Ciência da Computação

Centro de Informática

2008.2

OBTENDO DIVERSIDADE EM ALGORITMOS EVOLUCIONÁRIOS MULTI-OBJETIVO

TRABALHO DE GRADUAÇÃO

Aluno: Arthur Gonçalves de Carvalho (agc@cin.ufpe.br)

Orientador: Aluizio Fausto Ribeiro Araújo (aluizioa@cin.ufpe.br)

Recife, 25 de novembro de 2008

Universidade Federal de Pernambuco

Graduação em Ciência da Computação

Centro de Informática

2008.2

OBTENDO DIVERSIDADE EM ALGORITMOS EVOLUCIONÁRIOS MULTI-OBJETIVO

TRABALHO DE GRADUAÇÃO

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aluno: Arthur Gonçalves de Carvalho (agc@cin.ufpe.br)

Orientador: Aluizio Fausto Ribeiro Araújo (aluizioa@cin.ufpe.br)

Assinaturas

Este Trabalho de Graduação é resultado dos esforços do aluno Arthur Gonçalves de Carvalho, sob a orientação do professor Aluizio Fausto Ribeiro Araújo, conduzido no Centro de Informática da Universidade Federal de Pernambuco. Todos abaixo estão de acordo com o conteúdo deste documento e os resultados deste Trabalho de Graduação.

_____

Arthur Gonçalves de Carvalho

_____

Aluizio Fausto Ribeiro Araújo

"- Em resumo - disse Tarrou com simplicidade -, o que me interessa é saber como alguém pode tornar-se um santo.

- Mas você não acredita em Deus...

- Justamente. Poder ser um santo sem Deus é o único problema concreto que tenho hoje. "

Albert Camus - A peste

Agradecimentos

Agradeço aos meus pais, Ulisssiano Batista de Carvalho e Maria dos Remédios Costa Gonçalves de Carvalho, e ao meu irmão, Jardel Gonçalves de Carvalho, por servirem como modelo de referência em perseverança e honestidade e pelo apoio incondicional ofertado nesta busca pela Verdade, pelo Justo e pelo Belo.

Agradeço ao meu orientador Aluizio Fausto Ribeiro Araújo pela atenção e paciência e pelos ensinamentos que resultaram neste trabalho.

Agradeço aos meus amigos Renato Marcelino (Carcanaldinho Gaúcho), Icamaan Botelho e Christian Diêgo (Billoquinha) pela amizade oferecida durante estes anos.

Obrigado a todos.

Resumo

O termo otimização, no âmbito da matemática, refere-se ao estudo de problemas em que se buscam soluções ótimas, mínimas ou máximas, para uma dada função através de mudanças sistemáticas nos valores das variáveis. A partir do momento em que desejamos otimizar sistematicamente e simultaneamente variadas funções objetivo (geralmente conflitantes) teremos o processo conhecido como otimização multi-objetivo. Um bom algoritmo de otimização para solucionar os problemas advindos desta área deve possuir duas características básicas: 1) capacidade para determinar múltiplas soluções ótimas de Pareto, isto é, um conjunto de soluções não-dominadas entre si no espaço de busca e 2) habilidade para encontrar uma boa diversidade nos valores das funções objetivo.

Variações dos algoritmos evolucionários constituem a metaheurística mais utilizada para solucionar problemas multi-objetivos, sendo conhecidos na literatura como *Multi-Objective Evolutionary Algorithms* (MOEAs). Usualmente, tais algoritmos empregam unicamente um operador de mutação não-dinâmico para se explorar novas regiões do espaço de busca e, conseqüentemente, obter soluções distintas. Tal procedimento pode acarretar ineficiência tanto na convergência para soluções ótimas de Pareto quanto na obtenção da diversidade.

Dentro do contexto apresentado, este trabalho tem por objetivo propor um operador dinâmico e adaptativo de mutação para MOEAs. Experimentos realizados sobre diferentes problemas mostraram que a abordagem proposta é promissora e pode futuramente ser utilizada para melhorar a performance de algoritmos tanto na velocidade da convergência quanto na obtenção de uma distribuição próxima da uniforme entre as soluções finais.

Palavras chave: Otimização Multi-Objetivo, Algoritmos Genéticos, Controle de Parêmtros, Operador Dinâmico de Mutação.

Abstract

The term optimization, in the field of mathematics, refers to the study of problems in which we are looking for optimal solutions, minimum or maximum, for a given function through systematic changes in the values of variables. When we want to optimize systematically and simultaneously various objective functions (usually conflicting between themselves), we will have the process known as multi-objective optimization. A good algorithm created to solve these sorts of problems must: 1) find multiple Pareto optimal solutions, i.e., a set of non-dominated solutions in the search space and 2) find a good diversity (near to a uniform distribution) among the values of objective functions.

Variations of the evolutionary algorithms, known as Multi-Objective Evolutionary Algorithms (MOEAs), are the metaheuristic most used to solve these kinds of problems. They usually employ only in a non-dynamic mutation operator to exploit new regions of the search space. This approach can result in inefficiency both in convergence to the Pareto optimal solutions and in obtaining diversity.

Within the context presented, this work proposes a dynamic and adaptive mutation operator to be used in MOEAs. A number of experiments considering different problems suggest that the approach proposed here is promising and it can be used in the future to improve the performance of the algorithms in both convergence and diversity.

Keywords: Multi-Objective Optimization, Genetic Algorithms, Parameter Control, Dynamic Mutation Operator.

Sumário

1 – Introdução 12
2 – Otimização Multi-Objetivo 14
2. 1 – Dominância e Pontos Ótimos de Pareto 16
3 – Computação Evolucionária 20
3.1 Inicialização 21
3.2 Representação 21
3.3 Função de Aptidão 22
3.4 População 22
3.5 Mecanismo de Seleção 22
3.6 Operadores Genéticos 22
3.7 Mecanismo de Seleção dos Sobreviventes 23
3.8 Condição de Término 23
3.9 Multi-Objective Evolutionary Algorithms 24
4 – Diversidade em MOEAs 25
4.1 – Sharing Function 25
4.2 - Crowded Comparison Operator 29
4.2.1 – Crowding Distance 30
4.3 - Clustering 31
4.4 – Diversity-Reference Adaptative Control 32
4.5 – Outras Técnicas 36
5 – Operador de Mutação Adaptativo para MOEAs 37
5.1 –Mutação Adaptativa em MOEAs 38
5.1.1 – Mutação Polinomial 38
5.1.2 – Esquema de Adaptação 39
5.2 – NSGA-II 42
5.1.2 – NSGA-II com Operador de Mutação Adaptativo 44
6 – Experimentos 45
6.1 – Problemas 45
6.2 – Métricas 48
6.2.1 – Generational Distance 49
6.2.2 – Spread 49
6.3 – Resultados 50
7 – Conclusões & Trabalhos Futuros 54
Referência Bibliográfica 55

Lista de Figuras

Figura 1 – Ilustração de problemas de otimização 14
Figura 2 – Ilustração da função de Rastrigin e seu mínimo global em (0,0) 15
Figura 3 – Ilustração do conceito de dominância 17
Figura 4 – Ilustrações de fronteiras de Pareto 19
Figura 5 – Fluxo de um algoritmo genético canônico 21
Figura 6 – Ilustração de dificuldades com o conceito de *shared fitness function* 28
Figura 7 – Cálculo da *crowding distance* 30
Figura 8 – Diagrama de blocos do DRAC 33
Figura 9 – Distribuição polinomial utilizada para diferentes valores do parâmetro *n* 39
Figura 10 – Representação do algoritmo NSGA-II 43
Figura 11 – Fronteira de Pareto do problema ZDT1 45
Figura 12 – Fronteira de Pareto do problema ZDT2 46
Figura 13 – Fronteira de Pareto do problema ZDT3 47
Figura 14 – Fronteira de Pareto do problema ZDT4 48
Figura 15 – Fronteira de Pareto do problema ZDT6 48
Figura 16 – Ilustração da métrica *spread* 50

Lista de Tabelas

Tabela 1 – Valores obtidos para o *spread* utilizando 100 gerações 51
Tabela 2 – Valores obtidos para o *generational distance* utilizando 100 gerações 52
Tabela 3 – Valores obtidos para o *spread* utilizando 1000 gerações 52
Tabela 4 – Valores obtidos para o *generational distance* utilizando 1000 gerações..... 52

Lista de Algoritmos

Algoritmo 1 – Cálculo da *shared fitness function* para uma população 20
Algoritmo 2 – Cálculo da *crowding distance* 31
Algoritmo 3 – Esquema de adaptação 40
Algoritmo 4 – Laço principal do NSGA-II 42
Algoritmo 5 – NSGA-II com o operador de mutação proposto 44

1 – Introdução

O termo otimização, no âmbito da matemática, refere-se ao estudo de problemas em que se buscam soluções ótimas, mínimas ou máximas, para uma dada função através de mudanças sistemáticas nos valores das variáveis. A partir do momento em que desejamos otimizar sistematicamente e simultaneamente variadas funções objetivo (geralmente conflitantes) temos o processo conhecido como otimização multi-objetivo (ou otimização com múltiplos objetivos).

Um algoritmo de otimização para solucionar os problemas advindos desta área deve: 1) encontrar múltiplas soluções ótimas de Pareto, isto é, um conjunto de soluções não-dominadas entre si no espaço de busca e 2) estabelecer uma diversidade próxima de uma distribuição uniforme entre as soluções finais.

De acordo com a classificação realizada por Fonseca e Fleming (1993), as chamadas técnicas matemáticas clássicas possuem a limitação de não tratar adequadamente as múltiplas funções existentes, usualmente convertendo-as em uma única função e utilizando técnicas da otimização uni-objetivo para gerar uma solução ótima por execução.

Devido a esta limitação, os Algoritmos Evolucionários propiciaram um meio elegante e eficiente para se obter soluções ótimas, tornando-se, na atualidade, um dos principais métodos para solucionar problemas de otimização multi-objetivo devido ao seu caráter estocástico e ao fato de trabalharem diretamente com uma população de soluções.

O trabalho realizado por Chaiyaratana *et al*. (2007) mostra que o uso do controle de diversidade em algoritmos evolucionários promove o aparecimento de soluções ótimas enquanto mantém uma distribuição próxima da uniforme. Ao se estudar diversidade em algoritmos evolucionários no decorrer do processo evolucionário, podemos enxergá-la sobre a ótica da geração de soluções distintas e/ou da manutenção destas soluções durante gerações consecutivas.

Os modernos MOEAs propuseram diversas técnicas para garantir a manutenção de soluções distintas durante o processo evolucionário. Entretanto, os algoritmos, usualmente, empregam unicamente um operador de mutação não-dinâmico para se gerar soluções diversas, o que pode acarretar ineficiência tanto na convergência quanto na obtenção da diversidade (Eiben e Smith, 1998).

Com isto em mente, este trabalho apresenta um operador de mutação dinâmico e adaptativo para MOEAs. Experimentos realizados sobre diferentes problemas mostraram que a abordagem proposta é promissora e pode futuramente ser utilizada para melhorar a performance de algoritmos tanto na velocidade da convergência quanto na obtenção de uma distribuição próxima da uniforme entre as soluções finais.

Além desta seção introdutória, o restante deste trabalho será apresentado na seguinte ordem:

- Capítulo 2 – Este capítulo apresentará uma introdução aos conceitos básicos da otimização multi-objetivo.
- Capítulo 3 – Este capítulo apresentará uma introdução aos conceitos básicos da Computação Evolucionária.
- Capítulo 4 – Este capítulo terá por objetivo apresentar uma revisão das principais técnicas utilizadas por MOEAs para gerar e manter a diversidade entre soluções.
- Capítulo 5 – Apresentaremos neste capítulo o operador de mutação dinâmico e adaptativo para MOEAs.
- Capítulo 6 – Apresentaremos neste capítulo os experimentos realizados e um estudo comparativo entre os resultados obtidos com o novo operador e com operadores de mutação estáticos.
- Capítulo 7 – Apresentaremos neste capítulo a conclusão do nosso trabalho e uma breve apresentação de possíveis trabalhos futuros.

2 – Otimização Multi-Objetivo

Poderíamos pensar no conceito de otimização em termos de um dado sistema cuja saída ótima desejada (ou sua descrição) é conhecida, bem como seu modelo interno, e objetivamos encontrar a entrada que leve à resposta ótima (Eiben e Smith, 1998). Uma ilustração desta idéia está presente na Figura 1.

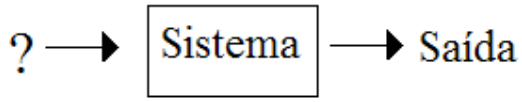

Figura 1 – Ilustração de problemas de otimização.

Um problema de otimização é representado formalmente nos seguintes termos:

> *Dada uma função* $f : A -> R$ *desejamos encontrar um elemento* x_0 *em* A *tal que* $f(x_0) \leq f(x)$ *para todo* $x \in A$ *(ou encontrar* x_0 *tal que* $f(x_0) \geq f(x)$ *no caso de se desejar maximizar o resultado da função)*

sendo o domínio *A*, conhecido por espaço de busca, especificado por um conjunto de restrições de igualdade e desigualdade em que seus elementos, chamados na literatura de soluções candidatas ou pontos, devem satisfazer. A função *f* é chamada função objetivo e a solução que minimize ou maximize seu resultado, de acordo com o problema, é conhecida como solução ótima.

Para exemplificar um problema de otimização utilizaremos uma variação da função de Rastrigin com duas variáveis independentes que é freqüentemente utilizada para testar novos algoritmos (Back, 1992). O problema é definido como:

$$Minimizar\ Ras(\bar{x}) = 20 + x_1^{\,2} + x_2^{\,2} - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

Logo, a função objetivo a ser otimizada, minimizada neste caso, será a $Ras(\bar{x})$, sem restrições para o espaço de busca. Nossa missão é encontrar o valor para o vetor que possibilite o retorno do menor valor possível para a função de

Rastrigin, ou seja, o mínimo global da função. Este valor, bem como o gráfico da função, pode ser visualizado na Figura 2.

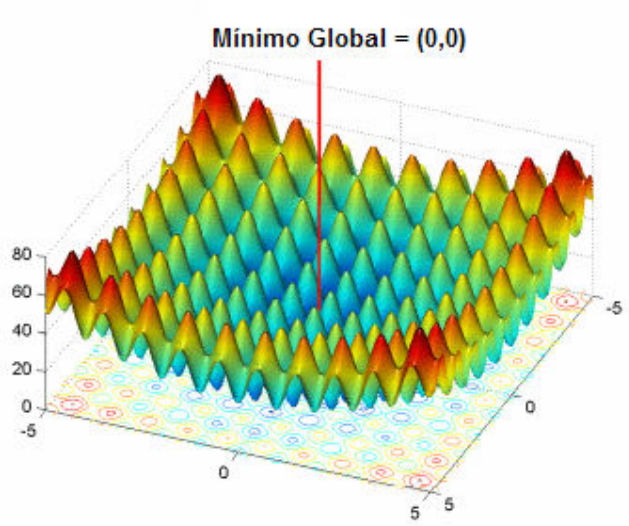

Figura 2. Ilustração da função de Rastrigin e seu mínimo global em (0,0).

A complexidade do processo de otimização pode ser visualizado na função de Rastrigin pela existência de diversos mínimos locais, também conhecidos como vales, que representam soluções não-ótimas e acabam por dificultar o processo de busca pela solução mínima.

A partir do momento que desejamos otimizar sistematicamente e simultaneamente variadas funções objetivo (geralmente conflitantes) teremos o processo conhecido como otimização multi-objetivo. De acordo com Deb (2001), podemos definir formalmente um problema de otimização com múltiplos objetivos nos seguintes termos:

$$\text{Minimizar/Maximizar } f(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), ..., f_M(\bar{x}));$$

$$\text{Relativo às restrições } g_j(\bar{x}) \geq 0, \quad j = 1,2,...,J;$$

$$h_k(\bar{x}) = 0, \quad k = 1,2,...,K;$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1,2...n$$

Ao contrário da otimização com objetivo único, na otimização com objetivos múltiplos existirão M funções objetivo, representadas por $f(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), ..., f_M(\bar{x}))$, que poderão ser individualmente maximizadas ou minimizadas. Associadas com o problema existem J restrições de desigualdade

e *K* restrições de igualdade. A solução $\bar{x}$ que não satisfizer todas as (*K* + *J*) restrições será considerada uma solução não-viável. O conjunto de soluções que satisfazem estas restrições forma o espaço de busca.

A solução $\bar{x}$ será um vetor com *n* variáveis de decisão, representadas por $\bar{x} = (x_1, x_2, ..., x_n)^T$. O último conjunto de restrições, representadas por $x_i^{(L)} \leq x_i \leq x_i^{(U)}$, delimita o valor das variáveis que formam o chamado espaço de decisão. Uma ilustração de um problema de otimização multi-objetivo com funções conflitantes seria a minimização das funções $f_1(x) = x_1$ e $f_2(x) = \frac{1 + x_2}{x_1}$ com as restrições $x_2 + 9x_1 \geq 6$ e $-x_2 + 9x_1 \geq 1$.

Outro conceito importante é o de convexidade. Uma função $f : R^n \rightarrow R$ será convexa se $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$, para $0 \leq \lambda \leq 1$. Em outras palavras, dado um espaço convexo *F* e quaisquer dois pontos *a* e *b* pertencentes a ele, o segmento de reta que unirá estes dois pontos também estará em *F*. Caso contrário o conjunto será conhecido como côncavo (ou não-convexo). Extrapolando este conceito, afirmamos que um problema de otimização com múltiplos objetivos é convexo se todas as funções objetivo e o espaço de busca forem convexos.

2. 1 – Dominância e Pontos Ótimos de Pareto

Após da definição de otimização multi-objetivo é importante citar que esta classe de problemas não possue uma única solução ótima, como no caso da otimização uni-objetivo. Ao contrário, suas soluções são caracterizadas por uma família de alternativas que devem ser consideradas equivalentes na ausência de informações relevantes que venham a discriminá-las. Logo, existirá um compromisso específico do problema (*trade-off*) na escolha da solução final. Com isto, o procedimento ideal ao lidar com problemas possuindo múltiplos objetivos é:

1. Encontrar soluções ótimas e esparsas no espaço de busca;
2. Escolher uma solução utilizando informações de alto nível.

O segundo procedimento faz parte do campo da pesquisa operacional conhecido como *multiple criterion decision-making* (MCDM) e não será coberto neste trabalho. Devido a esta multiplicidade de soluções se faz necessário estabelecer critérios que venham a classificar os indivíduos considerados ótimos. Para ajudar nesta discriminação usualmente é utilizado o conceito de dominância.

Afirmamos que uma solução $\bar{x}$ domina uma segunda solução $\bar{x'}$ se ambos os fatores abaixo são verdadeiros (Carvalho *et al.*, 2008):

1. $f_i(\bar{x}) \leq f_i(\bar{x'}), \quad i = 1, 2, \ldots, n$ (A solução $\bar{x}$ não é pior que $\bar{x'}$ em todos os objetivos);
2. $\exists j \in \{1, 2, \ldots, M\} \mid f_j(\bar{x}) < f_j(\bar{x'})$ (A solução $\bar{x}$ é estritamente melhor que $\bar{x'}$ em pelo menos um objetivo);

Na Figura 3 são ilustradas cinco diferentes soluções no espaço de busca. Supondo que desejamos maximizar a função f_1 e minimizar a função f_2, e utilizando a definição citada acima podemos perceber que, entre outros fatos, a solução 5 domina a solução 1, 2 e 4 mas não pode ser comparada às soluções 3 e 6, visto que aquela é melhor em um dado objetivo e pior em outro.

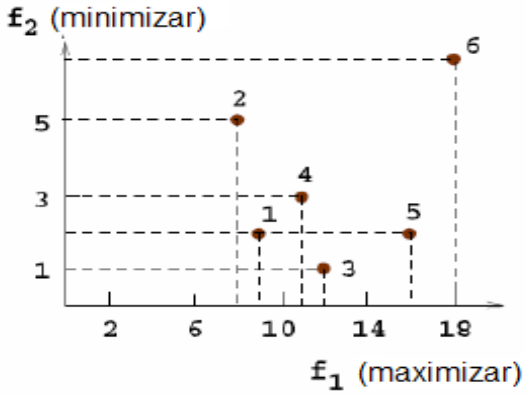

Figura 3. Ilustração do conceito de dominância.

Por motivos didáticos, poderíamos classificar a relação de dominância como transitiva (o que permite criar ordens), mas não simétrica ou reflexiva (o

que não permite criar séries), sendo considerada, então, uma ordem parcial estrita. A comparação entre duas soluções $x^{(1)}$ e $x^{(2)}$, no âmbito desta relação, resultará em um dos três possíveis casos abaixo:

1. $x^{(2)}$ é dominada por $x^{(1)}$;
2. $x^{(1)}$ é dominada por $x^{(2)}$;
3. $x^{(2)}$ não é dominada por $x^{(1)}$ e vice-versa.

Quando ocorre a terceira situação descrita acima afirmamos que $x^{(1)}$ e $x^{(2)}$ são soluções não-dominadas entre si e, neste caso, não seria possível discernir, sem informações adicionais, qual solução seria considerada melhor.

No âmbito da otimização com múltiplos objetivos, afirmar que uma dada solução *a* domina outra solução *b* equivale a afirmar que *a* é preferível a *b*. Deste modo, poderíamos comparar diferentes soluções em um espaço finito, par a par, utilizando este critério e obter um conjunto de soluções que são não-dominadas entre si. É interessante notar que para qualquer ponto *i* fora deste conjunto poderíamos encontrar uma solução pertencente a ele que domine *i*. Logo, o objetivo dos algoritmos para estes problemas passaria a ser a busca pelo conjunto de todas as soluções não-dominadas entre si.

Quando as soluções a serem comparadas formam o espaço de busca, as soluções não-dominadas entre si recebem uma denominação especial chamada soluções ótimas de Pareto. Quando aplicamos as funções objetivo ao conjunto de soluções ótimas de Pareto e plotamos os resultados obtidos, surgem as curvas conhecidas como fronteira de Pareto. Tais curvas estarão em uma região limitante no espaço de busca. A Figura 4 ilustra quatro cenários marcando as devidas fronteiras de Pareto.

Um algoritmo para obter o conjunto de soluções ótimas de Pareto com complexidade $O(N(\log N)^{M-2})$, sendo M a quantidade de funções objetivo e N o tamanho do conjunto de soluções desejado, pode ser encontrado em Kung (1975).

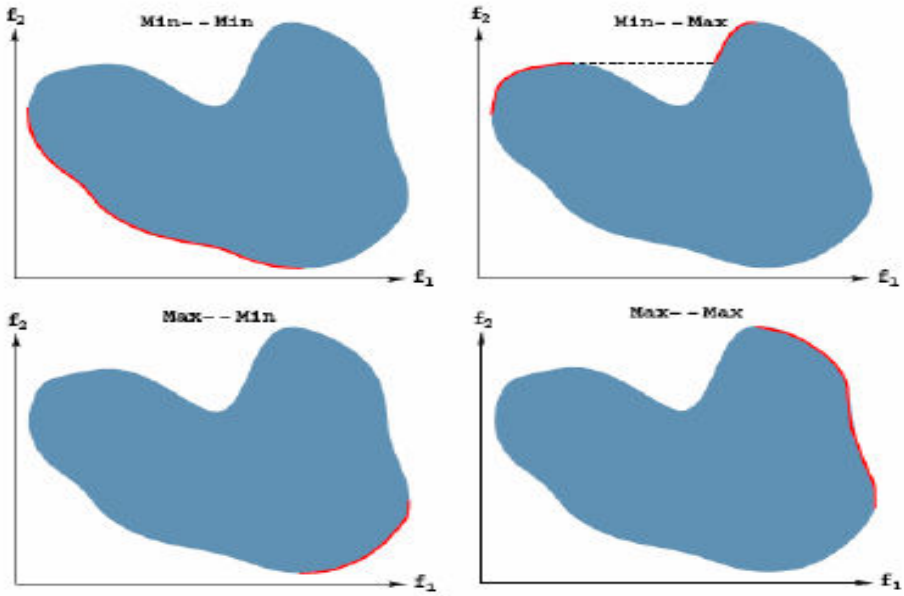

Figura 4. Ilustrações de fronteiras de Pareto. Na ordem esquerda-direita e cima-baixo desejamos, respectivamente, para f_1 e f_2: minimizar/minimizar, minimizar/maximizar, maximizar /minimizar, maximizar / maximizar.

3 – Computação Evolucionária

De acordo com Darwin (1859), a evolução das espécies ocorre a partir de um dado ambiente habitado por uma população de indivíduos que lutam pela sua sobrevivência e reprodução. A aptidão destes indivíduos (determinada pelo ambiente) relaciona o quão bem sucedido eles estarão na obtenção destes objetivos.

Motivada por tal definição, a Computação Evolucionária é uma subárea da inteligência computacional que busca inspirações neste processo biológico evolucionário para solucionar problemas computacionais.

Na computação evolucionária, o processo de solução de problemas ocorre por tentativa e erro (a principal metáfora utilizada do processo biológico), onde teremos uma coleção de soluções candidatas cujas aptidões (o quão bem resolverão o problema figurado pelo ambiente) determinarão a probabilidade que elas terão de serem aceitas e utilizadas como sementes para construírem outras soluções candidatas, ou seja, reproduzirem.

De acordo com Eiben e Smith (1998), os componentes básicos dos algoritmos evolucionários são: inicialização; representação; função de aptidão; população; mecanismo de seleção; operadores genéticos; mecanismo de seleção dos sobreviventes e condição de término.

Baseando-se em diferenças presentes nestes componentes, a Computação Evolucionária pode ser divida em quatro tradicionais variantes: Estratégia de Evolução; Programação Evolucionária; Algoritmos Genéticos e Programação Genética. No trabalho atual nos ateremos aos algoritmos genéticos visto que o mesmo possui maior popularidade e assimilação dentro da comunidade de otimização multi-objetivo. O Fluxo do algoritmo genético canônico é exibido na Figura 5 e nas subseções seguintes seus componentes serão brevemente detalhados.

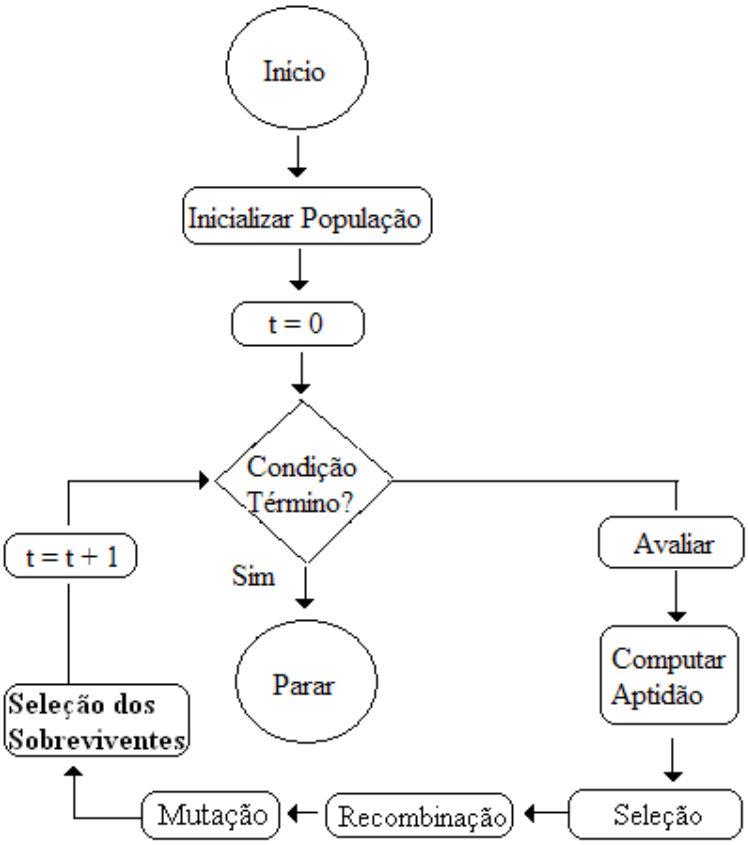

Figura 5. Fluxo de um algoritmo genético canônico.

3.1 - Inicialização

A primeira população do processo evolucionário usualmente é inicializada de forma aleatória. Entretanto, heurísticas específicas ao problema podem ser inseridas com o intuito de incrementar a eficiência na busca pela solução.

3.2 - Representação

A representação faz o mapeamento entre os indivíduos, ou fenótipos (possíveis soluções no contexto do problema original), e suas codificações, também conhecidas como genótipos. É importante notar que a busca no processo evolucionário ocorre no espaço dos genótipos e que a aptidão de um indivíduo é calculada com base no espaço dos fenótipos.

Entre as representações mais comuns estão a representação binária, a representação inteira e a representação por valores reais.

3.3 - Função de Aptidão

No contexto da otimização, a função de aptidão usualmente equivale à(s) função(ões) objetivo. Ela é utilizada para avaliar os fenótipos obtidos da busca realizada no espaço de codificação (genótipos).

3.4 - População

A população, unidade básica da evolução, pode ser vista como um conjunto (podendo existir elementos repetidos) de soluções.

3.5 - Mecanismo de Seleção

O objetivo básico do mecanismo de seleção é a escolha de indivíduos que possuam melhores aptidões para a reprodução. Como citado em Eiben e Smith (1998), está seleção é geralmente probabilística, de modo a permitir que indivíduos com alta aptidão possuam uma maior chance de serem selecionados como pais do que os com menor qualidade, que ainda assim possuirão uma pequena chance, evitando assim possíveis paradas em mínimos locais durante a busca pela solução. Os mecanismos mais comuns são:

- Seleção proporcional à aptidão: A probabilidade de um candidato ser selecionado depende do valor de sua aptidão em relação ao resto da população;
- Seleção baseada em *rank*: A população é ordenada de acordo com as aptidões dos indivíduos e os mais bem posicionados serão escolhidos determinística ou probabilísticamente);
- Seleção baseada em torneios: Indivíduos são agrupados de forma aleatória, geralmente em duplas, e competirão com os demais sobressaindo o candidato com maior aptidão;

3.6 - Operadores Genéticos

O objetivo primário dos operadores genéticos é a criação de novos indivíduos a partir dos existentes, ou seja, a geração de novas soluções

candidatas. De acordo com a quantidade de indivíduos que recebem como parâmetro, os operadores genéticos são divididos em recombinação (operação sobre dois ou mais indivíduos) e mutação (operação sobre um único indivíduo) (Deb, 2001).

Na recombinação, criamos um novo indivíduo a partir da reprodução dos seus pais. Apesar do fato de poderem existir mais do que dois pais, esta situação é pouco explorada na literatura (Eiben e Smith, 1998). Tal operador é estocástico, no sentido em que a escolha das partes de cada pai que será utilizada na combinação é aleatória. Como exemplos de recombinação podemos citar a permutação em ponto único ou em múltiplos pontos na representação binária.

A mutação é um operador estocástico que possibilita novos pontos do espaço de soluções serem incluídos na busca. De acordo com a representação utilizada teremos diferentes modos de se realizar mutações, como a inversão de bits na representação binária ou a adição (ou subtração) de pequenos valores na representação numérica (acréscimo de pequenas pertubações). Tal operador segue a recombinação, isto é, os filhos é que sofrem mutações.

3.7 - Mecanismo de Seleção dos Sobreviventes

O papel deste mecanismo é a seleção dos indivíduos que comporão a próxima geração da população. Para tal, podemos levar em consideração não somente a aptidão, mas diversas outras características tais como a escolha de soluções que favoreçam a diversidade.

3.8 - Condição de Término

Se o problema possuir um resultado ótimo conhecido para a função de aptidão então a condição de término poderá ser definida como o alcance do mesmo. No entanto, devido ao caráter estocástico dos algoritmos evolucionários este valor poderá não ser alcançável em tempo hábil. Com isto, a condição de término comumente utilizada é a disjunção entre o alcance do valor

ótimo e a comparação do número de gerações processadas com um valor definido *a priori*.

3.9 - *Multi-Objective Evolutionary Algorithms*

Os Slgoritmos Evolucionários são utilizados na otimização multi-objetivo basicamente por três razões:

- O conceito de população: Permite atender a necessidade de se encontrar múltiplas soluções;
- Métodos que preservam nichos: Podem ser explorados para encontrar soluções diversas;
- Paralelismo explícito: Possibilita acelerar a busca através de execuções em paralelo.

No geral, as mudanças básicas em relação ao algoritmo canônico exibido anteriormente serão: a modificação na computação da aptidão de um dado indivíduo levando em consideração o conceito de dominância e as múltiplas funções objetivo; ênfase na seleção de soluções não-dominadas para alcançar a convergência para a fronteira de Pareto e a ênfase na seleção de soluções distintas para se obter diversidade na fronteira de Pareto.

4 – Diversidade em MOEAs

Como atestado em Coello et al (2007), existem diversas técnicas para se obter e manter diversidade em MOEAs incluindo o conceito de *fitness sharing/niching*, *crowding distance*, *clustering*, *weight vector*, formas relaxadas de dominância e restrições de combinações. A seguir detalharemos algumas das principais técnicas.

4.1 – *Sharing Function*

O trabalho realizado por Goldberg e Richardson (1987) propõe um modo de manter a diversidade através da degradação do compartilhamento da aptidão de soluções similares. Dentre os algoritmos que utilizam tal conceito podemos citar o *Weight-Based Genetic Algorithm* (WBGA) proposto por Hajela et al. (1993), *Multiple Objective Genetic Algorithm* (MOGA) proposto por Fonseca e Fleming (1993), *Non-Dominated Sorting Genetic Algorithm* (NSGA) proposto por Srinivas e Deb (1994) e o *Niched-Pareto Genetic Algorithm* (NPGA) proposto por Horn et al. (1994).

Goldberg e Richardson (1987) afirmam que dado que desejamos encontrar q soluções ótimas dentro de um problema de otimização multimodal e que possuímos como recurso uma população de tamanho N, de modo que $q << N$ para que se possa trabalhar com uma quantidade adequada de soluções em cada ótimo (nicho), teríamos que cada ótimo i possuiria uma ocupação m_i de modo que:

$$\sum_{i=1}^{q} m_i = N$$

Com isto, caso algum ótimo i possua mais soluções que a sua ocupação esperada, o mesmo se dará em detrimento da quantidade de soluções presentes em outro ótimo. A idéia básica é a de que a aptidão de cada solução presente neste nicho, densamente povoado, seja degradada com o intuito de que o operador de seleção reduza a quantidade de indivíduos oriundos do mesmo em

populações futuras. No cenário oposto, no caso da quantidade de soluções presentes em *i* ser menor do que a sua ocupação esperada, sua aptidão deverá ser elevada para se aumentar o número de indivíduos deste nicho em populações futuras. Logo, teríamos uma nova função de aptidão que levaria em conta não somente a qualidade de um indivíduo, mas a ocupação do espaço ao seu redor e, com isto, manteríamos soluções diversas, pertencentes a diferentes nichos, durante o processo evolucionário. Em termos matemáticos a nova função de aptidão, para cada indivíduo *j* pertencente ao nicho *i*, seria:

$$f_j^{'} = \frac{f_j}{m_i}$$

Esta função hipotética, conhecida como *shared fitness function*, possibilitaria que todos os pontos de um ótimo (locais e globais) possuíssem aptidões similares. Se um operador de seleção proporcional fosse aplicado utilizando estes novos valores de aptidão, cada ótimo obteria sua quantidade esperada de indivíduos.

Entretanto, a identificação precisa das soluções pertencentes a cada nicho requer um avançado conhecimento em relação ao problema, o que usualmente não se possui. No trabalho descrito em Goldberg e Richardson (1987), os autores sugerem uma estratégia adaptativa para se estimar o número de soluções pertencentes a cada nicho utilizando a seguinte *sharing function*:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{share}} \right)^{\alpha}, & se\, d_{ij} \le \sigma_{share} \\ 0, & caso\, contrário \end{cases}$$

A imagem da função está contida no intervalo $[0,1]$, o parâmetro d_{ij} é igual à distância euclidiana entre as soluções *i* e *j* calculada no espaço de decisão, o valor de α informa como será o comportamento da função no intervalo $[0,1]$ (por exemplo, se $\alpha = 1$ a função reduzirá linearmente de 1 para 0) e o parâmetro σ_{share} informa o raio do nicho, em outras palavras, a extensão da influência que uma solução *i* terá na aptidão de uma segunda solução *j*.

No cenário em que $d_{ij} > \sigma_{share}$ e, conseqüentemente, $Sh(d_{ij}) = 0$ podemos afirmar que as soluções *i* e *j* não estão no mesmo nicho. Por outro lado, se *i* = *j* teremos que $Sh(d_{ij}) = 1$ e a influência será máxima. Em casos fora dos extremos, quanto menor a distância entre as duas soluções maior será a influência que uma exercerá sobre a aptidão da outra. Podemos afirmar que $Sh(d_{ij})$ é uma função para estimação de densidade baseada na soma das distâncias entre soluções. Fixando um dado indivíduo *i* e utilizando está *sharing function* em todos os outros *j* membros da população (inclusive a si mesmo) poderemos estimar o número de soluções presentes no seu nicho do seguinte modo:

$$nc_i = \sum_{j=1}^{N} Sh(d_{ij})$$

Por fim, considerando f_i com a aptidão original da solução, computaríamos a nova função de aptidão (a *shared fitness function*) para o indivíduo *i* como:

$$f_i' = \frac{f_i}{nc_i}$$

Com isto, todo ótimo que estiver massivamente representado possuirá suas soluções com aptidões degradadas por possuírem um alto nc_i. É visível que uma ênfase será dada para ótimos pouco povoados. O algoritmo em pseudocódigo, para computar a *shared fitness function* das soluções presentes em uma população é descrito a seguir:

1.	para cada indivíduo *i*
2.	para cada indivíduo *j*
3.	calcular $nc_i = nc_i + \sum_{j=1}^{N} Sh(d_{ij})$
4.	calcular $f_i' = \frac{f_i}{nc_i}$

Algoritmo 1: Cálculo da *shared fitness function* para uma população.

Como pode ser visto do seu conceito, a definição correta do parâmetro σ_{share} é crítica para se obter um tamanho de nicho adequado. O trabalho realizado por Deb e Goldberg (1989) sugere diferentes modos para se obter tal valor. Entretanto, os autores adicionam a necessidade de se informar *a priori* uma quantidade q de ótimos a serem calculados. Se q for maior que a quantidade real de ótimos, a nova função de aptidão criará mais nichos do que o necessário e, conseqüentemente, soluções subótimas. Por outro lado, se q for menor do que a quantidade real de ótimos do problema, nem todos os ótimos poderão ser encontrados.

O trabalho realizado por Fonseca e Fleming (1993) sugere uma forma adaptativa para o cálculo do σ_{share}. Entretanto, somando-se à computação extra realizada existe o fato de não haver uma comprovação real das vantagens de se utilizar tal abordagem.

Outra dificuldade encontrada foi levantada por Oei et al. (1991) quando argumentam que o conceito de *shared fitness function* não funcionaria bem com operadores de seleção baseados em rankings ou torneios. Somente operadores baseados em proporcionalidade poderiam manter subpopulações com um número proporcional de soluções.

Por fim, existe um problema inerente na forma de computação da nova aptidão. Com o intuito de ilustrat tal erro, plotamos na Figura 6 a função $f(x) = |sex(\pi x)|$ no intervalo $0 \le x \le 2$.

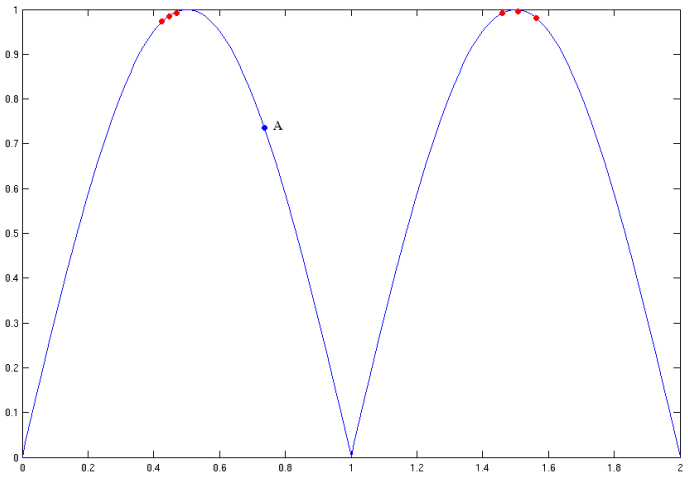

Figura 6. Ilustração de dificuldades com o conceito de *shared fitness function*.

Supondo que desejamos maximizar a função, teríamos dois ótimos globais ($x = 0.5$ e $x = 1.5$). Na Figura 6 existe um total de sete soluções e, claramente, as soluções em vermelho deveriam possuir maior aptidão em relação à solução em azul, visto que estão mais próximas dos ótimos globais. Entretanto, quando utilizamos uma *shared fitness function* para o cálculo da aptidão tal fato não ocorrerá devido ao superpovoamento perto dos nichos. Logo, uma solução com uma aptidão original não muito alta obteria neste contexto a maior *shared fitness* dentre todas.

Uma possível forma de amenizar este problema poderia ser obtida com uma restrição no operador de recombinação (Deb, 2001). Através da limitação de possíveis combinações de soluções para as que estejam contidas no mesmo nicho, reduziríamos a probabilidade de criar indivíduos distantes dos ótimos.

Deb (1989) propôs um esquema para restrição de recombinação utilizando um parâmetro σ_{mating}. Para se obter um par para determinada solução *i*, é calculada a distância euclidiana entre *i* e uma segunda solução escolhida aleatoriamente. Se o resultado for menor do que σ_{mating} então as soluções podem ser combinadas. Caso contrário, uma terceira solução será escolhida aleatoriamente e uma nova verificação deverá ser realizada.

4.2 - *Crowded Comparison Operator*

Este operador de seleção foi proposto por Deb et al. (2002) na formulação do algoritmo *Non-Dominated Sorting Genetic Algorithm* II (NSGA-II). O mesmo assume que para cada solução *i* teremos:

1. Uma posição dentro de um *rank* formado por grupos de soluções não-dominadas entre si, que dominam soluções de posições superiores e são dominadas por soluções com posições inferiores;
2. Um valor d_i que mede o espaço ao redor de *i* que não está ocupado por outras soluções (densidade).

Com estas premissas em mente, afirmamos que uma solução *i* venceria uma solução *j* (seria selecionada), de acordo com este operador, se uma das duas condições seguintes for verdade:

1. A solução *i* possui um *rank* melhor (menor) que a solução *j*;
2. Se ambas possuem o mesmo ranking, mas $d_i > d_j$.

A primeira condição constitui uma forma explícita de elitismo ao se dar preferência a soluções não-dominadas. A segunda condição garante que entre tais soluções a que possuir um maior espaço vizinho não ocupado por outra solução será a preferida. Com isto, o algoritmo privilegia a diversidade entre soluções através da escolha daquela presente em um espaço menos povoado. A subseção a seguir descreve em maiores detalhes a função que estima a densidade ao redor de uma dada solução.

4.2.1 – *Crowding Distance*

Para estimar a densidade de soluções ao redor de um ponto *i*, os autores propuseram que se use a distância média das soluções de ambos os lados de *i* para todas as funções objetivo. Esta quantidade, chamada de *crowding distance* ($i_{distance}$), serviria como uma estimativa do tamanho do maior cubóide englobando *i* formado pelo uso dos seus vizinhos mais próximos como vértices e sem a presença de outras soluções. Na Figura 7, considerando os pontos dados como a fronteira de Pareto, o conceito de *crowding distance* para a solução *i* seria a soma de $r = f(i+1) - f(i-1)$ com $s = g(i-1) - g(i+1)$.

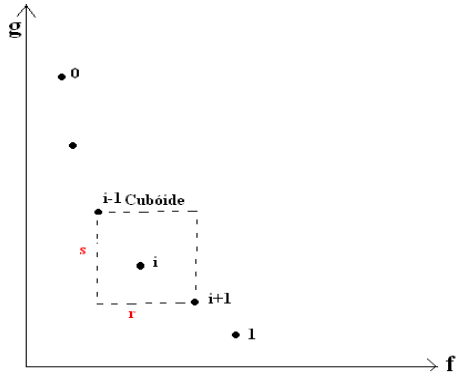

Figura 7. Cálculo da *crowding distance*.

O algoritmo, em pseudocódigo, utilizado para calcular a *crowding distance* de cada solução pertencente a um conjunto *L* é exibido abaixo.

1.	$\ell = \|L\|$
2.	para cada indivíduo *i*
3.	$L[i]_{distance} = 0$
4.	para cada objetivo m
5.	$L = sort(L, m)$
6.	$L[1]_{distance} = L[l]_{distance} = \infty$
7.	para i = 2 até i = (L-1)
8.	$L[i]_{distance} = L[i]_{distance} + (L[i+1].m - L[i-1].m)$

Algoritmo 2: Cálculo da *crowding distance*.

Inicialmente, ordenamos cada função objetivo *m* (passo 5) de acordo com o seu valor para cada indivíduo da população. Após isto (passo 6), colocamos como infinito os valores de pontos extremos (maior e menor valor para a função objetivo). Por fim (passo 7 e 8), adicionamos à $L[i_{distance}]$ de um indivíduo a diferença entre seus dois vizinhos para a função em questão. Logo, esta métrica estima o tamanho do maior cubóide que engloba somente a solução *i* e tem como vértices as soluções vizinhas de *i*. É importante notar que $L[i+1].m$ (passo 8) denota o valor da solução objetivo *m* para a solução *i + 1* normalizada. Com isto, teríamos que:

$$L[i]_{distance} = L[i]_{distance} + \frac{(L[i+1] - L[i-1])}{L^{\max} - L^{\min}}$$

Dentre as vantagens de se usar o conceito de *crowding distance* está o fato da mesma não necessitar de parâmetros extras como o σ_{share}.

4.3 - *Clustering*

O algoritmo *Strength Pareto Evolutionary Algorithm* (SPEA), proposto por Zitzler e Thiele (1998), utiliza o conceito de *clustering* para gerar diversidade entre soluções.

O algoritmo utiliza uma população externa *P* de tamanho *N* para armazenar as soluções não-dominadas encontradas até o presente momento. Quando, em uma dada época, o número de indivíduos ultrapassa *N*, o algoritmo descarta soluções pertencentes a um mesmo grupo, ou seja, soluções que estão próximas no espaço.

Para fins ilustrativos, supondo que *P* em um dado momento possua *N'* soluções (*N'* > *N*) e que *C* seja o conjunto de grupos formados. Inicialmente, o algoritmo de agrupamento considerará cada solução presente em *P* como um grupo, sendo $C = \{C_1, C_2, ..., C_{N'}\}$. Enquanto $|C| > N$, o algoritmo calculará a distância entre cada par de grupos (C_1, C_2) utilizando a distância euclidiana ponderada mostrada abaixo:

$$d_{(C_1, C_2)} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i, j)$$

onde $d(i, j)$ é a distância euclidiana computada pelos autores no espaço de busca (objetivo).

Dado o cálculo destas distâncias, os grupos que possuírem a menor distância entre si serão agrupados em um grupo maior. A partir do momento que $|C| = N$, cada grupo deverá escolher um único representante e descartar as demais soluções. Esta escolha se dá através do cálculo da menor distância das soluções ao centróide do grupo, ou seja, o indivíduo com a menor distância média das outras soluções será o escolhido.

Dentre algumas desvantagens presentes nesta técnica encontram-se o fato dela não priorizar soluções em extremos, o que ajudaria na criação de diversidade, e no elevado custo computacional de $O(MN^3)$, apesar de Deb (2001) informar a existência de possíveis mudanças que a tornariam $O(MN^2)$ ou mesmo linear em $O(N)$.

4.4 – *Diversity-Reference Adaptative Control*

O trabalho realizado por Gouvea e Araujo (2007) propõe um algoritmo evolucionário que buscará soluções ótimas ao mesmo tempo em que um

controlador almeja controlar a diversidade na população, de forma adaptativa, tomando por base uma dada função de diversidade e penalizando aptidões de determinados indivíduos. Apesar de proposto para a otimização de funções uni-objetivo, esta técnica pode ser estendida para a otimização multi-objetivo.

A estratégia de controle captura o processo evolucionário como um problema da teoria do controle, cuja saída do processo (variável controlada) será um valor representativo da diversidade da população e a entrada do processo (sinal de controle) será um ou mais parâmetros genéticos a serem ajustados.

Para controlar a diversidade na população, o método proposto, chamado *diversity-reference adaptative control* (DRAC), se baseia no *model-reference adaptative system* (MRAS) (Aström e Wittenmark, 1995; Landau, 1981; Wagg, 2003) e cria um mecanismo de reprodução e seleção de sobreviventes que ajusta a aptidão dos indivíduos considerando dois fatores: A distância da solução para o ponto médio da população e o erro entre a diversidade atual e a desejada. Com isto, o método pode evitar uma convergência prematura e permite a exploração de diversas áreas no espaço de soluções. A Figura 8 ilustra o diagrama de blocos do DRAC.

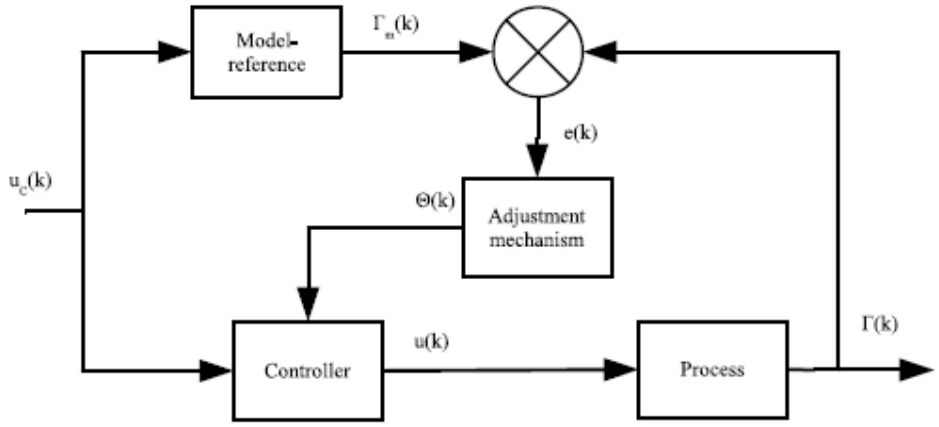

Figura 8. Diagrama de blocos do DRAC.

O bloco *process* representa o processo evolucionário, possui como entrada um sinal advindo do bloco *controller* e retorna o valor da diversidade atual. Os

trabalhos presentes em Bui *et al*. (2005) e Meiyi *et al*. (2004) apresentam modos de se estimar a diversidade de uma população em problemas com um único objetivo. No trabalho original, os autores propuseram a distância euclidiana entre os indivíduos e o ponto médio da população.

O bloco *Model-reference* representa a característica da população com relação à diversidade, ou seja, enquanto o processo evolucionário tenta determinar a solução ótima, o sistema de controle tenta regular a diversidade da população para o nível desejado especificado por este componente. O mesmo pode ser expresso matematicamente nos seguintes temos:

$$\Gamma_m(k+1) = \Gamma_m(k) + u_c \cdot (\Gamma_m(k) - \Gamma(0)) \cdot \exp(-u_c k)$$

onde $\Gamma(0)$ é a diversidade inicial da população, $\Gamma_m(k)$ é a diversidade final quando $k \to \infty$ e u_c é o sinal inicial que define a intensidade em que a população deve variar. O caráter exponencial da equação acima advém de características naturais dos algoritmos genéticos que possuem a diversidade reduzida de forma aproximadamente exponencial durante o processo evolucionário (Prügrl-Bennett, 1997; Uyar e Harmanci, 2005). Com isto, o modelo expressa uma diversidade cujo caráter exponencial pode ser ajustado em termos de convergência e velocidade através de ajustes nos valores do sinal inicial u_c e da diversidade desejada $\Gamma_m(k)$.

Em posse do valor desejado e do obtido, a diferença entre os dois, o erro $e(k)$, será repassado para o componente *Adjustment mechanism*. O mesmo será responsável por atualizar o sinal de controle θ e repassá-lo para o componente *Controller*. Este ajuste se dará de acordo com a regra MIT (Aström e Wittenmark, 1995). Esta regra define que, dado um erro $e(k)$, uma função custo $J(\theta)$ é calculada, onde θ é o controle de parâmetro que será adaptado. Uma função custo típica seria:

$$J(\theta) = \frac{1}{2} e^2$$

Se o objetivo em mente é minimizar o custo relacionado com o erro, o parâmetro θ pode ser mudado de acordo com o gradiente negativo de *J*. Logo:

$$\frac{d\theta}{dt} = -n\frac{\partial J}{\partial \theta} = -n\frac{\partial e}{\partial \theta}$$

onde $n \in [0,1]$ é a magnitude do ajuste de θ e a derivada parcial $\frac{\partial e}{\partial \theta}$ estabelece como o erro será influenciado pelo parâmetro θ. Com isto a equação utilizada para o ajuste do sinal de controle, na sua forma discreta, será:

$$\theta(k+1) = \theta(k) - n' e(k)$$

onde $n' = n\frac{\partial e}{\partial \theta}$. Recebendo como entrado o sinal de controle θ e o sinal inicial u_c, o componente *Controller* retornará o sinal $u(k)$ que segue a seguinte equação:

$$u(k) = \theta(k) \cdot u_c(k)$$

Como parte do processo evolucionário os autores criaram um mecanismo de seleção, utilizado tanto para a escolha dos pais quanto para a escolha dos sobreviventes, executado em três estágios:

1. Seleção do indivíduo com maior aptidão (elitismo) para a próxima geração;
2. Seleção de αN indivíduos baseada em operadores padrões de seleção;
3. Seleção de $(1-\alpha)\cdot(N-1)$ indivíduos baseando-se nas distâncias dos mesmos ao ponto médio da população, com o intuito de manter diversidade.

No passo 3, a aptidão de um indivíduo será calculada a partir da distância ponderada pelo sinal $u(k)$ de acordo com a função:

$$f_i'(d_i, u_k) = 1 - \exp(-u_k \cdot d_i)$$

É interessante notar que quanto menor a distância entre uma solução *i* e o ponto médio da população menor será a aptidão de *i*. Assim, um indivíduo

distante e que contribua para manter a diversidade é mais provável de ser selecionado. O sinal $u(k) > 0$ regula a influência da distância sobre o mecanismo de seleção. Quanto menor o seu valor, maior será a influência da distância na seleção e maior será a diversidade obtida para a próxima geração.

É possível visualizar de todas as equações e funções exibidas acima que quanto maior o erro (diferença entre saída deseja e saída obtida) menor será o sinal de controle θ e, conseqüentemente, menor será o sinal $u(k)$, o que implicará em uma maior influência da distância na aptidão dos indivíduos e uma maior pressão para se escolher soluções distantes e distintas.

4.5 – Outras Técnicas

Uma abordagem mencionada por Coello et al (2007) é a *weight vector*. Nesta, um vetor com pesos é utilizado no espaço objetivo com o intuito de influenciar a busca para diferentes direções e, assim, mover as soluções para longe dos seus vizinhos, criando diversidade na fronteira de Pareto.

Laumanns et al (2002) propôs uma forma relaxada de dominância chamada *e-dominance*. Em linhas gerais, definimos um conjunto de caixas com tamanhos iguais no espaço objetivo. Somente uma solução não-dominada será mantida por caixa. No caso de haver mais de uma solução não-dominada na caixa, certas funções objetivo serão priorizadas em detrimento de outras. Então, o vértice da caixa que possuir os maiores valores para as funções priorizadas será considerado o ponto ótimo e a solução escolhida para ser mantida na caixa será a que tiver menor distância para este vértice.

5 – Operador de Mutação Adaptativo para MOEAs

A execução de um algoritmo evolucionário é um processo intrinsecamente dinâmico e, com isto, o uso de parâmetros constantes entra em contradição com este espírito (Eiben e Smith, 1998). Empiricamente, é visível o fato de que diferentes valores de parâmetros possam ser mais adequados para diferentes estágios do processo evolutivo. Logo, um controle sobre os parâmetros se faz necessário.

De acordo com os trabalhos realizados por Eiben et al. (1999), Hinterding et al. (1997) e Smith e Fogarty (1997), o controle de parâmetros em algoritmos genéticos pode ser classificado como determinístico, adaptativo e auto-adaptativo. As técnicas determinísticas modificam parâmetros sem a existência de realimentações do processo evolucionário. As técnicas adaptativas capturam realimentações do processo evolucionário para determinar a direção e/ou magnitude das mudanças de cada parâmetro. As técnicas auto-adaptativas utilizam a idéia de evolução para os parâmetros do algoritmo genético, ou seja, os mesmos são codificados em cromossomos e passam por processos de seleção, recombinação e mutação. A idéia é a de que os melhores valores codificados destes parâmetros levarão às melhores soluções.

Como explicitado anteriormente, os modernos MOEAs propuseram diversas técnicas para garantir a manutenção de soluções distintas durante o processo evolucionário. Entretanto, os algoritmos, usualmente, confiam unicamente em um operador de mutação não-dinâmico para se gerar soluções diversas.

Com isto em mente, apresentamos neste capítulo um operador de mutação adaptativo que utiliza informações oriundas do conceito de *crowding distance* (vide seção 4.2.1) para determinar a magnitude da mutação a ser realiza sobre os indivíduos. Almejamos com isto obter uma diversidade final próxima da ideal entre as soluções (distribuição uniforme na fronteira de Pareto) e aumentar a eficiência e eficácia da convergência do algoritmo utilizado.

5.1 – Mutação Adaptativa em MOEAs

Como ponto de partida para se criar uma mutação adaptativa para MOEAs, utilizamos o operador proposto por Deb e Goyal (1996) em conjunto com um esquema de adaptação de parâmetros que influenciará a intensidade da mutação. Nas subseções seguintes, elucidamos o operador original e a proposta de adaptação de seus parâmetros.

5.1.1 – Mutação Polinomial

Deb e Goyal (1996) propuseram que, para uma variável contínua, a pertubação do seu valor atual ocorrerá com o auxílio de uma distribuição polinomial, com a média no valor da variável e a variância em função de um parâmetro *n*. O valor mutado é calculado com uma distribuição de probabilidade, dependente de um fator de pertubação δ válido no intervalo $(-1,1)$, de acordo com a função:

$$P(\delta) = 0.5(n+1)(1-|\delta|)^n$$

A Figura 9 ilustra tal distribuição para diferentes valores do parâmetro *n*. Como pode ser inferido da figura e da função apresentada, quanto maior o valor de *n* menor será a variância da distribuição e, conseqüentemente, menor será a mutação (pertubação) do valor original. No cenário oposto, quanto menor o valor de *n* maior será a variância e, conseqüentemente, maior será a diferença do novo valor obtido para o valor original.

Para se obter uma mutação de uma dada variável, inicialmente geramos um valor aleatório $u \in (0,1)$ e, em seguida, a equação abaixo é utilizada para calcular o valor de pertubação δ de acordo com *u*:

$$\delta = \begin{cases} (2u)^{\frac{1}{n+1}} - 1, & se\ u < 0.5 \\ 1 - [2(1-u)]^{\frac{1}{n+1}}, & se\ u \geq 0.5 \end{cases}$$

Por fim, o valor resultante *c* será calculado de acordo com a equação:

$$c = p + \delta * \Delta\max$$

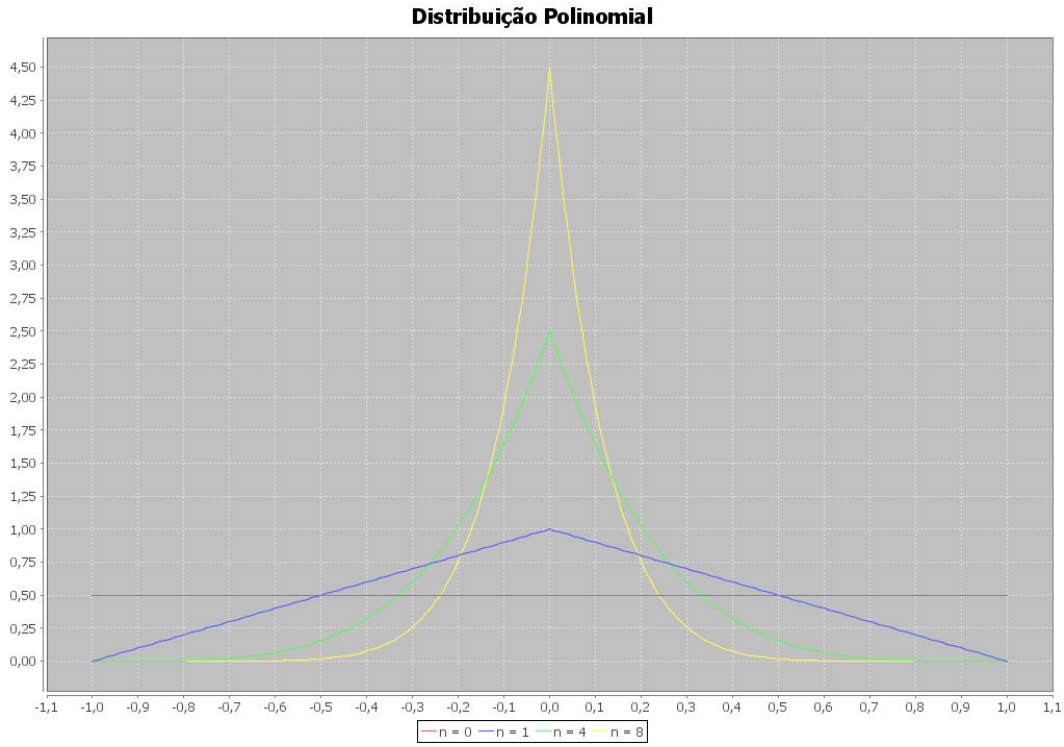

Figura 9. Distribuição polinomial utilizada para diferentes valores do parâmetro *n*.

onde p é o valor atual da variável e Δ_{max} é o valor máximo permitido para a mutação (a diferença entre os limiares para a variável). Logo, o esquema de adaptação deverá atualizar o parâmetro n para aumentar ou diminuir a variância permitida na mutação e, com isto, afetar a diversidade e a convergência do algoritmo. É importante notar que utilizamos as equações acima para variáveis contínuas. Entretanto, a extensão do operador apresentado para variáveis discretas seria uma tarefa trivial.

5.1.2 – Esquema de Adaptação

O esquema de adaptação proposto se baseia no conceito de *crowding distance* apresentado na subseção 4.2.1. A idéia explorada é a de que no início do processo evolucionário, usualmente, as soluções estão dispersas no espaço de busca e distantes da fronteira de Pareto. Neste cenário, as soluções possuem variações bruscas para os valores da *crowding distance*. No decorrer do processo, as soluções se aproximariam de ótimos globais e, com o auxílio de alguma

técnica de manutenção de soluções diversas, tais variações se tornariam mais suaves, com os valores dos cubóides mais próximos entre si.

Seria de extrema importância para a velocidade e corretude da convergência para a fronteira de Pareto e para a obtenção de soluções distintas aplicar uma mutação forte nos estágios caóticos e iniciais do processo evolucionário e uma mutação suave nas gerações finais, evitando a destruição de soluções ótimas previamente criadas e tentando uma leve aproximação para a fronteira real de Pareto.

Aproveitando-se do fato de que o uso de técnicas para prover a manutenção de soluções distintas acarreta em valores para a *crowding distance* mais uniformes no decorrer do processo evolucionário, a idéia do esquema de adaptação aqui apresentado é explorar tal característica para guiar mudanças no parâmetro do operador de mutação e, assim, criar uma geração eficiente de soluções distintas. O algoritmo do esquema pode ser visto abaixo:

1.	$\text{crowding_distance}(P_i)$
2.	$\Delta = Max(P_{i_dis\tan ce}) - Min(P_{i-dis\tan ce})$
3.	$n = Sigm(t)/\Delta$

Algoritmo 3: Esquema de adaptação.

Inicialmente, deve ser calculado a *crowding distance* para cada solução presente na população atual. No segundo passo calculamos a diferença entre o maior (diferente de infinito) e o menor valor da *crowding distance* encontrado. Por fim, nosso parâmetro será calculado através da divisão entre a função sigmóide logística aplicada na geração corrente e o valor Δ previamente calculado.

A inspiração para o uso de tal função sigmóide advém do fato de que ela modela a curva em *S* do crescimento de um algum conjunto *P*, onde *P* poderia ser visto como uma população. O estágio inicial do crescimento é aproximadamente exponencial. Então, a saturação começa e o crescimento se

torna lento. Na maturidade (estágio final), o crescimento cessa e o tamanho da população se mantém constante. (Kingsland, 1995).

Por esta razão tal função se encaixaria perfeitamente na nossa proposta, visto que gostaríamos de aplicar uma mutação forte nos estágios iniciais do processo evolucionário e uma mutação leve e constante nos estágios finais. Utilizando a função sigmóide teríamos um valor extremamente baixo nas primeiras gerações (o que acarretaria em um n menor e, conseqüentemente, maior variância na distribuição do operador de mutação) que cresceria exponencialmente e se estabilizaria em *1* (um), mantendo a influência constante da quantidade de gerações passadas na mutação atual. A função sigmóide logística utilizada pode ser expressa matematicamente de acordo com a seguinte função:

$$Sigm(t) = \frac{1}{1 + e^{\frac{-7t}{T}}}$$

onde T é o número máximo de gerações que o algoritmo executará (condição de término), e t é a época atual. A constante *-7* foi determinada heuristicamente de modo que e^{-7} produz uma boa aproximação do valor 0 (zero).

Logo, tal esquema garante que quanto maior a diferença entre os valores extremos para a *crowding distance* de uma população, menor será o valor de n (parâmetro da mutação) e, conseqüentemente, maior a mutação realizada. O esquema de adaptação também leva em consideração a geração atual do processo evolucionário. Quanto maior a quantidade de épocas passadas maior será o numerador de n e, conseqüentemente, menor será a mutação a ser realizada, visto que se espera que a técnica para manter soluções diversas esteja diminuindo a diferença entre os valores extremos da *crowding distance*.

Na seção a seguir, explicaremos brevemente o algoritmo NSGA-II e como o operador de mutação proposto poderia ser inserido nele. A idéia explorada e a de que possamos unir a capacidade de manutenção do NSGA-II com uma geração de soluções eficiente através do operador de mutação adaptativo proposto.

5.2 – NSGA-II

O algoritmo NSGA-II (Deb et al., 2002) foi proposto primariamente devido à críticas recebidas pelo seu antecessor com respeito ao alto custo computacional envolvido (complexidade de $O(mN^3)$ para m funções de aptidão e N indivíduos na população), a necessidade de se especificar o parâmetro σ_{share} (vide seção 4.1) e à falta de um elitismo explícito.

Para atacar a primeira crítica, os autores propuseram uma forma de classificação de soluções em níveis, de acordo com a quantidade de indivíduos que a dominam, com a complexidade de $O(mN^2)$ computações. Como tal procedimento é considerado o mais custoso de todo o algoritmo, a sua complexidade total foi reduzida para $O(mN^2)$.

Uma segunda contribuição do algoritmo (para atacar a segunda e a terceira crítica) foi a proposta da técnica *crowding distance*, livre de parâmetros, para se estimar a densidade de soluções ao redor de um dado ponto (vide seção 4.2.1). Com o uso de tal técnica, os autores propuseram um operador chamado *Crowded Comparison* (vide seção 4.2) para guiar o processo de seleção de soluções e obter maior convergência mantendo a diversidade. O laço principal do algoritmo pode ser visto no Algoritmo 4.

1.	$R_t = P_t \cup Q_t$
2.	$F = sort(R_t)$
3.	enquanto $\lvert P_{t+1} \rvert < N$
4.	$crowding_dis\tan ce(F_i)$
5.	$P_{t+1} = P_{t+1} \cup F_i$
6.	$sort(P_{t+1}, \geq_n)$
7.	$P_{t+1} = P_{t+1}[0:N]$
8.	$Q_{t+1} = nova_população(P_{t+1})$
9.	$t = t + 1$

Algoritmo 4: Laço principal do NSGA-II.

Inicialmente, o algoritmo realiza a união entre a população formada pelos pais e a população formada pelos filhos para se criar uma população com tamanho *2N* (linha 1). A partir desta junção, o algoritmo classificará cada solução em diferentes fronteiras de Pareto de acordo com o nível de dominância utilizando o novo método previamente citado (linha 2).

Os próximos passos serão para o preenchimento da nova população de pais (linhas 3 a 5). Enquanto o tamanho da população não for maior do que *N*, ou seja, enquanto não extrapolar o tamanho permitido, o algoritmo calculará a *crowding distance* de cada solução presente na fronteira *i* e as adicionará na nova população. Quando o tamanho da população for maior do que *N* (tamanho máximo permitido) o algoritmo deverá remover algumas soluções.

Tal procedimento é efetuado pelas linhas 6 e 7 do algoritmo. Na primeira, a população de pais (cujo tamanho é maior do que *N*) é ordenada com o uso do operador *Crowded Comparison* (vide seção 4.1). Logo, uma ênfase será dada a soluções que estejam em melhores (menores) fronteiras de Pareto seguido de soluções que possuam uma maior vizinhança sem indivíduos (menor densidade). Após a ordenação, serão selecionadas as primeiras *N* soluções para formar a nova população de pais.

Por fim, a população de filhos é formada a partir do uso de uma cópia da população de pais e dos operadores usuais dos algoritmos genéticos (seleção, recombinação e mutação). Um elitismo explícito é garantido pelo fato de se guardar na população formada pelos pais os melhores indivíduos e pelo fato de que os mesmos serão utilizados para gerar novas soluções. Uma ilustração do NSGA-II pode ser visto na Figura 10.

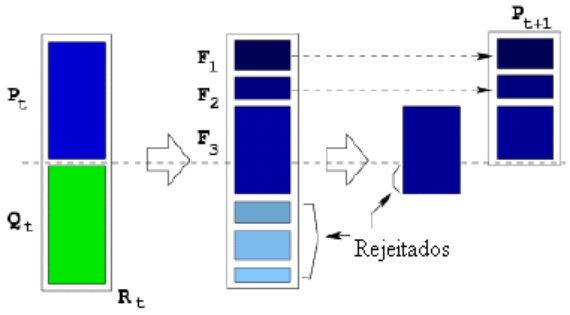

Figura 10. Representação do algoritmo NSGA-II.

5.1.2 – NSGA-II com Operador de Mutação Adaptativo

Como pode ser observado da subseção anterior, o algoritmo NSGA-II utiliza o conceito de *crowding distance* dentro do seu *crowded comparator*. Logo, a inclusão do operador de mutação aqui proposto é direta e intuitiva. O Algoritmo 5 mostra como é calculado o parâmetro da mutação e utilizado o novo operador de mutação dentro do NSGA-II.

1.	$R_t = P_t \cup Q_t$
2.	$F = sort(R_t)$
3.	Enquanto $\lvert P_{t+1} \rvert < N$
4.	$crowding_dis\tan ce(F_i)$
5.	$P_{t+1} = P_{t+1} \cup F_i$
6.	$sort(P_{t+1}, \geq_n)$
7.	$P_{t+1} = P_{t+1}[0:N]$
8.	$\Delta = Max(P_{i_dis\tan ce}) - Min(P_{i-dis\tan ce})$
9.	$n = Sigm(t)/\Delta$
10.	$Q_{t+1} = nova_população(P_{t+1})$
11.	Reprodução(Q_{t+1})
12.	Recombinação(Q_{t+1})
13.	Mutação(Q_{t+1}, n)
14.	$t = t + 1$

Algoritmo 5: NSGA-II com o operador de mutação proposto.

Na linha 8, calculamos a diferença Δ entre o maior e o menor valor para a *crownding distance* da população P_{t+1} (os valores originais já foram previamente calculados na linha 4). Na linha 9, obtemos *n* através da divisão entre a função sigmóide aplicada na geração atual (*t*) pelo valor de Δ. Na linha 13 aplicamos a mutação na população Q_{t+1} utilizando *n* como parâmetro. As demais situações continuam iguais ao algoritmo original.

6 – Experimentos

Para validar o modelo proposto, realizamos diversos experimentos com o intuito de avaliar a diversidade entre as soluções finais, bem como suas distâncias para a real fronteira de Pareto. Nas seções seguintes descreveremos os problemas, as métricas e as configurações utilizadas nos experimentos, bem como os resultados obtidos.

6.1 – Problemas

Os problemas utilizados nos experimentos pertencem à família ZDT proposta em Zitzler et al. (2000). O primeiro problema, ZDT1, possui $n = 30$ variáveis de decisão, limitadas no intervalo $[0,1]$, uma fronteira de Pareto contínua (Figura 11), convexa e com uma distribuição uniforme de soluções. A maior dificuldade deste problema é o fato de haver muitas variáveis de decisão. As funções que descrevem o problema ZDT1 são:

$$f_1(x) = x_1$$

$$f_2(x, g) = g(x) \cdot \left(1 - \sqrt{\frac{f_1}{g(x)}}\right)$$

$$g(x) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$$

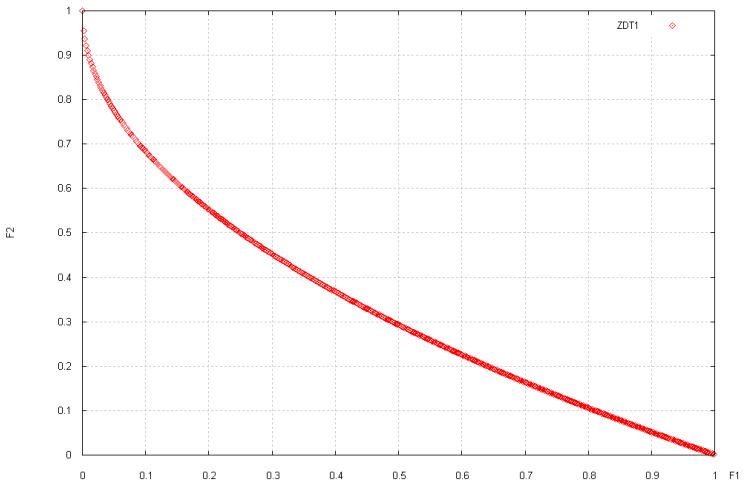

Figura 11. Fronteira de Pareto do problema ZDT1.

O segundo problema, ZDT2, possui $n = 30$ variáveis de decisão, limitadas no intervalo $[0,1]$, uma fronteira de Pareto contínua (Figura 12), não-convexa e com uma distribuição uniforme de soluções. A maior dificuldade deste problema é que a região de pontos ótimos de Pareto é não-convexa. As funções que descrevem o problema ZDT2 são:

$$f_1(x) = x_1$$

$$f_2(x,g) = g(x) \cdot \left[1 - \left(\frac{f_1}{g(x)} \right)^2 \right]$$

$$g(x) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$$

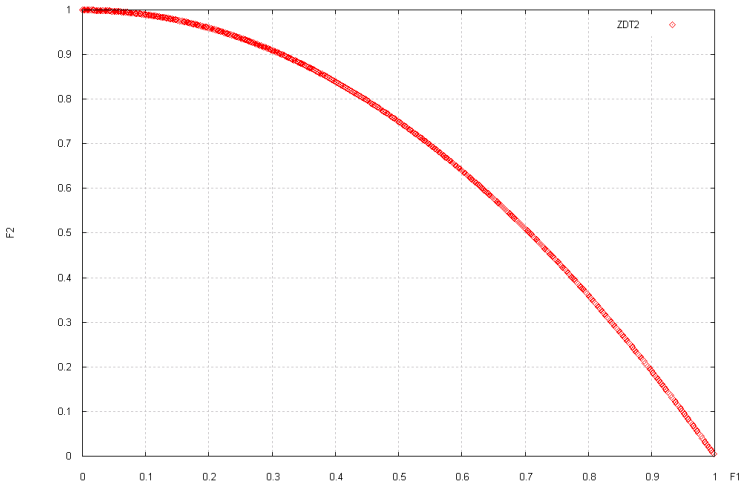

Figura 12. Fronteira de Pareto do problema ZDT2.

O terceiro problema, ZDT3, possui $n = 30$ variáveis de decisão, limitadas no intervalo $[0,1]$ e uma fronteira de Pareto não-contínua (Figura 13), o que proporciona extrema dificuldade para se obter pontos ótimos de Pareto. As funções que descrevem este problema são:

$$f_1(x) = x_1$$

$$f_2(x,g) = g(x) \cdot \left[1 - \sqrt{\frac{f_1}{g(x)}} - \left(\frac{f_1}{g(x)} \right) \cdot sen(10 \cdot \pi \cdot f_1) \right]$$

$$g(x) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$$

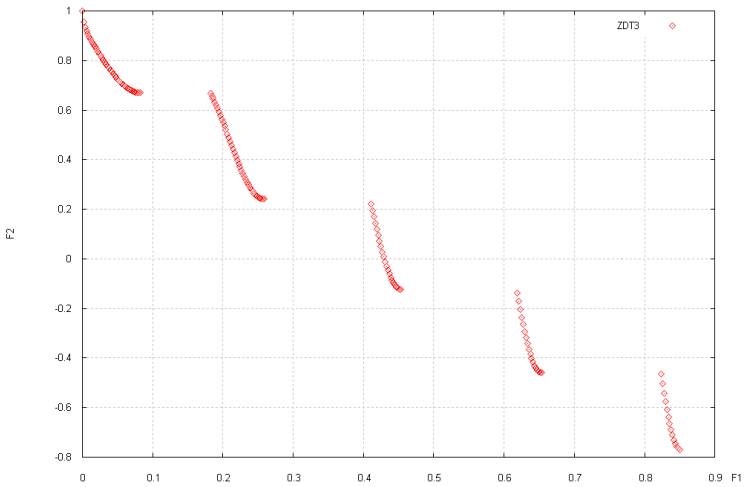

Figura 13. Fronteira de Pareto do problema ZDT3.

O quarto problema, ZDT4, possui $n = 10$ variáveis de decisão. A primeira variável x_1 pertence ao intervalo $[0,1]$ e as demais ao intervalo $[-5,5]$. A fronteira de Pareto resultante e contínua (Figura 14), convexa e com uma distribuição uniforme de soluções. A maior dificuldade deste problema advém do fato de existirem cerca de 100 fronteiras de Pareto não-ótimas. As funções que descrevem este problema são:

$$f_1(x) = x_1$$

$$f_2(x,g) = g(x) \cdot \left(1 - \sqrt{\frac{f_1}{g(x)}}\right)$$

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right)$$

O quinto problema, ZDT6, possui $n = 10$ variáveis de decisão, limitadas no intervalo $[0,1]$, uma fronteira de Pareto contínua (Figura 15), não-convexa e com uma distribuição não-uniforme de soluções, o que acarreta na maior dificuldade para o problems. As funções que descrevem a ZTD6 são:

$$f_1(x) = 1 - \exp(-4 \cdot x_1) \cdot sen^6(6 \cdot \pi \cdot x_1)$$

$$f_2(x,g) = g(x) \cdot \left[1 - \left(\frac{f_1}{g(x)}\right)^2\right]$$

$$g(x) = 1 + 9\left[\sum_{i=2}^{n}\left(\frac{x_i}{9}\right)\right]^{0.25}$$

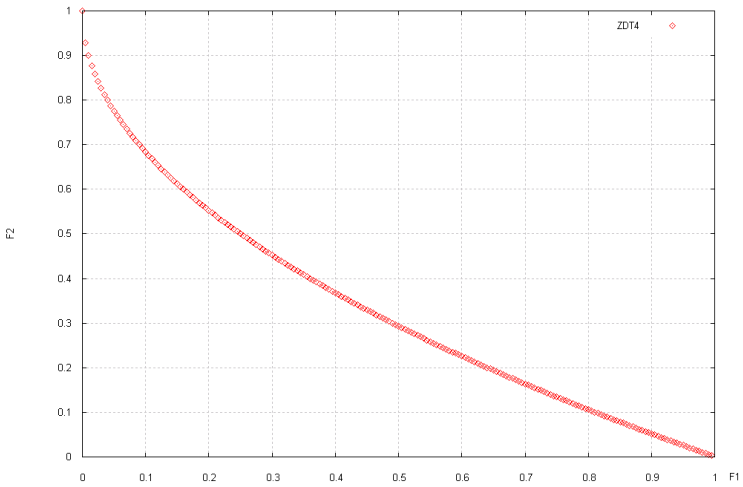

Figura 14. Fronteira de Pareto do problema ZDT4.

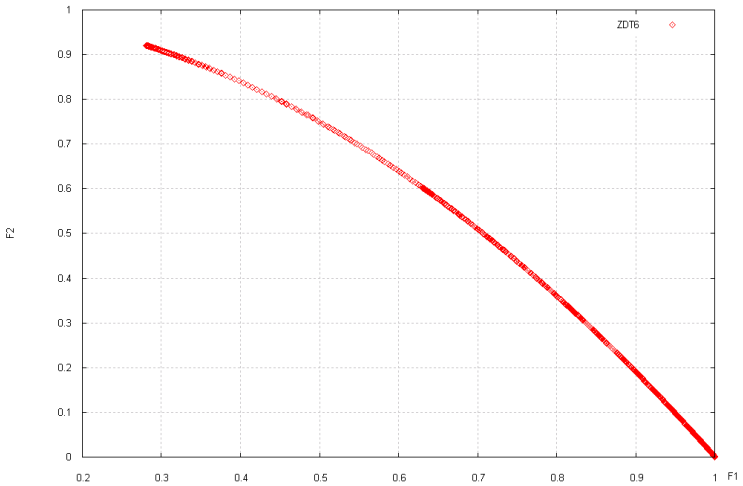

Figura 15. Fronteira de Pareto do problema ZDT6.

Faz mister citar que existe ainda o problema ZDT5 que não foi utilizado pois ele utiliza variáveis de decisão binárias. Entretanto, a solução aqui apresentada pode ser facilmente adaptada para o mesmo.

6.2 – Métricas

Como a convergência para a fronteira ótima de Pareto e a manutenção de uma distribuição uniforme ao longo dela são objetivos distintos (e de certo modo conflitantes) na otimização multi-objetivo, não existe uma métrica única

que decida a performance de um algoritmo de forma absoluta (Deb, 2000). Logo, duas métricas diferentes para avaliar adequadamente tais objetivos serão necessárias. Nas subseções a seguir detalharemos as métricas utilizadas neste trabalho.

6.2.1 – Generational Distance

Esta métrica, proposta por Veldhuizen (1999), calcula explicitamente a proximidade de um conjunto Q com N soluções para o conjunto de pontos ótimos de Pareto P^* através da equação:

$$GD = \frac{\left(\sum_{i=1}^{|Q|} d_i^{\,p} \right)^{\frac{1}{p}}}{|Q|}$$

Para $p = 2$, o parâmetro d_i é a distância euclidiana (no espaço de busca) entre a solução $i \in Q$ e o membro mais próximo de P^*:

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^{M} \left(f_m^{(i)} - f^{*(k)}_m \right)^2}$$

onde $f^{*(k)}_m$ é o valor da m-ésima função objetivo do k-ésimo membro de P^*. É válido ressaltar que, para diferentes graus de magnitude, os valores das funções objetivo precisam estar normalizados e que deverá existir uma grande quantidade de soluções presentes em P^* para tornar o cálculo da distância confiável. Pode-se concluir desta métrica que quanto menor for seu valor, maior será a convergência conquistada pelo algoritmo, pois mais próximas estarão as soluções obtidas das soluções ótimas de Pareto.

6.2.2 – *Spread*

Métrica proposta por Deb et al. (2002) para obter a diversidade entre as soluções não-dominadas de uma população. Sua equação pode ser vista abaixo:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} \left| d_i - \overline{d} \right|}{\sum_{m=1}^{M} d_m^e + |Q| \overline{d}}$$

onde d_i é uma medida de distância entre soluções vizinhas (geralmente usa-se a distância euclidiana) e $\overline{d}$ é a média destas distâncias. O parâmetro d_m^e é a distância entre as soluções extremas de P^* e Q. Um exemplo deste parâmetro e das distâncias entre soluções consecutivas pode ser visto na Figura 16.

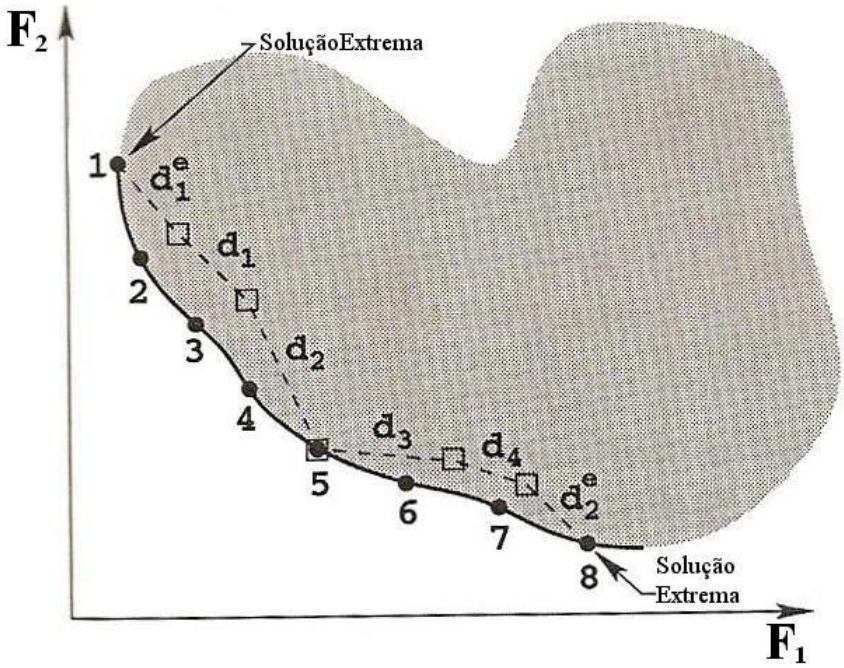

Figura 16. Ilustração da métrica *spread*.

Esta métrica possui um valor ideal igual a zero quando todas as soluções estão distribuídas uniformemente na fronteira de Pareto. Logo, um algoritmo que encontra um baixo Δ encontrará uma melhor diversidade entre as soluções não-dominadas.

6.3 – Resultados

Para validar o operador de mutação proposto, criamos três diferentes configurações do algoritmo NSGA-II, com o intuito de comparar os resultados

obtidos por um algoritmo utilizando o operador de mutação adaptativo com um algoritmo utilizando um operador estático:

1. NSGA-II com o operador de mutação adaptativo proposto neste trabalho;
2. NSGA-II utilizando o operador de mutação estático com o parâmetro de mutação $n = 20$ (mutação branda);
3. NSGA-II utilizando o operador de mutação estático com o parâmetro de mutação $n = 5$ (mutação forte);

Os demais parâmetros do algoritmo foram mantidos constantes em todas as configurações. Para facilitar a visualização da convergência e da distribuição das soluções, utilizamos uma população relativamente pequena com 30 indivíduos. Foi utilizada uma probabilidade de mutação igual a $1/n$ (para n igual ao número de variáveis), o operador de recombinação SBX (Deb e Goyal, 1996) com variância igual a 20 e probabilidade de ocorrência igual a 0.90.

Propomos dois diferentes cenários para os experimentos. No primeiro, para cada problema apresentado na seção 6.2, executamos 100 vezes cada configuração (de forma independente) do NSGA-II durante 100 gerações. Os resultados para os valores do *spread* (Tabela 1) e *generational distance* (Tabela 2) obtidos pelas soluções não-dominadas resultantes são exibidos em duas colunas: Na primeira calculamos a média das execuções e na segunda o desvio-padrão. O intuito de tal experimento é visualizar a velocidade de convergência e a obtenção da diversidade ocasionada pelas diferentes configurações propostas em uma baixa quantidade de gerações.

Tabela 1 – Valores obtidos para o *spread* utilizando 100 gerações.

Mutação	ZDT1		ZDT2		ZDT3		ZDT4		ZDT6	
n = 5	**0.3932**	0.0556	0.4644	0.0971	0.6222	0.0314	**0.9172**	0.0764	0.4444	0.1793
n = 20	0.5310	0.0748	0.8641	0.1361	0.6972	0.0476	0.9646	0.0533	0.4726	0.1464
Adaptativo	0.4009	0.0546	**0.4134**	0.0630	**0.6183**	0.0352	0.9327	0.0609	**0.4064**	0.0853

Tabela 2 – Valores obtidos para o *generational distance* utilizando 100 gerações.

Mutação	ZDT1		ZDT2		ZDT3		ZDT4		ZDT6	
n = 5	0.0032	0.0007	0.0033	0.0013	0.0022	0.0010	**0.3280**	0.1823	0.0092	0.0208
n = 20	0.0255	0.0100	0.0581	0.0485	0.0189	0.0077	0.5558	0.3734	0.0096	0.0138
Adaptativo	**0.0028**	0.0011	**0.0022**	0.0006	**0.0018**	0.0010	0.5390	0.2924	**0.0035**	0.0034

No segundo cenário, para cada problema apresentado na seção 6.2, executamos 100 vezes cada configuração (de forma independente) do NSGA-II durante 1000 gerações e novamente calculamos a média e o desvio-padrão para os valores do *spread* (Tabela 3) e *generational distance* (Tabela 4) obtidos pelas soluções não-dominadas resultantes. O intuito de tal experimento é visualizar a convergência e diversidade ao término de uma quantidade maior de gerações.

Tabela 3 – Valores obtidos para o *spread* utilizando 1000 gerações.

Mutação	ZDT1		ZDT2		ZDT3		ZDT4		ZDT6	
n = 5	0.4302	0.0634	0.4426	0.0607	0.6254	0.0363	0.4708	0.0718	0.7215	0.0946
n = 20	0.4226	0.0683	**0.4239**	0.0581	**0.6157**	0.0432	0.4597	0.0645	0.7284	0.0859
Adaptativo	**0.4108**	0.0590	0.4488	0.0677	0.6261	0.0398	**0.4509**	0.0662	**0.7108**	0.0854

Tabela 4 – Valores obtidos para o *generational distance* utilizando 1000 gerações.

Mutação	ZDT1		ZDT2		ZDT3		ZDT4		ZDT6	
n = 5	0.0011	0.0004	**0.0009**	0.0005	0.0008	0.0001	0.0020	0.0011	0.0014	0.0111
n = 20	**0.0009**	0.0003	**0.0009**	0.0004	**0.0007**	0.0001	**0.0009**	0.0003	0.0003	0.0001
Adaptativo	**0.0009**	0.0003	0.0010	0.0004	0.0008	0.0001	0.0011	0.0004	**0.0002**	0.0001

Marcamos de negrito em cada tabela os melhores (menores) valores médios obtidos. Da Tabela 1 podemos inferir que na maioria dos problemas (três dos cinco) o operador de mutação ajudou a criar rapidamente uma distribuição uniforme ao longo da fronteira formada pelas soluções não-dominadas. Como era de se esperar, o operador com $n = 20$ obteve piores

resultados em todos os problemas visto que aplicou uma mutação demasiadamente fraca em um estado inicial caótico.

Da Tabela 2 obtivemos os resultados que comprovam que o uso do operador adaptativo favorece a convergência mais rápida, obtendo em 4 dos 5 problemas propostos o menor valor para a *genertional distance*. Novamente, o operador com $n = 20$ obteve piores resultados em todos os problemas.

No segundo cenário, novamente o operador proposto foi melhor do que os demais no quesito diversidade, obtendo uma distribuição mais uniforme ao longo da fronteira formada pelas soluções não-dominadas em 3 dos 5 problemas (Tabela 3). Nos outros 2 problemas, o operador com $n = 20$ obteve melhores resultados.

Por fim, a Tabela 4 nos informa que no quesito convergência ao longo de várias gerações, o operador adaptativo foi o segundo melhor, atrás apenas do operador com parâmetro $n = 20$, obtendo a melhor performance em 2 dos 5 problemas propostos. Entretanto, devido à proximidade dos valores e aos resultados extremamente baixos obtidos, os mesmo não confirmam (de forma absoluta) qual configuração utilizada seria considerada a melhor para a convergência ao longo de várias gerações. As três poderiam ser consideradas igualmente eficazes.

Como pode ser observado dos experimentos, nos cenários propostos a mutação adaptativa conseguiu obter melhor distribuição das soluções e uma maior velocidade na convergência para a fronteira de Pareto. Logo, o operador de mutação adaptativo proposto conseguiu aliar a eficiência de uma forte mutação, que deve ser aplicada nos estágios iniciais do processo evolucionário, com a eficácia de se aplicar uma mutação mais branda nos estágios finais e, com isto, aproximar-se da fronteira de Pareto.

O grande benefício da abordagem aqui proposta reside no fato de não ser necessário o teste de várias configurações (diferentes valores de n) para se obter os melhores resultados em diferentes problemas, visto que os valores conseguidos com a mutação adaptativa foram extremamente satisfatórios.

7 – Conclusões & Trabalhos Futuros

Após uma introdução à área da otimização multi-objetivo e à computação evolucionária, este trabalho apresentou um estudo sobre as técnicas mais utilizadas para se manter soluções distintas durante o processo evolucionário.

Com o intuito de acelerar a convergência para a fronteira de Pareto e propiciar uma distribuição uniforme das soluções finais ao longo da fronteira, apresentamos em seguida um operador de mutação adaptativo para MOEAs que utiliza informações sobre a máxima e a mínima *crowding distance* de uma população para guiar a intensidade da mutação.

A motivação do novo operador se baseia nos algoritmos que utilizam unicamente um operador de mutação não-dinâmico para se gerar soluções diversas, o que pode acarretar ineficiência e ineficácia tanto na convergência quanto na obtenção da diversidade.

Após alguns experimentos realizados, os resultados obtidos foram encorajadores acerca da eficácia e eficiência do método. Na maioria dos problemas testados, o algoritmo NSGA-II com o uso do operador proposto convergiu mais rapidamente que o NSGA-II original, gerando também uma distribuição de soluções mais uniforme.

Como trabalhos futuros, novos experimentos devem ser realizados com problemas mais complexos, contendo maiores números de funções objetivo. Além disto, podemos estender o conceito aqui criado para controlar outros parâmetros no processo evolucionário, como exemplo o parâmetro que controla a variância do operador de recombinação ou a probabilidade de ocorrência da mutação

É válido ressaltar que o escopo do operador de mutação aqui apresentado é focado na população, ou seja, a mudança no parâmetro da variância se aplica igualmente a todos os indivíduos. Uma adaptação que tenha como escopo um indivíduo, isto é, uma mudança de parâmetros personalizada para cada solução, seria provavelmente mais eficaz.

Referência Bibliográfica

Aström, K. J. e Wittenmark, B. (1995). *Adaptative Control*. Addison-Wesley, 2 edição.

Back, T. (1992). Self-adaptation in genetic algorithms. Em *Proceedings of the First European Conference on Artificial Life*, p. 263-271.

Bui, L. T.; Branke, J. e Abbass, H. A. (2005). Multi-objective optimization for dynamic environments. Em *proceedings of the Congress on Evolutionary Computation*.

Carvalho, A. G. ; Araujo, A. F. R. ; Silva, I. B. V. e Daniel, C. D. A. (2008) Um modelo de otimização para o problema de design ótimo da programação N-Versões. Em *Anais do VIII Encontro Regional de Matemática Aplicada e Computacional*.

Chaiyaratana, N.; Piroonratana, T. e Sangkawelert, N. (2007). Effects of diversity control in single-objective and multi-objective genetic algorithms. *Journal of Heuristics*, 13, p. 01- 34.

Coello, C. A. C.; Lamont, G. B. e Veldhuizen, D. A. V. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favored Races in the Struggle for Life*. John Murray.

Deb, K. (1989). Genetic algorithms in Multi-Modal Function Optimization. *Master's Thesis*, Tuscaloosa, AL. University of Alabama.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley.

Deb, K. e Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. Em *proceedings of the Third International Conference on Genetic Algorithms*, p. 42-50.

Deb, K. e Goyal, M. (1996). A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4), p. 34-45.

Deb, K.; Pratap, A.; Agarwal, S. e Meyarivan, T. (2002). A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), p. 182-197.

Eiben A. E.; Hinterding, R. e Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2), p. 124-141.

Eiben, A. E. e Smith, J. E. (1998). *Introduction to Evolutionary Computing*. Springer.

Fonseca, C. M. e Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. Em *proceedings of the Fourth International Conference on Evolutionary Programming*, p. 355-365.

Goldberg, D. E. e Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. Em *proceedings of the First International Conference on Genetic Algorithms and Their Applications*, p. 41-49.

Gouvea, M. M e Araujo, A. F. R. (2007). Diversity-based model reference for genetic algorithms in dynamic environment. Em *proceedings of the 2007 IEEE International Congress on Evolutionary Computation*, p. 01-07.

Hajela, P.; Lee, E. e Lin, C. Y. (1993). Genetic algorithms in structural topology optimization. Em *Proceedings of the NATO Advanced Research Workshop on Topology Design of Structures*, p. 117-133.

Hinterding, R.; Michalewicz, Z. e Eiben, A. E. (1997). Adaptation in evolutionary computation: a survey. Em *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, p. 65-69.

Horn, J.; Nafploitis, N. e Goldberg, D. (1994). A niched Pareto genetic algorithm for multi-objective optimization. Em *Proceedings of the First IEEE Conference on Evolutionary Computation*, p. 82-87.

Kingsland, E. S. (1995). *Modeling Nature - Episodes in the History of Population Ecology*. The University of Chicago Press.

Kung, H. T., Luccio, F., Preparata, F. P. (1975). On Finding the Maxima of a Set of Vectors. *Journal of the Association for Computing Machinery*, 22(4), p. 469-476.

Landau, I. D. (1981). Model reference adaptive controllers and stochastic self-tuning regulators – a unified approach. *Journal of Dynamic Systems, Measurement, and Control*, 103(4), p. 404-416.

Laumanns, M.; Thiele, L.; Deb, K. e Zitzler, E. (2002). Combining Convergence and Diversity in Evolutionary Multi-Objective Optimization. *Evolutionary Computation*, 10(3), p. 263-282.

Meiyi, L.; Zixing, C. e Guoyun, S. (2004). An adaptive genetic algorithm with diversity-guided mutation and its global convergence property. *Journal of Central South University of Technology*, 11(3), p. 323-327**.**

Oei, C. K., Goldberg, D. E. e Chang, S. J. (1991). Tournament selection, niching, and the preservation of diversity. Em *IlliGAl Report N. 91011, University of Illinois at Urbana-Champaign*, Urbana, IL, EUA.

Prügel-Bennett, A. (1997). Modeling evolving populations. *Journal of Theoretical Biology*, 185, p. 81-95**.**

Smith, J. E. e Fogarty, T. C. (1997). Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1 (2), p. 81-87.

Srinivas, N. e Deb, K. (1994). Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation Journal*, 2 (3), p. 221-248.

Uyar, A. S. e Harmanci, A. E. (2005). A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments. *Soft Computing*, 9, p. 803-814.

Veldhuizen, D. V. (1999). Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. *Ph. D. Thesis, Dayton, OH: Air Force Institute of Technology*. Technical Report No. AFIT/DS/ENG/99-01.

Wagg, D. J. (2003). Adaptive control of nonlinear dynamical systems using a model reference approach. *MECCANICA*, 38 (2), p. 227-238.

Zitzler, E.; Deb, K. e Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation Journal*, 8 (2), p. 125-148.

Zitzler, E. e Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto Approach. Em *Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH)*, Zürich, Switzerland.