ALLYSSON STEVE MOTA LACERDA

PROPOSTA DE UM ALGORITMO EVOLUCIONÁRIO NEBULOSO PARA SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO MULTIOBJETIVO

Belo Horizonte – MG
Junho de 2010

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

PROPOSTA DE UM ALGORITMO EVOLUCIONÁRIO NEBULOSO PARA SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO MULTIOBJETIVO

Proposta de dissertação apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

ALLYSSON STEVE MOTA LACERDA

Belo Horizonte – MG
Junho de 2010

Resumo

Parte significativa das pesquisas em Otimização tem sido feita na linha de Otimização Multiobjetivo, área na qual os algoritmos evolutivos ganharam especial destaque. Isso ocorre, principalmente, devido à facilidade de tais métodos em trabalhar com problemas complexos e à qualidade das soluções encontradas. Baseado nisso, o presente trabalho propõe o Fuzzy-DE (*Fuzzy-based Differential Evolution Algorithm*), um novo algoritmo evolucionário multiobjetivo que engloba conceitos de Sistemas Nebulosos e Evolução Diferencial para o tratamento de problemas restritos e irrestritos. Seu principal objetivo é poder ser aplicado em problemas diversos sem a necessidade de se realizar grandes modificações em sua estrutura, tornando-o bastante generalista. Além disso, os testes mostraram que seus resultados são bastante competitivos e seu custo computacional baixo, quando comparado a outros métodos existentes.

Abstract

Multiobjective Optimization plays a major role in Optimization researches. In such area, evolutionary algorithms have gained special highlight, mainly due to the easiness to handle complex problems and the quality of the solutions found. Based on this, we propose Fuzzy-DE (Fuzzy-based Differential Evolution Algorithm), a new multiobjective evolutionary algorithm that incorporates concepts of Fuzzy Systems and Differential Evolution to handle constrained and unconstrained problems. The main goal is to address a large range of problems without the need to do major changes in its structure, making it very generalist. Moreover, tests showed competitive results and a low computational cost, when compared to other existing methods.

Agradecimentos

A meus pais (Maria Lúcia e Abissay) e irmãos (Abissay Júnior, Ábia Madeleine e Árllen Dick) por todo o carinho ao longo da minha vida.

Aos professores Walmir Caminhas e Rodney Saldanha pelo excelente trabalho de orientação durante o mestrado.

Aos amigos do Laboratório de Inteligência Computacional (LABICOMP), em especial a Luciana Balieiro, que muito contribuiu para o desenvolvimento dessa dissertação de Mestrado.

Aos colegas do Laboratório de Computação Evolucionária (LCE), que também fizeram parte do desenvolvimento das nossas pesquisas.

Aos demais amigos do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) e do Programa de Pós-Graduação em Ciência da Computação (PPGCC) da UFMG, cujos nomes eu não me atrevo a listar, sob o risco de cometer alguma injustiça.

Sumário

1 Introdução 1

1.1 Objetivos 1

1.2 Motivação 1

1.3 Organização do trabalho 2

2 Referencial teórico 3

2.1 Computação Evolucionária 3

2.1.1 Processo evolucionário 3

2.1.2 Histórico 4

2.2 Otimização multiobjetivo 8

2.2.1 O conjunto Pareto-ótimo 9

2.2.2 Métodos de resolução 10

2.3 Evolução Diferencial 15

2.3.1 Estrutura básica 16

2.4 Sistemas Nebulosos 18

2.4.1 Conjuntos nebulosos 18

2.4.2 Funções de pertinência 19

2.4.3 Operadores nebulosos 20

3 O algoritmo proposto 23

3.1 Estrutura do algoritmo 23

3.2 Especificidades do algoritmo 25

3.2.1 Geração da nova população (passo 2) 25

3.2.2 Cálculo da distância ponderada (passo 6.1) 25

3.2.3 Cálculo da densidade ponderada (passo 6.2) 26

3.2.4 Cálculo da violação ponderada (passo 6.3) 26
3.2.5 Cálculo do *fitness* (passo 6.4) 26
3.2.6 Seleção dos indivíduos (passos 7, 8 e 9) 29
3.3 Parâmetros utilizados 29
4 Testes realizados 31
4.1 Métricas de desempenho 31
4.1.1 Cardinalidade 31
4.1.2 Hipervolume 31
4.1.3 Espalhamento 32
4.1.4 Indivíduos dominados 32
4.1.5 Indivíduos dominantes 32
4.1.6 Tempo 32
4.1.7 Distância geracional 32
4.1.8 Distância geracional invertida 32
4.2 Competição do CEC 2009 33
4.2.1 Problema UF1 35
4.2.2 Problema UF2 36
4.2.3 Problema UF3 37
4.2.4 Problema UF4 38
4.2.5 Problema UF5 39
4.2.6 Problema UF6 40
4.2.7 Problema UF7 41
4.3 Problema TEAM22 42
4.3.1 Geometria do problema 42
4.3.2 Padronização do TEAM22 43

4.4 Resultados 44

4.4.1 Competição do CEC 2009 45

4.4.2 Problema TEAM22 55

5 Conclusões e trabalhos futuros 59

Referências bibliográficas 61

Lista de figuras

Figura 2-1 Representação da operação de cruzamento....6
Figura 2-2 Funções de exemplo e fronteira Pareto....10
Figura 2-3 Ordenação em camadas....12
Figura 2-4 Cálculo da Distância de Multidão....13
Figura 2-5 Operação de mutação diferencial....17
Figura 2-6 Resultado da geração de um novo indivíduo utilizando Evolução Diferencial..17
Figura 2-7 Representação de um conjunto crisp (a) e um conjunto nebuloso (b)....19
Figura 2-8 Funções de pertinência....20
Figura 2-9 Exemplos de t-normas....21
Figura 2-10 Exemplos de s-normas....22
Figura 3-1 Sequência de execução do Fuzzy-DE....24
Figura 3-2 Sensibilidade da seleção ao valor do peso....28
Figura 4-1 Fronteiras Pareto real e obtida....33
Figura 4-2 Conjunto e fronteira Pareto para o problema UF1....35
Figura 4-3 Conjunto e fronteira Pareto para o problema UF2....36
Figura 4-4 Conjunto e fronteira Pareto para o problema UF3....37
Figura 4-5 Conjunto e fronteira Pareto para o problema UF4....38
Figura 4-6 Conjunto e fronteira Pareto para o problema UF5....39
Figura 4-7 Conjunto e fronteira Pareto para o problema UF6....40
Figura 4-8 Conjunto e fronteira Pareto para o problema UF7....41
Figura 4-9 Geometria do SMES....42
Figura 4-10 Tempo de processamento para os problemas do CEC 2009....47

Figura 4-11 Fronteira Pareto obtida para o problema UF1 48

Figura 4-12 Fronteira Pareto obtida para o problema UF2 49

Figura 4-13 Fronteira Pareto obtida para o problema UF3 50

Figura 4-14 Fronteira Pareto obtida para o problema UF4 51

Figura 4-15 Fronteira Pareto obtida para o problema UF5 52

Figura 4-16 Fronteira Pareto obtida para o problema UF6 53

Figura 4-17 Fronteira Pareto obtida para o problema UF7 54

Figura 4-18 Representação do SMES com os resultados obtidos (03 parâmetros) 56

Figura 4-19 Diagrama do SMES com os resultados obtidos (08 parâmetros) 57

Lista de tabelas

Tabela 4.1 Configuração do SMES para 03 parâmetros 44

Tabela 4.2 Configuração do SMES para 08 parâmetros 44

Tabela 4.3 Resultado do IGD para os problemas do CEC 2009 45

Tabela 4.4 Resultado normalizado do IGD para os problemas do CEC 2009.....46

Tabela 4.5 Classificação dos algoritmos para os problemas do CEC 2009.....46

Tabela 4.6 Tempo de processamento, em segundos, para os problemas do CEC 2009..47

Tabela 4.7 Resultados das métricas para o problema UF1.....48

Tabela 4.8 Resultados das métricas para o problema UF2.....49

Tabela 4.9 Resultados das métricas para o problema UF3.....50

Tabela 4.10 Resultados das métricas para o problema UF4.....51

Tabela 4.11 Resultados das métricas para o problema UF5.....52

Tabela 4.12 Resultados das métricas para o problema UF6.....53

Tabela 4.13 Resultados das métricas para o problema UF7.....54

Tabela 4.14 Resultados obtidos para o TEAM22 com 03 parâmetros.....56

Tabela 4.15 Resultados obtidos para o TEAM22 com 08 parâmetros.....57

1 Introdução

A crescente busca por competitividade e desempenho tem levado um número cada vez maior de empresas e universidades a investir em pesquisas na área de Otimização. Além disso, uma importante fatia de tais investimentos tem sido aplicada na área de Otimização Multiobjetivo, cujos problemas possuem vários objetivos conflitantes que precisam ser otimizados simultaneamente.

Embora haja diversas abordagens para a solução de problemas multiobjetivo, os algoritmos evolutivos ganharam especial destaque nessa linha, especialmente devido à sua característica de trabalhar com populações de indivíduos. Como será apresentado, isso é muito desejável em tais problemas, já que não existe mais apenas uma solução ótima.

1.1 Objetivos

O principal objetivo do presente trabalho é apresentar um novo algoritmo evolucionário multiobjetivo que seja o mais generalista possível, isto é, que possa ser aplicado em problemas diversos sem a necessidade de se realizar grandes modificações em sua estrutura. Além disso, é desejável que o seu custo computacional seja relativamente baixo, especialmente quando comparado a outros métodos existentes.

Entretanto, de nada adianta o método ter baixo custo computacional se ele não apresentar bons resultados. Por isso, outro fator muito importante a ser levado em consideração é a qualidade das soluções encontradas. Para tal, o Fuzzy-DE será testado em problemas amplamente difundidos e complexos, além de ter seus resultados avaliados utilizando métricas reconhecidas mundialmente.

1.2 Motivação

Apesar das inúmeras pesquisas na área de Otimização Multiobjetivo, essa linha de pesquisa ainda está muito longe de ser esgotada. Isso se deve ao fato de muitos algoritmos trabalharem com conjuntos bastante restritos de problemas. Além disso, a

descoberta de novos modelos computacionais ou a agregação de modelos existentes pode auxiliar no desenvolvimento de novos métodos, ainda mais eficazes.

1.3 Organização do trabalho

Essa dissertação está organizada em capítulos, como descrito a seguir. O Capítulo 2 faz uma introdução sobre a área de Computação Evolucionária e algumas de suas características. O Capítulo 3 discorre sobre Otimização Multiobjetivo e apresenta alguns dos principais algoritmos evolutivos multiobjetivo encontrados na literatura. Já o Capítulo 4 apresenta o algoritmo Evolução Diferencial, enquanto o Capítulo 5 faz um breve resumo sobre Sistemas Nebulosos. A partir de conceitos extraídos de tais temas, é proposto, no Capítulo 6, o Fuzzy-DE, que será aplicado aos problemas do Capítulo 7. Os resultados de tais testes e uma comparação com outros métodos são apresentados no Capítulo 8. Por fim, no Capítulo 9, são apresentadas algumas considerações sobre o método proposto, conclusões e trabalhos futuros.

As áreas abordadas são bastante amplas e, por isso, o presente trabalho não busca ser uma referência completa sobre quaisquer dos temas. Entretanto, são apresentadas ao longo do texto diversas referências de excelente qualidade, que podem ser utilizadas para um estudo mais completo.

2 Referencial teórico

1.4 Computação Evolucionária

De todas as áreas da Computação Natural, os sistemas bioinspirados são os mais antigos e possivelmente os mais populares. Eles envolvem o estudo de fenômenos e processos biológicos, a fim de desenvolver sistemas computacionais e algoritmos capazes de resolver problemas complexos (de Castro and Von Zuben 2004).

Como os sistemas bioinspirados são, basicamente, projetados para resolver problemas, algumas propostas não são concebidas com o propósito de serem modelos com grande precisão. Em muitos casos, eles possuem um alto nível de abstração sobre alguns mecanismos da biologia (De Jong 2006).

Existem diversas abordagens bioinspiradas, dentre as quais podemos citar as redes neurais artificiais (*artificial neural networks*), inteligência de enxame (*swarm intelligence*), sistemas imunes artificiais (*artificial immune systems*) e algoritmos evolucionários (*evolutionary algorithms*), dentre outros.

No entanto, o presente trabalho aborda apenas os algoritmos evolucionários (EAs), que compreendem boa parte da área de Computação Evolucionária. Estudos mais completos sobre os diversos temas podem ser obtidos em (de Castro and Von Zuben 2004).

1.4.1 Processo evolucionário

De maneira geral, para um algoritmo ser considerado evolucionário, ele deve possuir:

- Uma ou mais populações de indivíduos competindo por recursos limitados;
- Populações dinâmicas, devido ao nascimento e à morte de indivíduos;
- Um conceito de aptidão (*fitness*) que reflete a habilidade de um indivíduo de sobreviver e se reproduzir;
- Um conceito de herança, segundo o qual os indivíduos são parecidos com os pais, mas não idênticos.

Tal caracterização define um sistema evolucionário como um processo que, a partir de certas condições iniciais, segue uma trajetória ao longo do tempo, através de um complexo espaço de estados (De Jong 2006).

1.4.2 Histórico

Embora a ideia básica de evolução como um processo computacional tenha sido trabalhada na primeira metade do século XX, tal abordagem ganhou destaque a partir da década de 1960 (De Jong 2006). Muitos grupos de pesquisa perceberam que mesmo modelos evolucionários simples podiam ser utilizados para resolver problemas computacionais complexos.

O princípio era conseguir alguma inspiração na natureza, mesmo que não fosse um modelo totalmente fiel (ou plausível) de um processo biológico. O ponto chave era identificar alguns aspectos do processo evolutivo que fossem úteis. Como consequência, ainda hoje, muitos algoritmos evolucionários utilizam algumas simplificações, como populações de tamanho fixo e funções de *fitness* estáticas, dentre outras características.

A fim de implementar tais algoritmos, decisões de projetos muito específicas tinham que ser tomadas, especialmente no que diz respeito à manutenção da população ou seleção de indivíduos para a geração de uma nova população. Tais decisões resultaram em três diferentes espécies de EAs: Programação Evolucionária (E*volutionary Programming*), Estratégias Evolutivas (*Evolutionary* Strategies) e Algoritmos Genéticos (*Genetic Algorithms*).

1.4.2.1 Programação Evolucionária

O paradigma da Programação Evolucionária, proposto por (Fogel, Owens and Walsh 1966), foi baseado em modelos envolvendo uma população de tamanho fixo N. A cada geração, uma nova população de mesmo tamanho é gerada e combinada com a anterior, formando uma nova população com $2N$ indivíduos. Então, ela é ordenada pelo *fitness* e apenas os N melhores são selecionados para a próxima geração. De maneira geral, a EP pode ser resumida em dois grandes passos: realizar mutação da população inicial e selecionar os melhores indivíduos da geração atual e da que sofreu mutação.

Na Programação Evolucionária, a mutação pode ser vista como o acréscimo de um ruído ao indivíduo, conforme a equação 2.1:

$$x_i = x_i + \sqrt{\beta_i \Phi(a) + \sigma_i} * N_i(0,1) \qquad (2.1)$$

onde

β_i e σ_i são exógenos e dependem do problema,

$\Phi(\mathrm{a})$ é a função de mérito ou fitness,

$N_i(0,1)$ é uma ditribuição normal entre -1 e 1.

Alguns autores sugerem outras formas de inserção do ruído, mas a idéia básica permanece a mesma.

Para a seleção, a forma clássica consiste em selecionar, para cada indivíduo, K indivíduos aleatoriamente e compará-los. Os que tiverem maior número de vitórias (melhor fitness) em relação a esses K formarão a próxima geração de indivíduos. Outros autores já sugerem uma seleção simples, apenas pegando os indivíduos com melhor fitness.

Muitos estudos da área de Computação Evolucionária foram focados em itens como estratégias de inicialização, frequência de mutação e operadores de cruzamento (ou recombinação). Seguindo essa linha, nos algoritmos de Programação Evolucionária, o tamanho da população e a intensidade da mutação influenciam fortemente em seu resultado de uma aplicação para outra. Por isso, tais parâmetros precisam ser analisados para cada problema, a fim de se obter bons resultados..

1.4.2.2 Estratégias Evolutivas

Os algoritmos de Estratégias Evolutivas têm como foco funções de otimização com valores reais. Devido a isso, os indivíduos são, naturalmente, representados como vetores de números reais e a reprodução assexuada (mutação) consiste em gerar uma perturbação em um ou mais genes (variáveis) dos pais. Essa perturbação é baseada em uma distribuição normal $G(0,\sigma)$ com média 0 e desvio padrão σ.

Estudos empíricos indicaram que o desempenho do algoritmo de Estratégias Evolutivas era muito dependente da escolha de $G(0,\sigma)$. Como consequência, foi desenvolvido um operador de mutação dinâmico, que co-evolui ao longo do tempo, de acordo com os resultados obtidos até o momento.

Com esse operador de mutação autoadaptativo, os algoritmos de Estratégias Evolutivas alcançaram bons resultados quando comparados às técnicas tradicionais de otimização, como pode ser observado em (Schwefel 1975).

1.4.2.3 Algoritmos Genéticos

Diferentemente dos outros métodos citados anteriormente, o desenvolvimento dos primeiros Algoritmos Genéticos (GAs) teve como objetivo principal criar algoritmos que obtivessem bons resultados independentemente do problema abordado. Para tal, a proposta foi desenvolver uma representação “universal” dos indivíduos e operadores genéticos (De Jong 2006) baseados em *strings* de *bits*.

Em tal proposta, a mutação passou a ser realizada através da inversão de bits com certa probabilidade e o cruzamento através de um operador de *crossover*. Nesse operador, são selecionados dois indivíduos e é definido um ponto de corte, que determina a posição de troca. Cada indivíduo gerado é uma combinação entre os lados de cada um dos pais. A Figura 2-1 mostra o resultado desse operador.

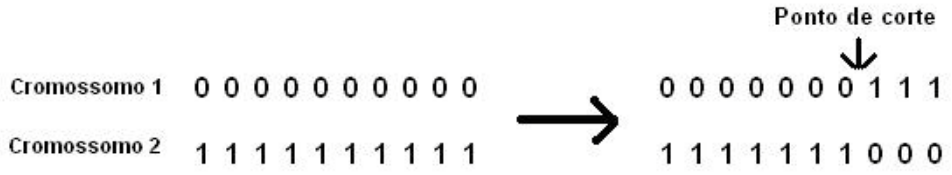

Figura 2-1 Representação da operação de cruzamento

Nos GAs, os pais são selecionados estocasticamente, proporcionalmente a seus valores de *fitness*. Com isso, os melhores indivíduos contribuem com mais “material genético” para a próxima geração. Diversos estudos, tanto analíticos quanto empíricos, apontaram os GAs como um robusto procedimento de busca adaptativo, que se mostrou eficaz como uma heurística de busca global.

Após o desenvolvimento dos métodos citados, o próximo passo foi utilizá-los como base para solução de problemas complexos. Embora isso tenha sido alcançado,

houve uma perda gradativa da inspiração biológica em muitos casos (De Jong 2006), causada por adaptações necessárias para o tratamento de tais problemas.

1.4.2.4 Unificação das áreas de pesquisa

Até o final da década de 1980, os grupos que pesquisavam sobre Programação Evolucionária, Estratégias Evolutivas e Algoritmos Genéticos trabalhavam de maneira isolada. No entanto, como resultado das primeiras interações, foi possível identificar semelhanças e diferenças entre as diversas abordagens, bem como trocar informações e conceitos.

Já no início da década de 1990, a área de pesquisa passou a ser conhecida como Computação Evolucionária (*Evolutionary Computation*) e foi lançado o primeiro periódico sobre o assunto, com o mesmo nome. Devido a essa unificação, houve uma crescente divulgação dos trabalhos realizados, o que levou ao surgimento de diversos grupos de pesquisa. Tais grupos tinham, principalmente, foco em pontos críticos para a melhoria e ampliação da capacidade dos métodos evolucionários (De Jong 2006):

- Prós e contras das representações de genótipos e fenótipos
- Operações não-aleatórias e especialização
- Modelos paralelos e descentralizados
- Sistemas autoadaptativos
- Inclusão das propriedades de Lamarck
- Sistemas coevolucionários
- Modelos baseados em agentes

Devido a essa série de fatores, embora seja uma área relativamente recente, a Computação Evolucionária tomou enormes proporções e representa atualmente uma importante área de estudos, especialmente no campo da Otimização.

Todo esse interesse pela área resultou na criação de publicações e eventos com foco em Computação Evolucionária, como:

- Evolutionary Computation
- IEEE Transactions on Evolutionary Computation

- Genetic Programming and Evolvable Machines
- GECCO: Genetic and Evolutionary Computation Conference
- CEC: Congress on Evolutionary Computation
- PPSN: Parallel Problem Solving from Nature
- FOGA: Foundations of Genetic Algorithms

Além disso, vários excelentes livros foram publicados, com especial destaque para (Goldberg 1989), que impulsionou uma ampla divulgação dessa emergente área de pesquisa. Outras ótimas fontes incluem (Eiben and Smith 2003), (Schwefel 1991), (De Jong 2006), (de Castro and Von Zuben 2004), (Fogel, Back e Michalewics 1997) e (Coello 2006).

1.5 Otimização multiobjetivo

Em problemas de otimização, quando há apenas um objetivo a ser otimizado, procura-se encontrar a melhor solução disponível, chamada de “ótimo global”, ou pelo menos uma boa aproximação dela (Coello 2006). Entretanto, em muitas situações, há diversos objetivos a serem otimizados simultaneamente e, normalmente, tais objetivos são conflitantes entre si. Tais problemas com duas ou mais funções objetivo são conhecidos como Problemas Multiobjetivo (*Multiobjective Problem* – MOP) e requerem novos modelos e algoritmos. Até mesmo o conceito de otimalidade muda quando se trabalha com tais problemas, como será mostrado a seguir.

Otimização Multiobjetivo, também conhecida como Otimização Multicritério ou Otimização Vetorial, pode ser definida como o problema de encontrar (Osyczka 1985):

> *“um vetor de variáveis de decisão que satisfaça um conjunto de restrições e otimize um vetor de funções cujos elementos representem a função objetivo. Estas funções formam a descrição matemática do chamado critério de desempenho, as quais usualmente são conflitantes. Assim, pode-se dizer que o termo otimizar significa descobrir uma solução para a qual os valores de todas as funções objetivo são considerados aceitáveis pelo projetista”.*

Formalmente, o problema pode ser descrito como:

$$\begin{gathered} minimizar\ \{f_1(x), f_2(x), \dots, f_k(x)\} \\ sujeito\ a \\ g_i(x) \leq 0, \qquad i = [1..p] \\ h_i(x) = 0, \qquad i = [1..q] \end{gathered} \tag{2.2}$$

onde $g_i(x)$ é uma restrição de desigualdade, $h_i(x)$ é uma restrição de igualdade e o vetor $x = [x_1, x_2, \dots, x_n]^T$ corresponde ao vetor de variáveis de decisão ou de otimização. Isto é, deseja-se determinar, dentro do conjunto $\boldsymbol{S}$ de todos os pontos que satisfaçam às restrições de igualdade e desigualdade, um conjunto particular de variáveis $[x_1^*, x_2^*, \dots, x_n^*]^T$ que permitam atingir os valores eficientes de todas as funções objetivo baseados em algum fator de decisão.

O conceito de eficiência foi introduzido por (Pareto 1896) e consiste em dizer que uma solução é ótima se não há outra solução factível que melhore um critério (objetivo) sem causar uma piora simultânea em pelo menos um outro. Matematicamente, pode-se dizer que um ponto x_i pertencente ao conjunto viável é Pareto-Ótimo se é possível afirmar que não existe:

$$f(x_j) \leq f(x_i), \qquad \forall x_j \in \boldsymbol{S}, f(x_i) \neq f(x_j) \tag{2.3}$$

Tal definição modifica o conceito de ótimo, já que agora não há apenas uma solução ótima, mas um conjunto de soluções conhecido como conjunto Pareto-ótimo.

1.5.1 O conjunto Pareto-ótimo

O conjunto Pareto-ótimo constitui a origem das pesquisas envolvendo Otimização Multiobjetivo e suas soluções são chamadas não-inferiores, não-dominadas ou eficientes.

Considere o MOP definido na equação 2.4.

$$F(x) = \begin{cases} F_1(x) = x^2 \\ F_2(x) = (x-2)^2 \end{cases} \tag{2.4}$$

Segundo o conceito de eficiência definido na seção anterior, é possível perceber que os pontos eficientes estão definidos no intervalo $x \in [0,2]$. Isso está representado na Figura 2-2, que mostra as funções objetivo e o espaço de objetivos, com as soluções eficientes em destaque.

Para os pontos $x = 2$ e $x = 3$, por exemplo, obtém-se $F(2) = \{4,0\}$ e $F(3) = \{9,1\}$. Como $f_1(2) < f_1(3)$ e $f_2(2) < f_2(3)$, pode-se dizer que o ponto $x = 2$ domina o ponto $x = 3$.

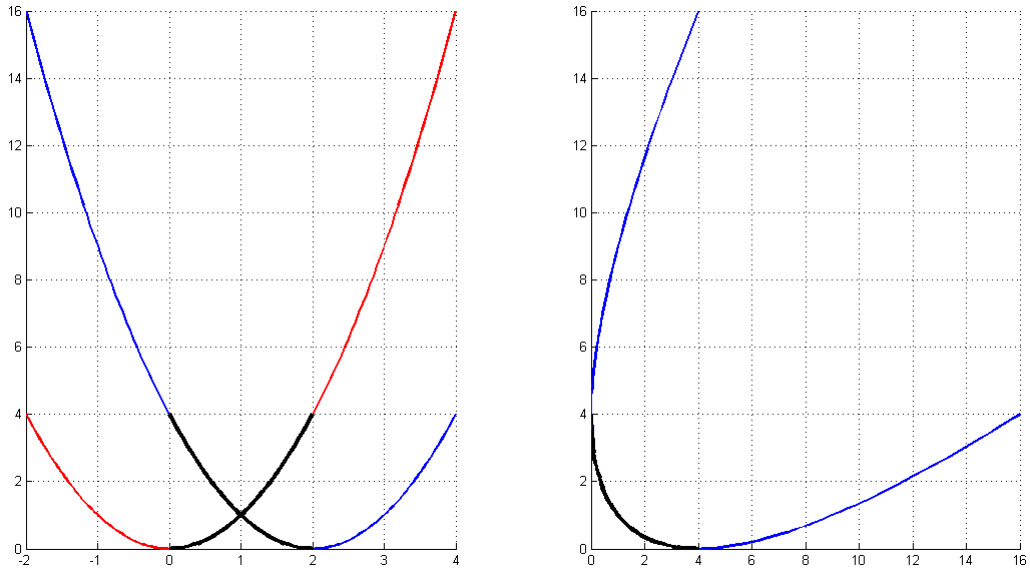

Figura 2-2 Funções de exemplo e fronteira Pareto

Como foi dito, essa nova modelagem requer novos métodos de resolução. A busca por um ponto ótimo foi transformada na busca por soluções eficientes, preferencialmente bem distribuídas ao longo da fronteira Pareto. Tais soluções devem ser repassadas a um decisor que irá, em uma etapa posterior, optar por uma.

1.5.2 Métodos de resolução

Na busca por soluções para problemas multiobjetivo, um fator importante é o número de pontos eficientes encontrados. Sob esse aspecto, os Algoritmos Evolucionários Multiobjetivo (*Multiobjective Evolutionary Algorithm* – MOEA) têm desempenhado um importante papel na resolução dos MOPs. Isso se deve ao fato de trabalharem com populações de indivíduos, em vez de apenas um ponto. Além disso,

muitos outros métodos exigem que os problemas sejam diferenciáveis ou convexos, o que não acontece com os MOEAs.

Embora haja um número crescente de MOEAs, muitos desses algoritmos acabam abordando problemas muito específicos ou não são robustos o suficiente para serem adotados em situações adversas. Devido a isso, alguns poucos permanecem como os grandes referenciais e são utilizados como *benchmark* para novos métodos propostos. Esse é o caso do NSGA-II (Deb, et al. 2000) e do SPEA-II (Zitzler, Laumanns and Thiele 2001), que ainda perduram como algoritmos de referência.

O NSGA-II e o SPEA2, bem como o MOEA/D (Zhang and Li 2007), serão apresentados a seguir.

1.5.2.1 SPEA e SPEA2

O Strength Pareto Evolutionary Algorithm 2 (SPEA2) foi proposto por (Zitzler, Laumanns and Thiele 2001) como uma evolução do SPEA (Zitzler and Thiele 1999). Na primeira versão, o SPEA utiliza uma população externa contendo os indivíduos eficientes encontrados até o momento e, a cada geração, essa população é atualizada. O cálculo da função de aptidão (*fitness*) de algum indivíduo leva em consideração a sua distância para a fronteira Pareto e a distribuição das soluções em um dado instante.

A diversidade das soluções encontradas pelo SPEA é avaliada quando a população externa atinge um número máximo de indivíduos, impossibilitando assim a inclusão de mais uma solução. Quando a população chega a este limite é aplicado um algoritmo chamado de *Clustering Method* também conhecido como Algoritmo de Agrupamento. Tal algoritmo tem como objetivo eliminar as soluções que excedem o valor limite das populações externas, contudo, sem prejudicar a diversidade da fronteira desta população.

Já o SPEA2 segue o princípio básico do seu antecessor, mas acrescenta três novas características (Coello 2006):

- Utilização de uma função de *fitness* baseada no número de indivíduos que dominam ou são dominados por outro;

- Cálculo de densidade baseado na distância dos vizinhos mais próximos;
- Método de seleção que preserva os indivíduos das extremidades da fronteira Pareto.

O SPEA2, com essas modificações, mostrou-se superior a seu antecessor em todos os testes realizados em (Zitzler, Laumanns and Thiele 2001), além de ser bastante competitivo com o NSGA-II.

1.5.2.2 NSGA e NSGA-II

O Nondominated Sorting Genetic Algorithm (NSGA) foi o primeiro algoritmo a ser publicado em uma revista especializada, a *Evolutionary Computation*. Ele foi proposto por (Srinivas and Deb 1994) e utiliza o conceito de classificação em camadas (*ranking*), descrito em (Goldberg 1989).

Em tal procedimento, primeiramente as soluções são separadas em dominadas e não-dominadas. É atribuído então um valor de *fitness* às soluções não-dominadas e o processo é repetido utilizando somente os indivíduos dominados, até que toda a população seja classificada. Tal procedimento está representado na Figura 2-3, que mostra as três primeiras fronteiras de uma dada população.

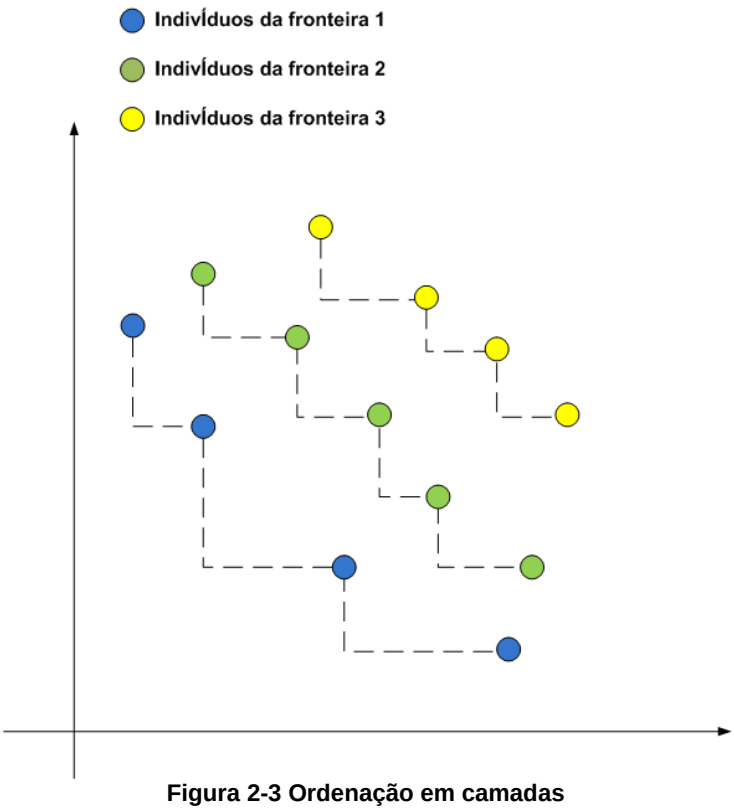

Figura 2-3 Ordenação em camadas

Vale ressaltar que os valores de *fitness* atribuídos aos indivíduos de uma camada são sempre maiores que os daqueles de camadas posteriores, de forma a garantir um número maior de cópias dos melhores indivíduos em cada geração. Entretanto, essa abordagem não é muito eficiente, uma vez que a classificação é repetida diversas vezes.

O NSGA-II (Deb, et al. 2000), por sua vez, utiliza uma abordagem denominada distância de multidão (*crowding distance*) para garantir a diversidade das soluções presentes na fronteira Pareto. Nesse método, é realizada uma estimativa de densidade de uma solução, através do cálculo do perímetro de um cuboide cujos vértices são os seus vizinhos mais próximos. Quanto maior o cubóide de uma determinada solução, mais distante ela se encontra das soluções vizinhas.

A utilização dessa métrica faz com que os indivíduos selecionados para a próxima geração sejam mais bem distribuídos ao longo da fronteira Pareto, o que evita aglomerações em algumas partes e 'buracos' em outras. Além disso, a fim de manter os pontos das extremidades da fronteira Pareto, é atribuído um valor infinito aos perímetros de seus cuboides, como pode ser visto na Figura 2-4.

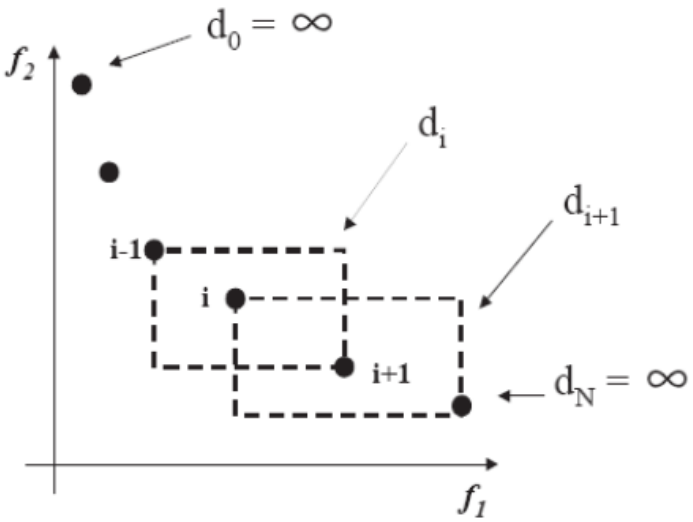

Figura 2-4 Cálculo da Distância de Multidão

O NSGA-II não utiliza uma população externa, como o SPEA2, mas agrega as melhores soluções da população atual (pais) e da população gerada (filhos), a fim de garantir o elitismo.

1.5.2.3 MOEA/D

Para um completo entendimento sobre o Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D), é preciso antes conhecer os conceitos de decomposição que ele utiliza. Tal decomposição consiste em transformar um MOP em subproblemas escalares e otimizá-los simultaneamente. A técnica utilizada pelo MOEA/D é a Decomposição de Tchebycheff (Miettinen 1999).

1.5.2.3.1 Decomposição de Tchebycheff

Nessa abordagem, os problemas escalares gerados assumem a forma

$$minimize\ G^{te}(X, \Lambda, Z^*): \max_{1 \le i \le m} \{\lambda_i | f_i(x) - z_i^*|\}, \qquad \forall\ x \in S \tag{2.5}$$

onde $\mathrm{Z}^* = \{\mathrm{z}_1^*, \mathrm{z}_2^*, \ldots, \mathrm{z}_\mathrm{m}^*\}$ é o ponto de referência (ponto utópico), isto é, $\mathrm{z}_\mathrm{i}^* = \min\{\mathrm{f}_\mathrm{i}(x), x \in \mathrm{S}\}$ para $i = 1, \ldots, m$. Para cada ponto ótimo x^* existe um vetor de pesos Λ tal que x^* é a solução ótima de (2.5) e cada solução ótima de (2.5) é uma solução Pareto-ótima do problema original (Zhang and Li 2007)[1].

Através da utilização de pesos diferentes, é possível encontrar pontos diferentes na fronteira Pareto. Baseado nisso, o MOEA/D busca otimizar diversos subproblemas paralelamente, cada um com um vetor de pesos distintos. Uma desvantagem dessa abordagem é que a função de agregação não é contínua ou diferenciável. Entretanto, o MOEA/D não utiliza derivadas, fazendo com que isso não seja um problema.

1.5.2.3.2 O algoritmo

O MOEA/D foi proposto por (Zhang and Li 2007) e, embora seja possível utilizar diversas abordagens de decomposição, seus autores obtiveram melhores resultados utilizando a abordagem Tchebycheff, descrita na seção anterior.

No MOEA/D, uma vizinhança do vetor de pesos λ_i é definida como os vetores de pesos mais próximos a ele. Como cada vetor de pesos define um subproblema diferente, pode-se dizer que os subproblemas vizinhos são aqueles com vetores de pesos mais

[1] Para fronteiras Pareto contínuas.

próximos. Essa definição é importante, pois esse algoritmo utiliza informações sobre os subproblemas vizinhos para buscar melhores soluções. A cada geração, a população do MOEA/D é formada pela melhor solução de cada subproblema até o momento.

Seguindo essa abordagem, subproblemas com vetores de pesos próximos tendem a ter soluções próximas. Portanto, existe a chance de que boas soluções de um subproblema sejam fortes candidatas para outros subproblemas. Além disso, o MOEA/D utiliza as soluções de subproblemas vizinhos para realizar cruzamento e mutação. Os passos do algoritmo serão descritos a seguir:

1. Criar uma população externa, inicialmente vazia.
2. Calcular a distância euclidiana entre todos os vetores de pesos e selecionar os mais próximo para a vizinhança de um dado vetor.
3. Gerar a população inicial.
4. Para cada subproblema, realizar os passos a seguir.
   4.1. Selecionar dois pontos da vizinhança e gerar uma nova solução utilizando operadores genéticos.
   4.2. Se a nova solução for melhor, atualizar os valores de Z^*.
   4.3. Para cada subproblema vizinho, verificar se a nova solução é melhor que a atual e, caso seja, substituí-la.
5. Atualizar a população externa, somente com pontos não-dominados.
6. Caso não seja encontrado um critério de parada, retornar ao passo 4.

Vale ressaltar que, embora o número de cálculos de função $\mathrm{F}(x)$ seja pequeno, é necessário realizar diversos cálculos para cada vizinho de cada subproblema. Isso torna o MOEA/D relativamente custoso computacionalmente, embora com ótimos resultados, como será visto na seção de testes, no Capítulo 4.

1.6 Evolução Diferencial

Evolução Diferencial (*Differential Evolution – DE*) é um algoritmo de otimização simples e eficiente que tem recebido cada vez mais destaque no âmbito da otimização não linear com variáveis contínuas. Ele foi proposto por (Storn and Price 1995) e ganhou destaque internacional nas competições do *International Contest on Evolutionary Computation – ICEC*, do IEEE, em 1996 (Storn and Price 1996) e 1997 (Price 1997).

Por seguir a linha dos algoritmos que evoluem uma população de soluções, o método de Evolução Diferencial é classificado como um algoritmo evolutivo, embora não tenha qualquer inspiração em um processo evolutivo natural. O seu operador de mutação, por exemplo, utiliza conceitos matemáticos e estratégias heurísticas.

1.6.1 Estrutura básica

O algoritmo de DE consiste em poucas etapas, seguindo a linha dos Algoritmos Evolucionários, que serão vistas nas próximas seções.

1.6.1.1 Inicialização

Antes de a população ser inicializada, é preciso estabelecer limites superior e inferior para cada parâmetro utilizado. Feito isso, a população inicial é gerada aleatoriamente dentro desse espaço de busca estabelecido. Sendo $rand$ um valor aleatório entre 0 e 1, a inicialização da $j-ésima$ variável do $i-ésimo$ indivíduo, por exemplo, pode ser descrita como:

$$x_i^j = L_{inf}^j + rand * (L_{sup}^j - L_{inf}^j) \quad (2.6)$$

Sendo os valores de $rand$ gerados de maneira uniformemente distribuída, a tendência é que o algoritmo realize uma varredura por todo o espaço de busca.

1.6.1.2 Mutação

Após a inicialização, o algoritmo DE realiza mutação e cruzamento sobre os indivíduos. Para a mutação, são selecionados três indivíduos diferentes da população e, em seguida é realizada a operação conhecida como mutação diferencial, descrita na Equação 2.7.

$$u_i = x_{r1} + \eta(x_{r3} - x_{r2}) \quad (2.7)$$

Nessa operação, é realizada uma perturbação em um indivíduo x_{r1}, denominado vetor base. Tal perturbação consiste em um vetor diferencial entre dois outros indivíduos x_{r2} e x_{r3}. Esse vetor é então multiplicado por um peso $\eta \in [0,1]$, que irá determinar o

tamanho do passo. A Figura 2-5 mostra em detalhes essa etapa do algoritmo, com a geração da nova solução u_i.

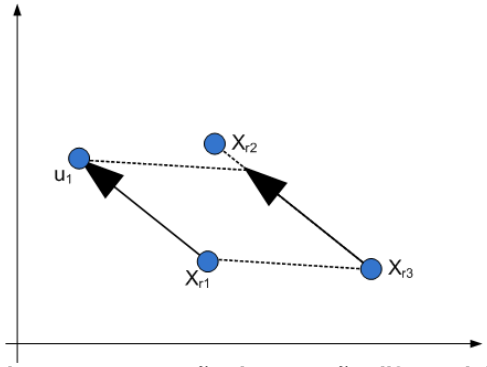

Figura 2-5 Operação de mutação diferencial

1.6.1.3 Cruzamento

Para completar o seu processo de busca, DE implementa o cruzamento. Conhecido também por recombinação discreta, tal processo consiste em criar um novo indivíduo a partir de dados copiados de outros dois. No caso da Evolução Diferencial, cada vetor da população atual é cruzado com um vetor mutante criado com o procedimento anterior. De maneira geral, esse procedimento pode ser descrito pela Equação 2.8.

$$u_i = \begin{cases} u_i, & se\ rand \leq C\ ou\ J = J_{rand} \\ x_i, & caso\ contrário \end{cases} \tag{2.8}$$

Nesse caso, $C \in [0,1]$ é a probabilidade de cruzamento e J é utilizado para garantir que pelo menos um parâmetro do vetor mutante será utilizado. A Figura 2-6 demonstra os possíveis resultados do cruzamento em um problema com apenas dois parâmetros.

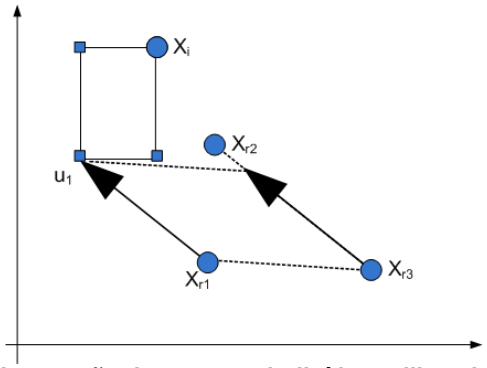

Figura 2-6 Resultado da geração de um novo indivíduo utilizando Evolução Diferencial

1.6.1.4 Seleção

Após a geração do novo indivíduo, o valor de sua função objetivo é calculado e, caso seja igual, ou melhor, a seu correspondente na população atual $\mathrm{x_i}$, esse é substituído. Caso contrário, o indivíduo atual é mantido, conforme a Equação 2.9.

$$x_i = \begin{cases} u_i, & f(u_i) \leq f(x_i) \\ x_i, & caso\ contrário \end{cases} \tag{2.9}$$

Uma vez que a nova população é criada, os processos de mutação, cruzamento e seleção são repetidos até que um critério de parada seja alcançado.

É realmente impressionante que um algoritmo tão simples apresente tantas características computacionais desejáveis, tais como robustez, versatilidade, eficiência e adaptabilidade (Guimarães 2009).

Embora haja diversas variações do algoritmo de Evolução Diferencial, no presente trabalho foi utilizada a versão original, descrita acima. No entanto, estudos mais detalhados sobre o tema podem ser encontrados em (Price, Storn and Lampinen 2005), (Chakraborty 2008) e (Guimarães 2009).

1.7 Sistemas Nebulosos

A fundamentação matemática dos sistemas nebulosos (*fuzzy systems*) parte da teoria dos conjuntos nebulosos (*fuzzy sets*), que pode ser entendida como uma extensão da teoria clássica dos conjuntos. Essa nova abordagem apresenta novos conceitos, notações e operações, que serão o objeto de estudo do presente capítulo.

1.7.1 Conjuntos nebulosos

Na teoria clássica dos conjuntos, existe um senso rígido de pertinência: um elemento simplesmente pertence ou não pertence a um conjunto. (Zadeh 1965) introduziu o conceito de conjuntos nebulosos, o que possibilitou atribuir certo grau de flexibilidade às funções de pertinência. O ponto de partida para conjuntos nebulosos é simplesmente expandir a avaliação da pertinência do par de números $\mathrm{A}\{0,1\}$ para todos os valores encontrados em $[0,1]$ (Tsoukalas and Uhrig 1996).

Sobremaneira, um conjunto pode ser considerado nebuloso quando sua definição é feita através de características subjetivas ou quando não há um limite rígido (*crisp*) para suas fronteiras. Esse conceito é altamente intuitivo e transparente, uma vez que retrata a maneira como o mundo real é percebido e descrito (Pedrycz and Gomide 2007).

A diferença entre tais conceitos pode ser vista na Figura 2-7, que mostra como o mesmo problema é representado por um conjunto *crisp* e um conjunto nebuloso. O problema em questão consiste em definir o conjunto de adolescentes, de acordo com a idade.

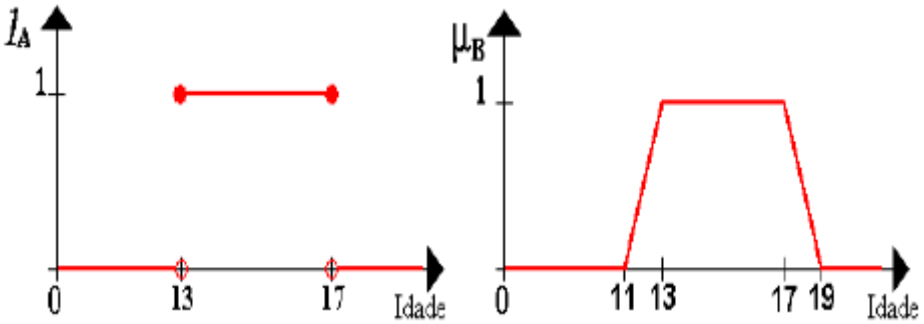

Figura 2-7 Representação de um conjunto crisp (a) e um conjunto nebuloso (b)

No primeiro exemplo, qualquer pessoa fora do intervalo $[13,17]$ seria totalmente excluída do conjunto. Já no modelo nebuloso, pessoas com idades entre $[11,13]$ ou $[17,19]$ fariam parte do conjunto, porém com grau de pertinência menor que aqueles do intervalo $[13,17]$.

Para definir o grau de pertinência de um determinado ponto, são utilizadas funções de pertinência, como será visto a seguir.

1.7.2 Funções de pertinência

Dado um conjunto nebuloso A, existe uma função $\mu_A(x)$, definida no intervalo $[0,1]$, associada a cada elemento x pertencente a um conjunto universo X. Essa função indica o grau de pertinência de x ao conjunto A, sendo que quanto mais próximo de 1, maior será a pertinência de x.

$$A = \{x/\mu_A(x)\}, \qquad \mu_A(x): X \rightarrow [0,1] \tag{2.10}$$

A partir disso, é possível criar a associação "x pertence ao conjunto A com grau de pertinência $\mu_A(x)$". Isso rompe os limites da lógica clássica, na qual um elemento simplesmente pertence ou não a um dado conjunto.

Obviamente, o conceito de conjunto nebuloso é apenas uma extensão da definição de um conjunto clássico. Se a função de pertinência $\mu_A(x)$ assumir apenas valores $\{0,1\}$, o conjunto A é reduzido a um conjunto clássico (Jang 1997).

Existem funções de pertinência das mais variadas formas, sendo que entre as mais comuns estão a triangular, a gaussiana e a trapezoidal, representadas na Figura 2-8.

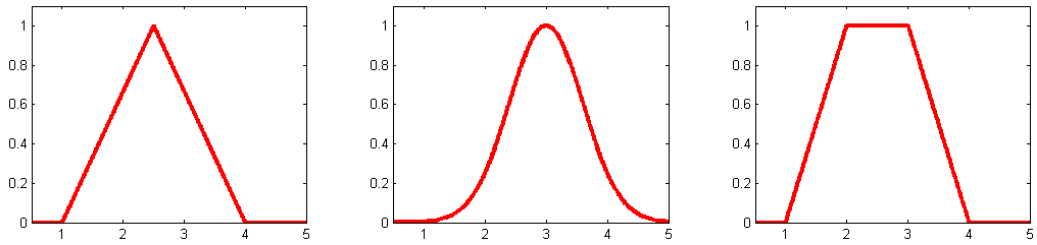

Figura 2-8 Funções de pertinência

A escolha da melhor função de pertinência a ser utilizada vai depender da natureza do problema abordado e não é possível dizer que uma é melhor que outra. É possível, ainda, definir novas funções de pertinência, de acordo com a necessidade.

1.7.3 Operadores nebulosos

Analogamente aos conjuntos clássicos, é possível realizar operações de interseção, união e negação, entre outras, nos conjuntos nebulosos. Em tais conjuntos, a interseção é implementada por operadores denominados de **t-normas** e a união é implementada por operadores denominados de **t-conormas** ou **s-normas** (Sandri and Correa 2001), que serão vistos a seguir.

1.7.3.1 T-normas

Uma t-norma é uma função $T: [0,1] \times [0,1] \rightarrow [0,1]$ que satisfaz os quatro axiomas a seguir:

$$T(x,1) = x \; e \; T(x,0) = 0, \qquad \forall x \in [0,1]$$

$$T(x1,x2) = T(x2,x1), \qquad \forall x \in [0,1]$$

$$T\big(x1,T(x2,x3)\big) = T(T(x1,x2),x3), \qquad \forall x \in [0,1]$$

$$Se\ x1 \leq x2, então\ T(x1,x3) \leq T(x2,x3), \qquad \forall x \in [0,1]$$

Embora qualquer função que atenda a esses axiomas possa ser considerada uma t-norma, algumas aparecem como as mais utilizadas. Elas estão descritas abaixo e os seus respectivos gráficos estão representados na Figura 2-9:

$$T(x1,x2) = min(x1,x2)$$

$$T(x1,x2) = x1.x2$$

$$T(x1,x2) = max(0, x1 + x2 - 1)$$

$$T(x1,x2) = \begin{cases} x2, & se\ x1 = 1 \\ x1, & se\ x2 = 1 \\ 0, & caso\ contrário \end{cases}$$

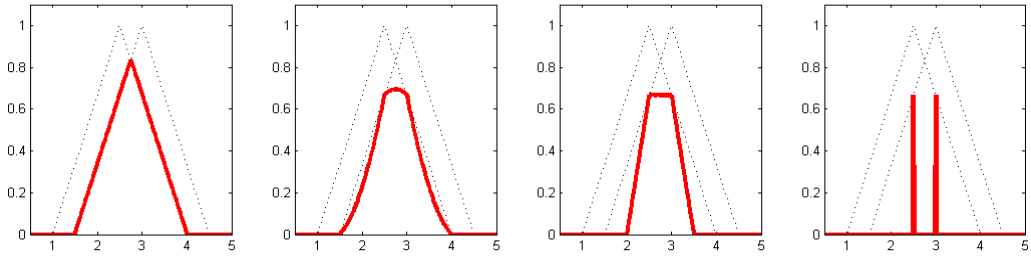

Figura 2-9 Exemplos de t-normas

1.7.3.2 S-normas

Uma s-norma é uma função $S: [0,1] \times [0,1] \rightarrow [0,1]$ que satisfaz os quatro axiomas a seguir:

$$S(x,0) = x\ e\ S(x,1) = 1, \qquad \forall x \in [0,1]$$

$$S(x1,x2) = S(x2,x1), \qquad \forall x \in [0,1]$$

$$S\big(x1, S(x2,x3)\big) = S(S(x1,x2),x3), \qquad \forall x \in [0,1]$$

$$Se\ x1 \leq x2, então\ S(x1,x3) \leq S(x2,x3), \qquad \forall x \in [0,1]$$

De forma similar às t-normas, qualquer função que atenda a esses axiomas pode ser considerada uma s-norma. Abaixo estão descritas algumas, juntamente com seus respectivos gráficos, representados na Figura 2-10:

$$S(x1,x2) = max(x1,x2)$$

$$S(x1,x2) = x1 + x2 - x1.x2$$

$$S(x1, x2) = min(1, x1 + x2)$$

$$S(x1, x2) = \begin{cases} x2, & se\ x1 = 0 \\ x1, & se\ x2 = 0 \\ 1, & caso\ contrário \end{cases}$$

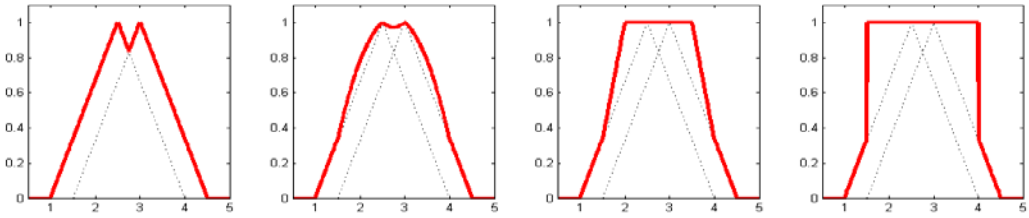

Figura 2-10 Exemplos de s-normas

Normalmente, a escolha da melhor t-norma ou s-norma a ser utilizada depende do problema abordado, embora, em geral, qualquer uma possa ser utilizada. Essa escolha é muito importante, pois o resultado de um sistema baseado em Lógica Nebulosa depende diretamente dos resultados de tais operadores.

Embora a área de Sistemas Nebulosos seja bastante vasta e com diversas aplicações, para o presente trabalho são utilizados apenas alguns dos conceitos citados acima, sem entrar, por exemplo, no campo da Lógica Nebulosa. No entanto, existem ótimas fontes sobre o tema disponíveis na literatura, para um estudo mais aprofundado, como (Jang 1997), (Pedrycz and Gomide 2007), (Cox 1994) e (Ross 2004).

3 O algoritmo proposto

Neste capítulo será apresentado o Fuzzy-DE (Fuzzy-based Differential Evolution Algorithm), que tem como principais objetivos ser simples e ter um baixo custo computacional. Para tal, esse MOEA utiliza conceitos de Sistemas Nebulosos e Evolução Diferencial, como será visto a seguir.

O Fuzzy-DE é uma evolução do FuGA, apresentado em (Lacerda and Caminhas 2009). Essa nova versão introduziu conceitos da Evolução Diferencial, além de algumas melhorias no tratamento de restrições e no cálculo do *fitness*.

1.8 Estrutura do algoritmo

O Fuzzy-DE pode ser resumido em poucos passos, que serão apresentados no pseudocódigo a seguir e na Figura 3-1. Há também uma seção sobre alguns pontos específicos do algoritmo, como será visto.

1. Gerar a população inicial, aleatoriamente
2. Calcular valores das funções objetivo e restrições
3. Gerar nova população utilizando Evolução Diferencial
4. Unir as populações
5. Calcular valores das funções objetivo e restrições
6. Separar indivíduos eficientes e não-eficientes
7. Executar os passos 7.1 a 7.4 para cada indivíduo não-eficiente
   7.1. Calcular a distância ponderada para o ponto eficiente mais próximo
   7.2. Calcular a densidade ponderada
   7.3. Calcular a violação ponderada das restrições
   7.4. Calcular o *fitness*, com base nos valores acima
8. Ordenar a população não-eficiente pelo *fitness*
9. Unir as populações eficiente e a não-eficiente ordenada
10. Excluir os indivíduos excedentes
11. Se não for atingido um critério de parada, retornar ao passo 3.
12. Retornar a população factível

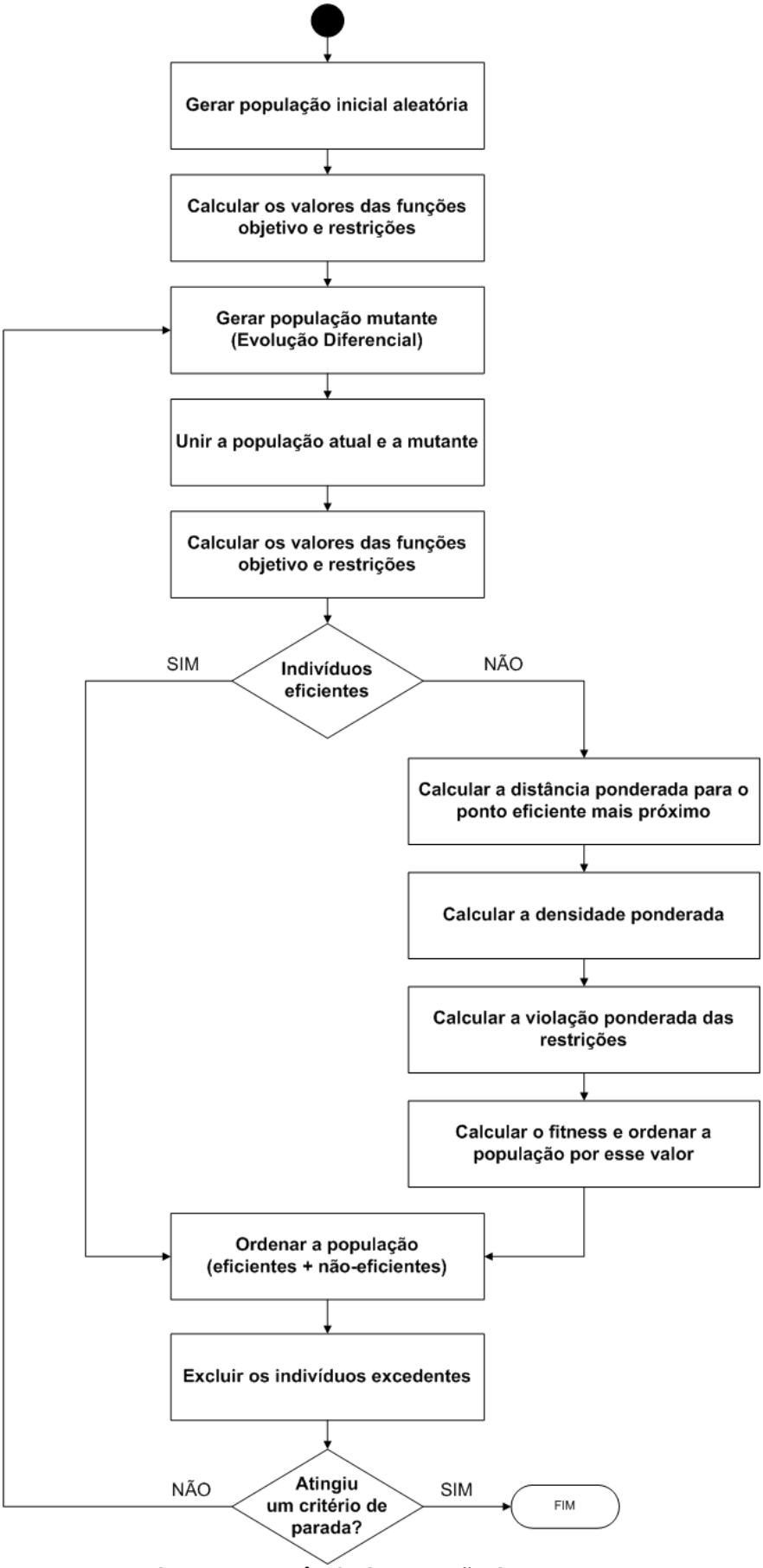

Figura 3-1 Sequência de execução do Fuzzy-DE

1.9 Especificidades do algoritmo

1.9.1 Geração da nova população (passo 3)

Para a geração da população mutante (passo 3), os pais são selecionados dinamicamente, conforme o pseudocódigo abaixo.

```
Se (t ≤ 0.2t_maximo ) ou (pop_factível < 5) ou (rand ≤ 0.2)
     Utilize toda a população
Senão
     Se (rand ≤ 0.8)
          Utilize somente a população factível
     Senão
          Utilize somente a população eficiente
     Fim
Fim
```

onde t representa o estágio atual da execução, em relação ao critério de parada t_{maximo} , $pop_{factível}$ representa o número de indivíduos factíveis na população atual e $rand \in [0,1]$.

É possível perceber que durante o estágio inicial da execução, todos os indivíduos podem ser selecionados, a fim de se realizar uma busca global. Após essa fase, procura-se priorizar a seleção de indivíduos factíveis (que não violam restrições), desde que seu número seja suficiente para tal. Tal procedimento tem como objetivo refinar as buscas e gerar melhores soluções. Caso o problema seja irrestrito, o funcionamento é análogo, porém os pais são selecionados dentre os indivíduos eficientes. Após a seleção dos pais, os novos indivíduos são gerados utilizando o algoritmo de Evolução Diferencial básico, visto anteriormente.

1.9.2 Cálculo da distância ponderada (passo 7.1)

A distância ponderada de cada ponto i é dada pela equação abaixo.

$$D1_i^{ponderada} = 1 - \frac{D1_i - D1_{minimo}}{D1_{maximo} - D1_{minimo}} \quad (3.1)$$

onde $D1_{\text{maximo}}$ e $D1_{\text{minimo}}$ são, respectivamente, a máxima e a mínima distâncias entre um ponto não-eficiente da população atual e o ponto eficiente mais próximo. Com

isso, os pontos eficientes recebem valor 1 e aquele mais distante da fronteira Pareto atual recebe valor 0. Os demais recebem valores relativos nesse intervalo.

1.9.3 Cálculo da densidade ponderada (passo 7.2)

Para o cálculo da densidade ponderada, o procedimento é parecido. Nesse caso, o valor 1 é atribuído aos pontos com menor densidade.

$$D2_i^{ponderada} = 1 - \frac{D2_i - D2_{minimo}}{D2_{maximo} - D2_{minimo}} \tag{3.2}$$

Embora qualquer método para cálculo de densidade possa ser utilizado, no Fuzzy-DE, optou-se pela Distância de Multidão (*Crowding Distance*), como utilizado no NSGA-II, apresentado anteriormente.

1.9.4 Cálculo da violação ponderada (passo 7.3)

Em problemas com restrições, é necessário calcular a violação ponderada de cada ponto i para cada restrição j. Para tal, utiliza-se um procedimento também similar aos anteriores, com a diferença de que todos os pontos factíveis recebem valor 1.

$$V_{i,j}^{ponderada} \begin{cases} 1, & se\ factível \\ 1 - \dfrac{V_{i,j} - V_{minimo\ ,j}}{V_{maximo\ ,j} - V_{minimo\ ,j}}, & caso\ contrário \end{cases} \tag{3.3}$$

De posse dos valores ponderados para cada restrição, o valor final da violação ponderada é dado por uma t-norma entre todas as restrições, a saber:

$$V_i^{ponderada} = \boldsymbol{T}\left(V_{i,j}\right), \quad j = [1..q] \tag{3.4}$$

Qualquer t-norma pode ser utilizada e, para os testes, foi utilizado o operador de mínimo. Com isso, o valor final é representado pela restrição mais violada por um dado ponto.

1.9.5 Cálculo do *fitness* (passo 7.4)

Após o cálculo da distância, da densidade e da violação ponderadas, é possível calcular o *fitness* de cada indivíduo, que é dividido em duas etapas.

Na primeira etapa, é realizada uma ponderação entre a distância e a densidade. Essa ponderação é uma s-norma convencional, acrescida de um peso α, que serve para dar maior prioridade a um dos termos, conforme a equação abaixo.

$$Fit_i = \boldsymbol{S}(\alpha D1_i, (1-\alpha)D2_i), \qquad \forall \alpha \in [0,1]$$

O valor de α é dinâmico, de forma a priorizar, em uma primeira etapa, a diversidade das soluções e, posteriormente, a qualidade das soluções. Novamente, o objetivo de tal procedimento é iniciar com uma busca global e terminar com uma busca mais direcionada ao final da execução. O valor de α é dado por $\alpha = 0.3 + 0.7(t/t_{maximo})$. Apenas para fins demonstrativos, foram analisados cenários com diferentes valores de α, para um mesmo problema. A Figura 3-2 representa a população final para cada cenário, nos quais α recebeu os valores 0 (a), 0.2 (b), 0.4 (c), 0.6 (d), 0.8 (e) e 1.0 (f). É possível perceber que o espalhamento nos primeiros casos é muito maior, devido ao maior peso da densidade para a seleção dos indivíduos. Já nos últimos cenários, a prioridade é a distância dos indivíduos para o ponto eficiente mais próximo, em detrimento da diversidade de soluções.

Já a segunda etapa do cálculo do *fitness* é utilizada apenas para problemas restritos. Em tais casos, é necessário calcular uma nova s-norma entre o valor de *fitness* atual e a violação ponderada das restrições, obtido no passo 7.3 do algoritmo:

$$Fit_i = \boldsymbol{S}(Fit_i, V_i^{ponderada})$$

De forma similar ao primeiro passo, é possível utilizar um peso para priorizar uma das partes, conforme necessidade do problema.

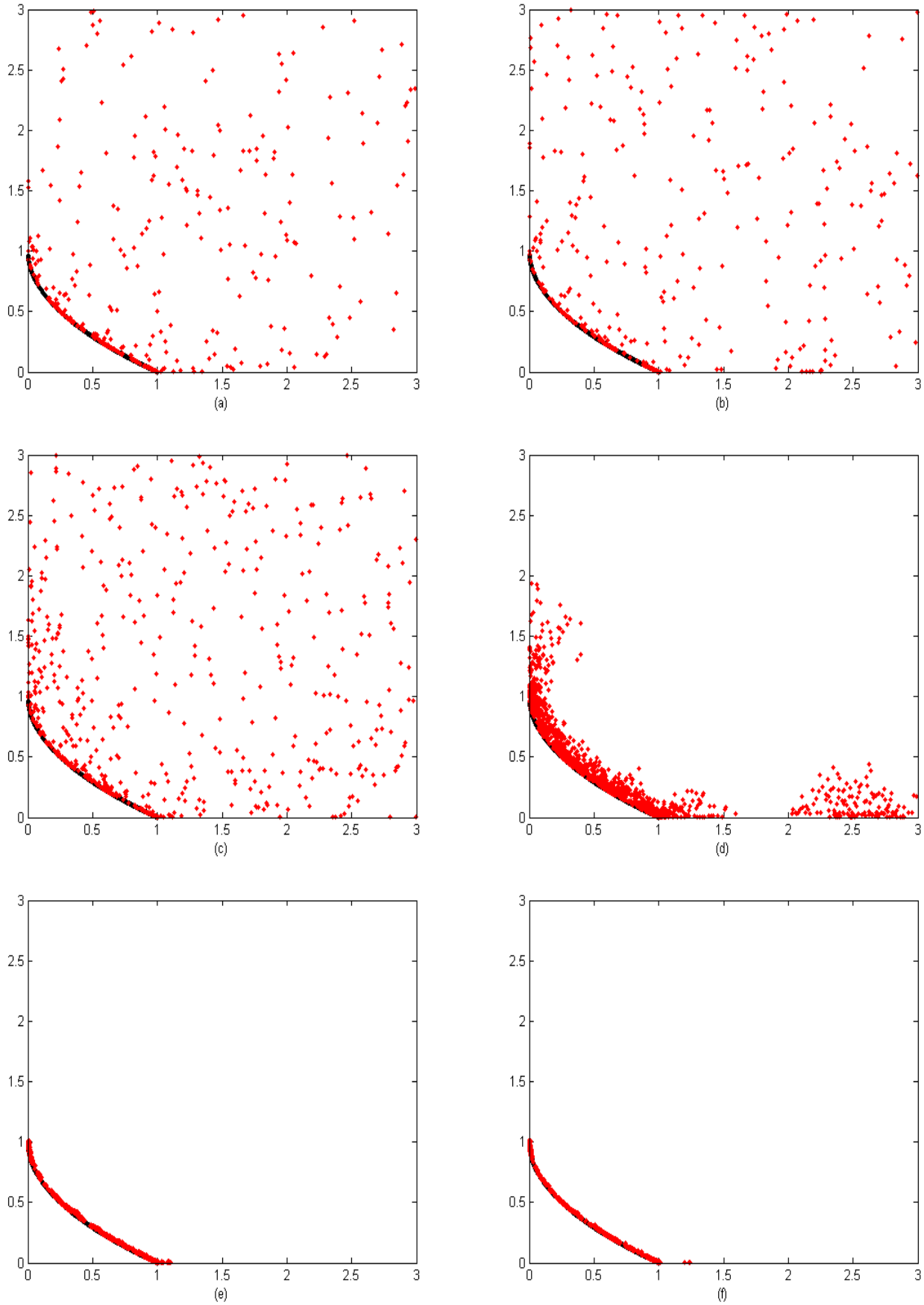

Figura 3-2 Sensibilidade da seleção ao valor do peso

1.9.6 Seleção dos indivíduos (passos 8, 9 e 10)

Após todos esses passos, é necessário selecionar quais indivíduos passarão à próxima geração. Para tal, são selecionados, primeiramente, os indivíduos eficientes e, em seguida, os não-eficientes ordenados pelo *fitness*. Caso o número de indivíduos eficientes seja maior que o tamanho da população, eles deverão ser ordenados pela densidade, analogamente ao passo 7.2.

Para problemas restritos, todos os indivíduos factíveis não-duplicados são armazenados em uma população externa, a fim de garantir que os mesmos permaneçam até o final da execução do algoritmo. Entretanto, isso é apenas uma garantia extra, já que nos testes realizados o número de indivíduos factíveis não foi superior o tamanho da população em nenhum caso.

Após essa ordenação, os indivíduos excedentes são excluídos e o algoritmo passa à próxima geração ou termina a sua execução, caso um critério de parada seja alcançado.

Embora possua uma estrutura bastante simples, o Fuzzy-DE apresentou bons resultados durante os testes e mostrou-se competitivo quando comparado a outros algoritmos. Detalhes sobre os testes serão apresentados a seguir.

1.10 Parâmetros utilizados

Parte dos parâmetros utilizados no Fuzzy-DE são dinâmicos e, em geral, não precisam ser ajustados. Tais parâmetros são utilizados para refinar o processo de busca, como mostrado para o passo 6 do algoritmo. Em geral, os parâmetros que precisam de intervenção do usuário são:

- ✓ Tamanho a população: número de indivíduos em cada geração.
- ✓ Critério de parada: número de cálculos de função ou tempo de processamento limite para o algoritmo.

Outros parâmetros passíveis de intervenção, caso o usuário queira um processo ainda mais específico, são

- ✓ Peso α para o cálculo do *fitness*, utilizado no passo 7.4.

- ✓ Peso β também para o cálculo do *fitness*, porém somente para problemas com restrição.
- ✓ Valor η utilizado na Evolução Diferencial para limitar o tamanho do passo.

Entretanto, não foi realizada uma análise da sensibilidade do algoritmo a tais parâmetros.

4 Testes realizados

Antes de avaliar o algoritmo proposto, é necessário definir quais métricas de desempenho serão utilizadas para compará-lo aos algoritmos de referência. A seguir serão apresentadas algumas amplamente utilizadas para avaliação de MOEAs.

1.11 Métricas de desempenho

As métricas apresentadas em (Van Veldhuizen 1999) e (Zitzler, Deb and Thiele 2000) representam um ótimo *benchmark* para MOEAs, principalmente por avaliarem fatores que vão além da distância para a fronteira Pareto real, como será visto.

1.11.1 Cardinalidade

A cardinalidade representa o número de soluções não-dominadas na população final e é especialmente importante para se definir um tamanho mínimo para a fronteira Pareto. Isso se deve, principalmente, a possíveis resultados inconsistentes de outras métricas, quando o número de indivíduos é pequeno.

Além disso, quanto maior número de soluções eficientes, mais opções o decisor terá ao escolher a mais adequada, em uma etapa posterior.

1.11.2 Hipervolume

A soma dos hipervolumes dos hipercubos formados por um ponto de referência e cada ponto da fronteira Pareto ajudam a mensurar a sua distância para a fronteira Pareto real. Para problemas de minimização, maiores hipervolumes implicam em melhores soluções.

Entretanto, tal métrica pode ser utilizada também para medir a cobertura do algoritmo sobre a fronteira Pareto real. Isso é especialmente útil em situações nas quais o algoritmo avaliado gera soluções apenas em regiões específicas do espaço. Nesses casos, maiores hipervolumes podem representar um melhor espalhamento das soluções sobre a fronteira Pareto.

1.11.3 Espalhamento

Um dos principais fatores para se definir a qualidade de um dado conjunto de soluções é a uniformidade do seu espalhamento. Para medir tal espalhamento, é utilizado o desvio padrão da distância de um ponto para o seu vizinho mais próximo. Novamente, menores valores são melhores, já que indicam uma menor variação de tais distâncias.

1.11.4 Indivíduos dominados

Essa métrica retorna apenas o número de indivíduos da população que são dominados por pontos do algoritmo de referência a ser comparado. Normalmente são utilizados valores proporcionais ao tamanho da população, geralmente sob a forma de um percentual.

1.11.5 Indivíduos dominantes

Similar à métrica anterior, tal métrica retorna o número de indivíduos do algoritmo de referência que são dominados por pontos do algoritmo analisado. Aqui, maiores valores indicam uma maior qualidade do algoritmo.

1.11.6 Tempo

Obter resultados satisfatórios em tempo hábil é de vital importância para diversas aplicações de MOEAs e, até mesmo, de outros algoritmos. Essa métrica, portanto, mede o tempo total de processamento de cada algoritmo, normalmente com valores relativos ao algoritmo de referência.

1.11.7 Distância geracional

Essa métrica só pode ser utilizada em problemas cujas fronteiras Pareto reais são conhecidas. Isso ocorre porque a distância geracional é o somatório das distâncias entre os pontos fronteira Pareto obtida e o ponto mais próximo da fronteira Pareto real. Intuitivamente, menores distâncias indicam melhores soluções.

1.11.8 Distância geracional invertida

Além das métricas citadas, há também a distância geracional invertida (*Inverted Generational Distance - IGD*), que representa o somatório das distâncias de cada ponto

da fronteira Pareto real para o ponto gerado mais próximo. Tal métrica é especialmente útil nos casos em que os pontos gerados são boas soluções (próximas à fronteira Pareto real), porém com presença de 'buracos', como representado na Figura 4-1.

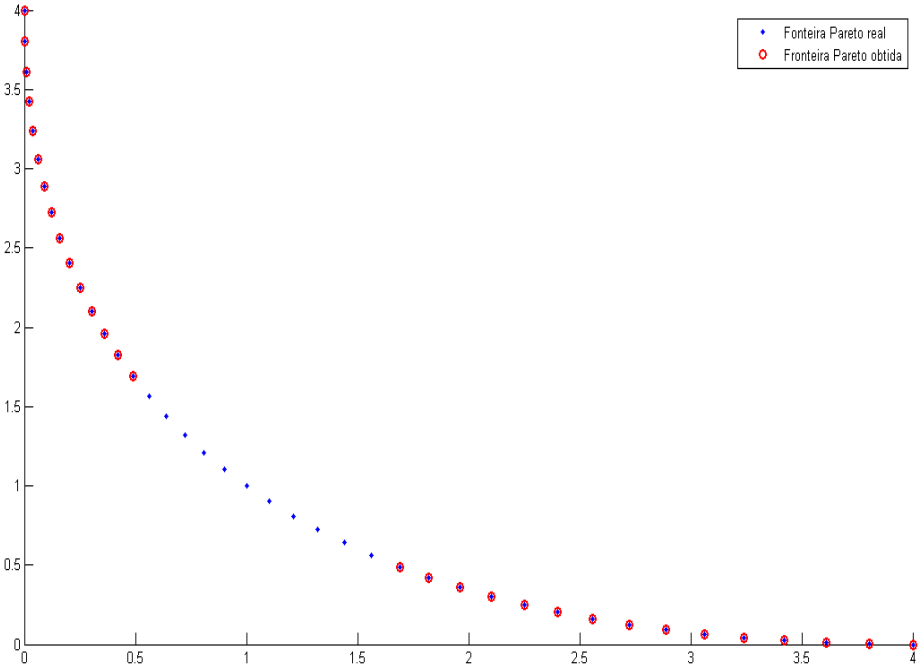

Figura 4-1 Fronteiras Pareto real e obtida

Em tal situação, a utilização da distância geracional não é a mais indicada, pois a distância de cada ponto para a fronteira Pareto real é nula. Já o IGD retornaria um valor diferente de zero, já que nem todos os pontos da fronteira Pareto real estão cobertos por um ponto obtido pelo algoritmo avaliado.

De posse de todas essas métricas, o próximo passo foi definir quais problemas deverão ser tratados, já que estes devem ser reconhecidamente complexos, abrangentes e bem difundidos. Para demonstrar o desempenho do algoritmo, após alguma análise, optou-se pelos problemas da competição do *Congress of Evolutionary Computation* de 2009 (CEC09) e pelo problema TEAM22, do Compumag.

1.12 Competição do CEC 2009

O *Congress of Evolutionary Computation* é um evento anual promovido pela *IEEE Computational Intelligence Society* e *Evolutionary Programming Society*. A cada edição, é

realizada uma competição envolvendo um tema específico e, em 2009, o foco foi trabalhar com problemas multiobjetivo utilizando algoritmos evolucionários.

Os testes foram propostos por (Zhang, Zhou, et al. 2009) e envolvem problemas restritos e irrestritos, em categorias distintas. Cada algoritmo poderia concorrer em cada uma isoladamente e, por isso, foram consagrados dois vencedores: o MOEA/D para problemas irrestritos e o DMOEA-DD, para os restritos. Devido à falta de acesso a uma versão funcional dos códigos-fonte do DMOEA-DD, foram abordados apenas os problemas irrestritos, já que não seria possível uma comparação detalhada entre os métodos.

Os problemas foram nomeados de UF1 a UF7 e serão apresentados a seguir. Embora tenham sido utilizados espaços de busca com três dimensões para exibição dos gráficos, durantes os testes foram utilizados sempre espaços com trinta dimensões, seguindo as regras do evento. Além disso, a métrica utilizada no evento foi o IGD, apresentado anteriormente. Por isso, optou-se por utilizar os mesmos cenários. Para todos os problemas, o critério de parada são 300 mil cálculos de cada função objetivo.

Os problemas a seguir podem ser encontrados em (Zhang, Zhou, et al. 2009).

1.12.1 Problema UF1

$$f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left[x_j - \sin 6\pi x_1 + \frac{j\pi}{n} \right]^2$$

$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left[x_j - \sin 6\pi x_1 + \frac{j\pi}{n} \right]^2$$

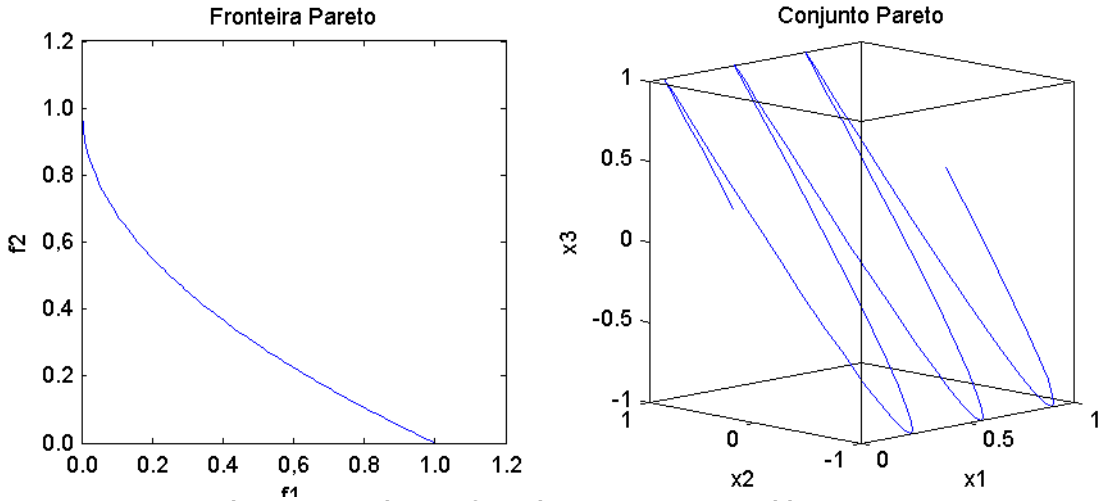

Figura 4-2 Conjunto e fronteira Pareto para o problema UF1

1.12.2 Problema UF2

$$f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} y_j^2$$

$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} y_j^2$$

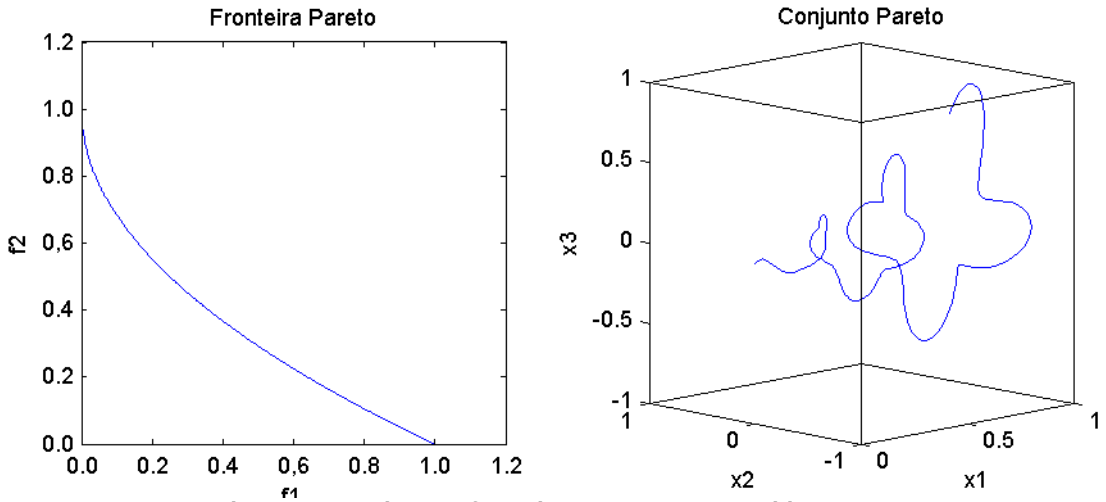

Figura 4-3 Conjunto e fronteira Pareto para o problema UF2

1.12.3 Problema UF3

$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j \in J_1}\left(4\sum_{j \in J_1} y_j^2 - 2\prod_{j \in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$$

$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j \in J_2}\left(4\sum_{j \in J_2} y_j^2 - 2\prod_{j \in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$$

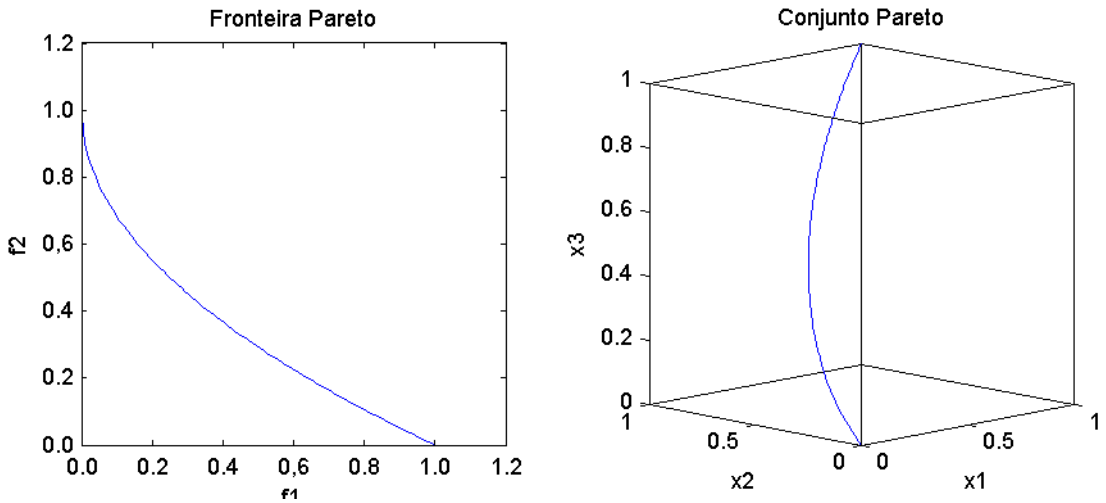

Figura 4-4 Conjunto e fronteira Pareto para o problema UF3

1.12.4 Problema UF4

$$f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} h(y_j)$$

$$f_2 = 1 - x_1^2 + \frac{2}{|J_2|} \sum_{j \in J_2} h(y_j)$$

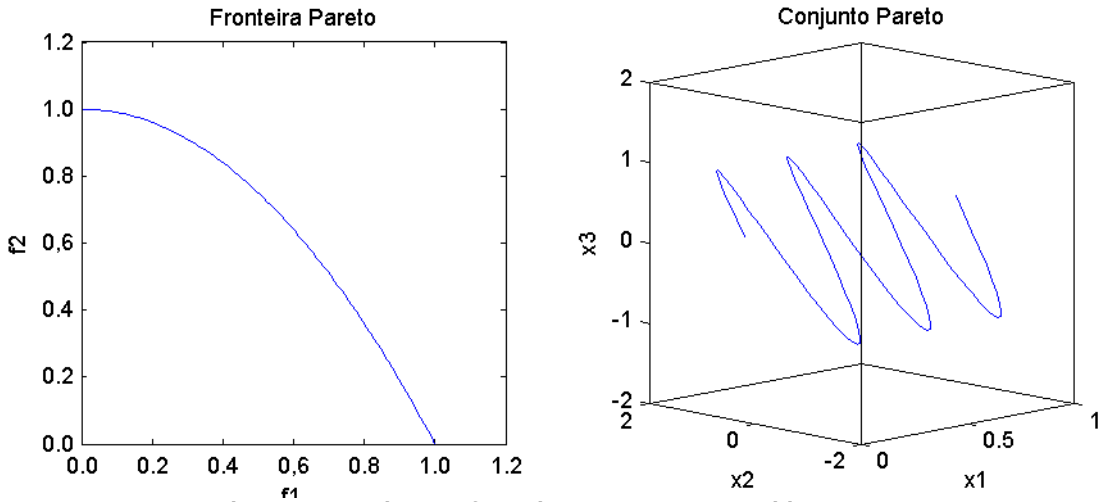

Figura 4-5 Conjunto e fronteira Pareto para o problema UF4

1.12.5 Problema UF5

$$f_1 = x_1 + \left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)| + \frac{2}{|J_1|}\sum_{j \in J_1} h(y_j)$$

$$f_2 = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)| + \frac{2}{|J_2|}\sum_{j \in J_2} h(y_j)$$

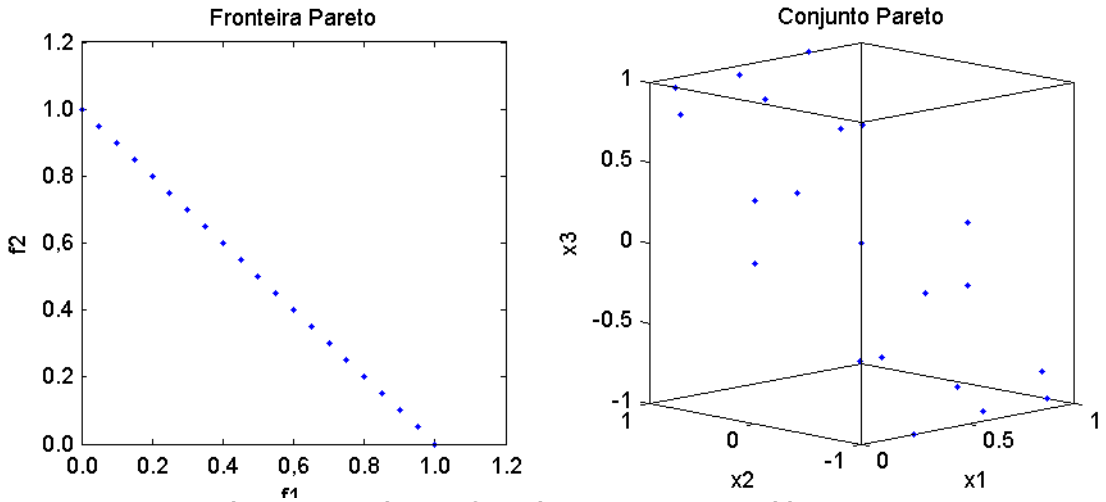

Figura 4-6 Conjunto e fronteira Pareto para o problema UF5

1.12.6 Problema UF6

$$f_1 = x_1 + max\left\{0{,}2\left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)|\right\} + \frac{2}{|J_1|}\left(4\sum_{j \in J_1} y_j^2 - 2\prod_{j \in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$$

$$f_2 = 1 - x_1 + max\left\{0{,}2\left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)|\right\} + \frac{2}{|J_2|}\left(4\sum_{j \in J_2} y_j^2 - 2\prod_{j \in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$$

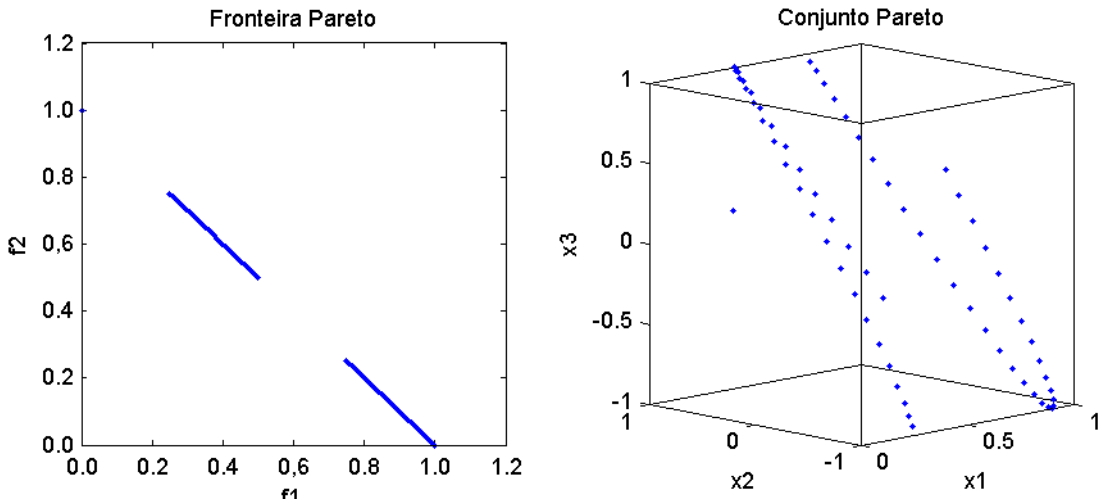

Figura 4-7 Conjunto e fronteira Pareto para o problema UF6

1.12.7 Problema UF7

$$f_1 = \sqrt[5]{x_1} + \frac{2}{|J_1|} \sum_{j \in J_1} y_j^2$$

$$f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} y_j^2$$

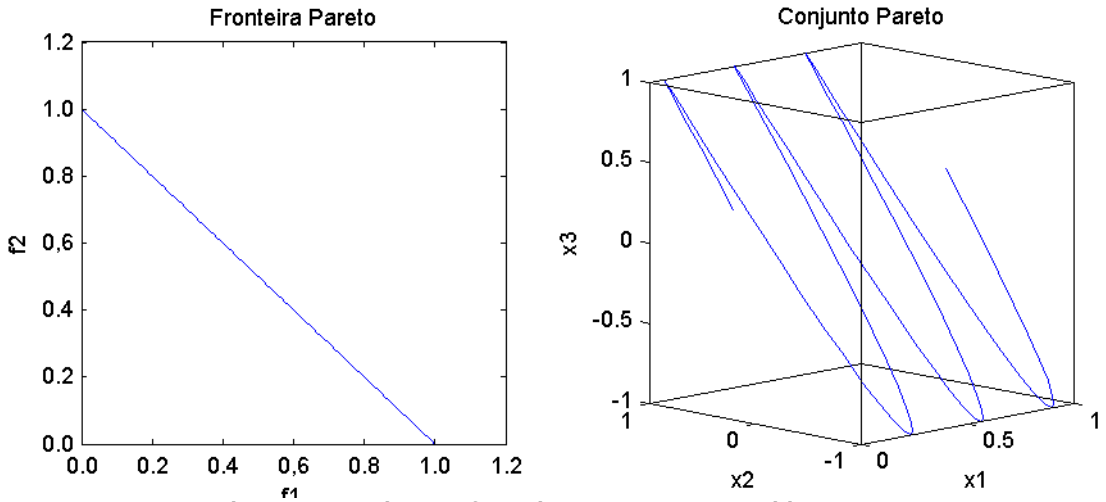

Figura 4-8 Conjunto e fronteira Pareto para o problema UF7

1.13 Problema TEAM22

Este problema teste consiste na otimização de um sistema de armazenamento de energia em forma magnética utilizando uma bobina supercondutora (Superconducting Magnetic Energy Storage - SMES). Ele foi construído e padronizado para ser utilizado como problema teste na comparação de algoritmos de otimização.

Os objetivos da otimização são fazer o SMES apto a armazenar uma dada quantidade de energia e minimizar o campo de dispersão.

Para garantir a supercondutividade, o fluxo magnético não pode superar certos limites, os quais dependem da densidade de corrente que passa pela bobina.

1.13.1 Geometria do problema

O SMES é composto de duas bobinas supercondutoras concêntricas em forma de anéis de seção retangular, nas quais passam correntes de sentidos opostos, com densidades de corrente J_1 e J_2, como mostra a Figura 4-9. Os anéis são parametrizados pelos seus raios (R_1 e R_2), e pelas alturas (h_1 e h_2) e larguras (d_1 e d_2) de suas seções.

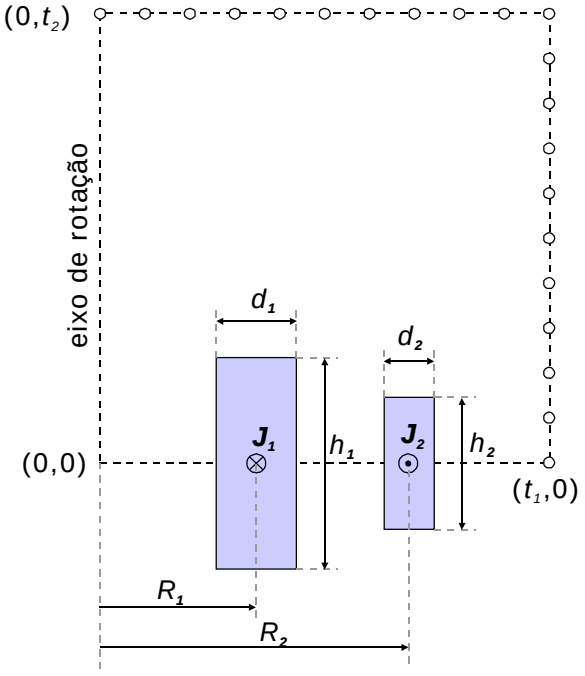

Figura 4-9 Geometria do SMES

Como pode ser observado na Figura 4-9, o problema possui simetria em relação ao eixo de rotação e também em relação ao plano que corta os centros das seções das bobinas. O fluxo de dispersão é amostrado em pontos sobre um cilindro de raio t_1 e altura $2t_2$, sendo que o ponto (t_1, t_2) se repete.

1.13.2 Padronização do TEAM22

Para que diferentes resultados possam ser comparados, foram definidas todas as características do SMES, além das funções objetivo e restrições para otimização. A densidade de fluxo magnético B_c induzida nas bobinas supercondutoras pelas quais passam uma corrente com densidade J_c, deve satisfazer à restrição

$$|J_C| \leq -6.4 * |B_C| + 54 \tag{4.1}$$

na qual, considerando a igualdade, corresponde a uma aproximação da curva crítica do supercondutor *NbTi* fabricado industrialmente. Os valores do campo máximo induzido nas bobinas acontecem nas suas interfaces com o meio envolvente e no plano que passa pelos centros das seções delas. Assim, $\mathrm{B_c}$ deve ser avaliado apenas em quatro pontos $(R_1 \pm \mathrm{d}_1, 0)$ e $(R_2 \pm \mathrm{d}_2, 0)$, ao invés de ao longo de toda a bobina.

O SMES pode ser abordado como um problema mono ou multiobjetivo. No primeiro caso, há uma junção das funções objetivo do modelo multiobjetivo, que serão apresentadas a seguir.

$$f_1 = \frac{\left|E - E_{ref}\right|}{E_{ref}}$$
$$f_2 = \frac{B_d^2}{B_{norm}^2}, \qquad B_d^2 = \frac{1}{N_a}\sum_{i=1}^{N_a}\left|B_{disp}^i\right|^2 \tag{4.2}$$

onde B_{d} é a média quadrática do módulo da densidade de fluxo magnético amostrada, B_{norm} é uma constante de normalização, E é a energia armazenada nas duas bobinas, $\mathrm{E_{ref}}$ é a energia que se deseja armazenar no SMES, B_{disp} é a densidade de fluxo amostrada e $\mathrm{N_a}$ o número de amostras.

As amostras são feitas em 11 pontos uniformemente distribuídos sobre a linha correspondente ao raio, mais 11 pontos uniformemente distribuídos sobre a linha correspondente à altura (ver Figura 4-9), lembrando que o ponto de interseção das duas linhas é repetido. Assim $\mathrm{N_a} = 22$.

Foram criadas duas versões para o problema: uma com três parâmetros (ver Tabela 4.1) e outra com oito (ver Tabela 4.2). Na primeira versão, vários parâmetros foram fixados, a fim de diminuir a sua complexidade. Já na versão completa, apenas $\mathrm{E_{ref}}$ e B_{norm} possuem valores fixos, mas não são parâmetros de entrada.

Tabela 4.1 Configuração do SMES para 03 parâmetros

	R_1	$h_1/2$	d_1	R_2	$h_2/2$	d_2	J_1	J_2	B_{norm}	E_{ref}
Min	--	--	--	2.6	0.204	0.1	--	--	--	--
Max	--	--	--	3.4	1.1	0.4	--	--	--	--
Fixo	2.0	0.8	0.27	--	--	--	22.5	22.5	0.003	180

Tabela 4.2 Configuração do SMES para 08 parâmetros

	R_1	$h_1/2$	d_1	R_2	$h_2/2$	d_2	J_1	J_2	B_{norm}	E_{ref}
Min	1,0	0,1	0,1	1,8	0,1	0,1	10	10	--	--
Max	4,0	1,8	0,8	5,0	1,8	0,8	30	30	--	--
Fixo	--	--	--	--	--	--	--	--	0.0002	180

A partir de tais dados é possível realizar os testes e comparar o algoritmo avaliado aos já existentes na literatura.

1.14 Resultados

De posse dos problemas e das métricas, passou-se para a etapa de execução dos testes. Seguindo as regras do CEC 2009, o Fuzzy-DE e o NSGA-II foram executados 30 vezes e o melhor resultado de cada um foi utilizado para a comparação. Já para o problema TEAM22, foram utilizados resultados oficiais conhecidos na literatura. Um resumo sobre as comparações será visto a seguir.

1.14.1 Competição do CEC 2009

Segundo os resultados oficiais do CEC 2009 disponíveis em (Zhang, Zhou, et al. 2009), o MOEA/D consagrou-se o vencedor da competição envolvendo somente problemas irrestritos. Portanto foi escolhido como referência para os testes. Além disso, buscou-se comparar o Fuzzy-DE com um algoritmo multiobjetivo de propósito geral, o NSGA-II.

Para evitar possíveis erros que pudessem comprometer os resultados dos testes, foi utilizada uma versão do NSGA-II, implementada em Matlab, disponível na Internet. O mesmo foi feito para o MOEA/D, cujo código foi publicado no *site* da competição. Foram realizadas então duas abordagens: a primeira utilizou apenas o IGD para comparação e a segunda utilizou as demais métricas apresentadas anteriormente.

1.14.1.1 Análise do IGD

Utilizando-se os códigos-fonte de avaliação disponibilizados pela organização do CEC 2009, o Fuzzy-DE e o NSGA-II foram executados 30 vezes e o melhor resultado para cada problema abordado foi destacado. Já para o MOEA/D, foram utilizados os resultados oficiais do evento. A Tabela 4.3 apresenta a comparação entre os métodos, utilizando apenas o IGD como métrica. Já a Tabela 4.4 apresenta os mesmos dados normalizados, a fim de se obter uma melhor comparação.

Como o IGD é uma medida de distância para a fronteira Pareto real, resultados menores são melhores. É possível perceber, então, que o NSGA-II obteve piores resultados em todos os testes e, em geral, o MOEA/D ficou com os melhores resultados.

Tabela 4.3 Resultado do IGD para os problemas do CEC 2009

	UF1	UF2	UF3	UF4	UF5	UF6	UF7
MOEA/D	0,00435	0,00679	0,00742	0,06385	0,18071	0,00587	0,00444
Fuzzy-DE	0,01090	0,01510	0,01220	0,04730	0,21960	0,07120	0,00730
NSGA-II	0,03820	0,02230	0,12200	0,06610	1,47460	0,09740	0,01770

Tabela 4.4 Resultado normalizado do IGD para os problemas do CEC 2009

	UF1	UF2	UF3	UF4	UF5	UF6	UF7
MOEA/D	0,11	0,30	0,06	0,97	0,12	0,06	0,25
Fuzzy-DE	0,29	0,68	0,10	0,72	0,15	0,73	0,41
NSGA-II	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Quando comparado aos demais algoritmos que participaram do CEC 2009, o Fuzzy-DE obteve bons resultados. A Tabela 4.5 mostra a classificação dos 13 algoritmos participantes, além do Fuzzy-DE e do NSGA-II, para cada problema.

Tabela 4.5 Classificação dos algoritmos para os problemas do CEC 2009

	UF1	UF2	UF3	UF4	UF5	UF6	UF7	MÉDIA
MOEA/D	1	3	1	14	7	1	1	4
MTS	4	1	6	1	1	2	13	4
DMOEA-DD	6	4	4	5	11	3	5	5
Fuzzy-DE	7	9	2	9	8	4	2	6
LiuLi	5	7	3	7	4	11	3	6
GDE3	2	6	13	2	2	13	11	7
MOEA/D-GM	3	2	5	10	15	15	4	8
AMGA	11	10	8	4	3	10	14	9
MOEP	13	11	10	6	9	8	7	9
DECMOSA-SQP	14	14	9	3	5	9	10	9
ClusteringMOEA	10	13	7	13	10	6	9	10
NSGA-II	12	12	14	15	14	7	6	11
NSGAII-ILS	8	8	12	12	13	14	8	11
OEMOSaDE	9	5	11	11	12	12	15	11
OMOEA-II	15	15	15	8	6	5	12	11

Ordenando essa classificação pela sua média, o Fuzzy-DE ficou em 4º lugar, muito à frente do NSGA-II, que ficou em 12º. É possível perceber ainda que nenhum algoritmo se destacou sobre os demais: o MOEA/D, melhor classificado, ficou em média na 4ª posição, o Fuzzy-DE em 6º e o NSGA-II em 11º.

Embora não tenha sido utilizado na competição do CEC 2009, foi avaliado também o tempo de processamento de cada algoritmo em um computador Intel Core 2 Quad 2.83GHz com 4GB de memória. Os resultados serão apresentados a seguir.

Tabela 4.6 Tempo de processamento, em segundos, para os problemas do CEC 2009

	UF1	UF2	UF3	UF4	UF5	UF6	UF7
MOEA/D	2777,51	2798,46	2792,89	2804,73	2626,07	2659,48	2767,04
Fuzzy-DE	202,98	186,01	160,98	112,93	119,57	144,64	169,43
Fuzzy-DE (%)	7,31%	6,65%	5,76%	4,03%	4,55%	5,44%	6,12%

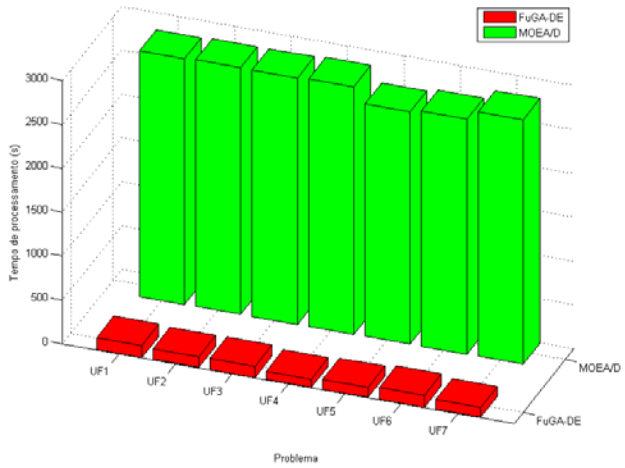

Figura 4-10 Tempo de processamento para os problemas do CEC 2009

Como pode ser observado, o custo computacional do Fuzzy-DE é muito mais baixo que o MOEA/D, sendo em média 17 vezes mais rápido.

1.14.1.2 Métricas de desempenho

Além do IGD, foram utilizadas as métricas citadas anteriormente para avaliação do FuGA, do NSGA-II e do MOEA/D. Todos os resultados foram normalizados com base no MOEA/D, que foi utilizado como referência. Além disso, no CEC 2009 o número de indivíduos para análise foi fixado em 100, o que não ocorre nessa nova bateria de testes.

O critério de parada permaneceu o mesmo, foram realizadas 30 execuções de cada algoritmo e os resultados médios serão apresentados. Para o NSGA-II e o Fuzzy-DE, foram utilizadas populações com 300 indivíduos. Já para o MOEA/D, o valor foi mantido em 600, a fim de evitar alterações em seu desempenho.

Vale ressaltar a complexidade dos problemas e o tamanho do espaço de busca (30 dimensões), o que dificulta a obtenção de fronteiras eficientes mais próximas da fronteira Pareto real.

1.14.1.2.1 Problema UF1

Tabela 4.7 Resultados das métricas para o problema UF1

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	300	5,02%	0,26%	0,00%	101,06%	77,15%	85,04%
NSGA-II	300	37,07%	0,33%	0,00%	99,61%	56,05%	97,81%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

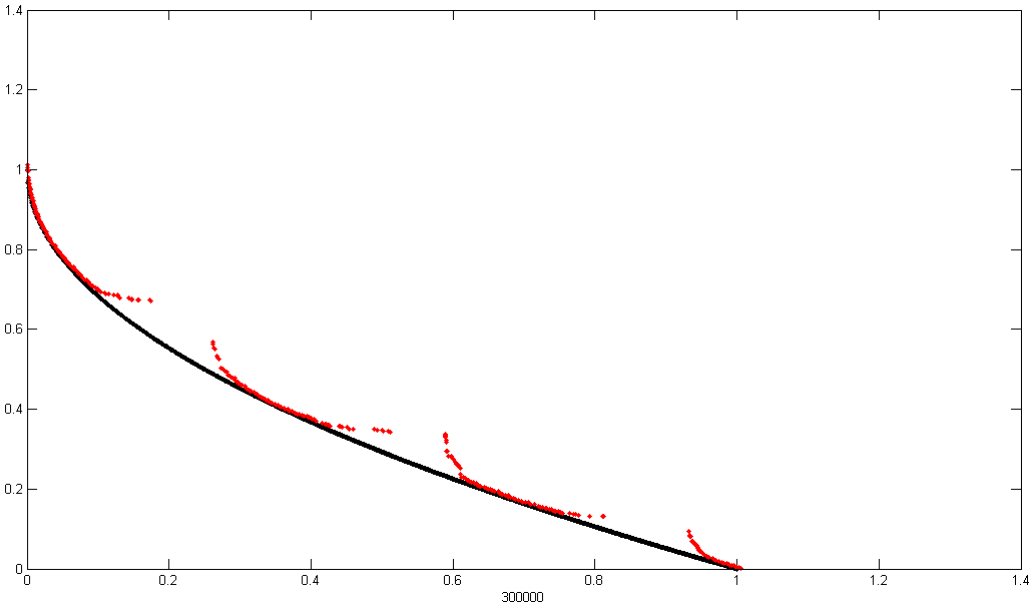

Figura 4-11 Fronteira Pareto obtida para o problema UF1 (30 dimensões)

1.14.1.2.2 Problema UF2

Tabela 4.8 Resultados das métricas para o problema UF2

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	300	4,46%	0,33%	0,00%	100,76%	95,96%	93,51%
NSGA-II	300	31,26%	0,33%	0,00%	101,00%	114,31%	90,34%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

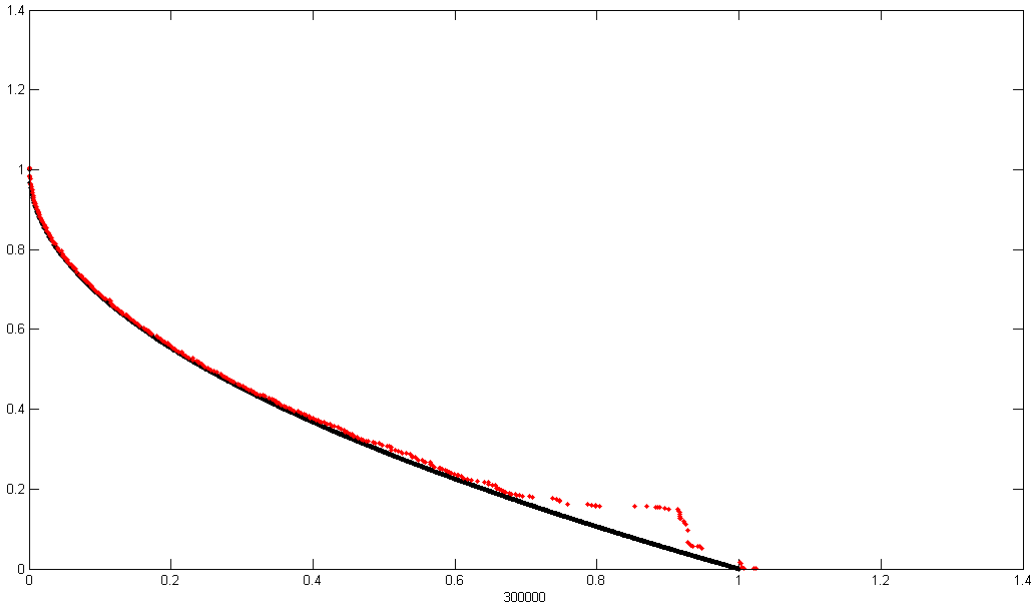

Figura 4-12 Fronteira Pareto obtida para o problema UF2 (30 dimensões)

1.14.1.2.3 Problema UF3

Tabela 4.9 Resultados das métricas para o problema UF3

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	278	5,67%	0,17%	0,00%	100,53%	351,83%	83,39%
NSGA-II	300	35,70%	0,33%	0,00%	88,05%	127,05%	256,61%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

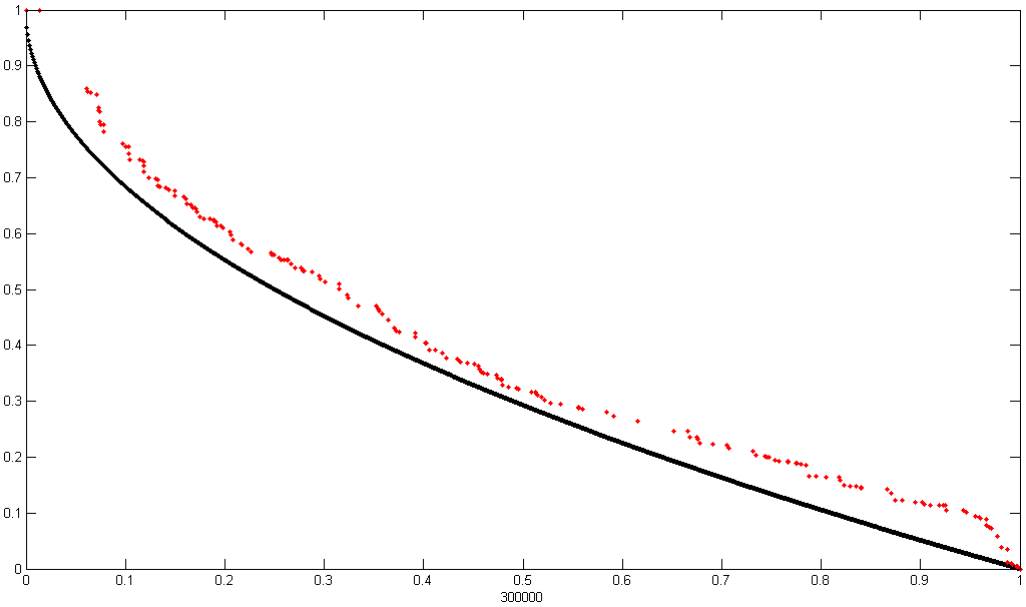

Figura 4-13 Fronteira Pareto obtida para o problema UF3 (30 dimensões)

1.14.1.2.4 Problema UF4

Tabela 4.10 Resultados das métricas para o problema UF4

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	300	3,34%	0,13%	0,20%	93,61%	102,66%	113,36%
NSGA-II	300	30,19%	0,22%	0,11%	93,13%	126,82%	115,07%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

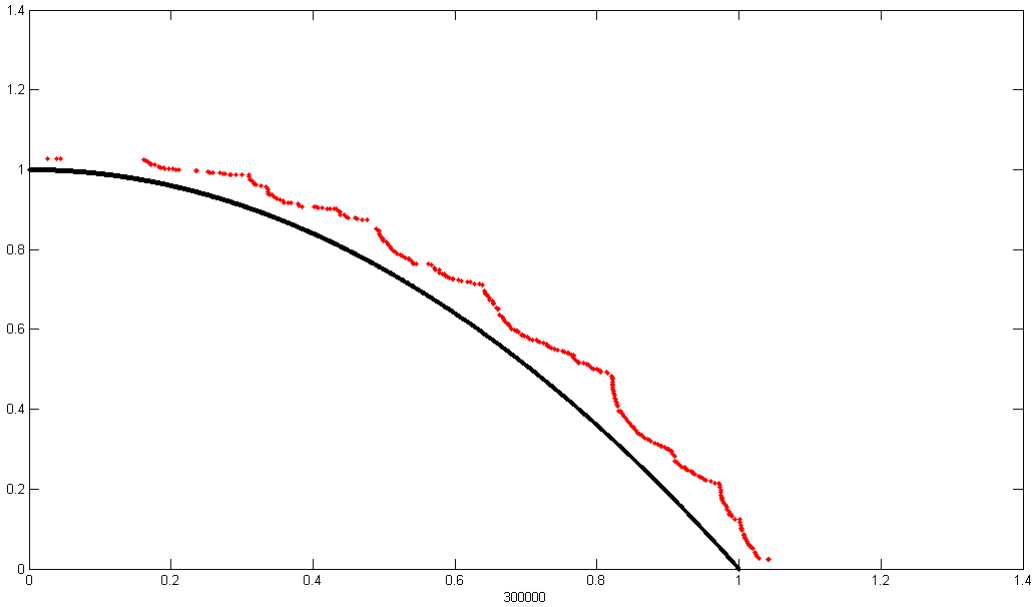

Figura 4-14 Fronteira Pareto obtida para o problema UF4 (30 dimensões)

1.14.1.2.5 Problema UF5

Tabela 4.11 Resultados das métricas para o problema UF5

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	121	4,36%	0,00%	0,84%	149,02%	174,10%	45,48%
NSGA-II	273	45,90%	0,37%	0,00%	396,26%	8,37%	1292,29%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

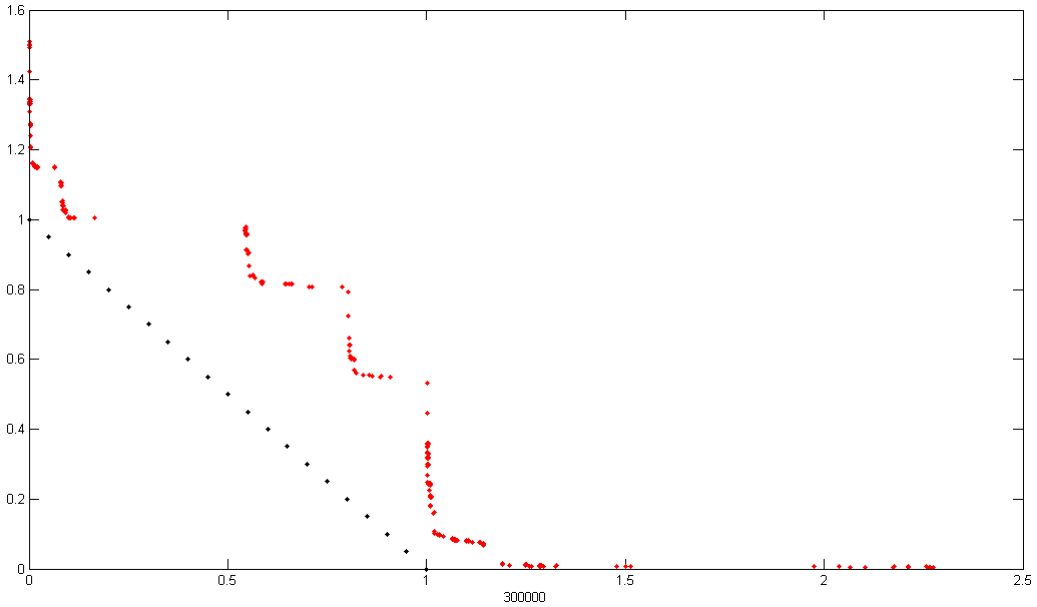

Figura 4-15 Fronteira Pareto obtida para o problema UF5 (30 dimensões)

1.14.1.2.6 Problema UF6

Tabela 4.12 Resultados das métricas para o problema UF6

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	100	5,13%	0,77%	0,00%	117,96%	245,00%	68,41%
NSGA-II	233	48,85%	0,44%	0,00%	101,22%	16,50%	101,47%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

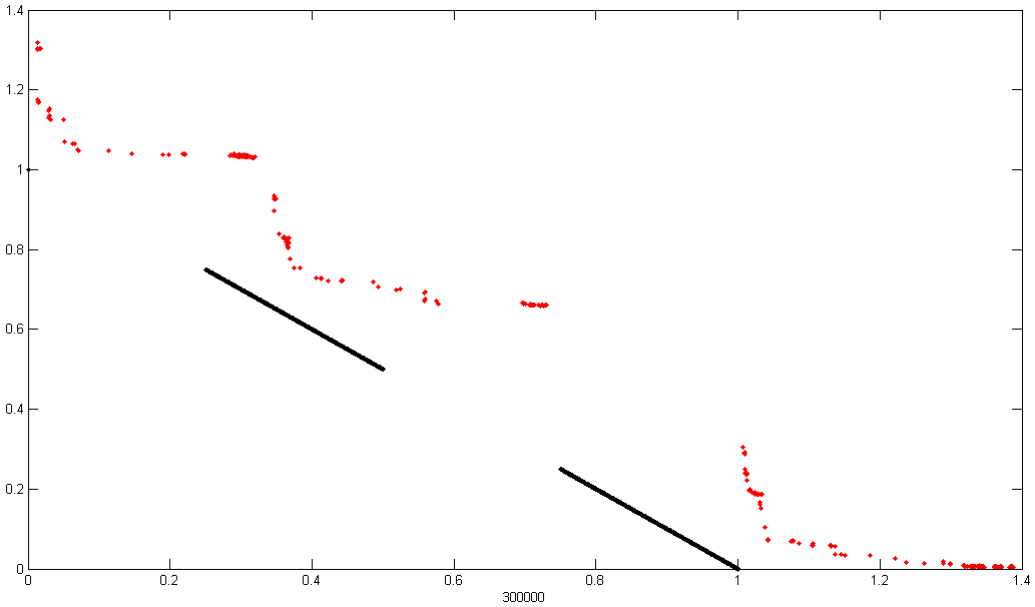

Figura 4-16 Fronteira Pareto obtida para o problema UF6 (30 dimensões)

1.14.1.2.7 Problema UF7

Tabela 4.13 Resultados das métricas para o problema UF7

	Cardinalidade	Tempo	Dominados	Dominantes	Distância	Espalhamento	Volume
Fuzzy-DE	300	3,94%	0,20%	0,00%	98,70%	129,08%	105,83%
NSGA-II	300	36,01%	0,22%	0,00%	98,05%	103,57%	107,75%
MOEA/D	600	100,00%	N/A	N/A	100,00%	100,00%	100,00%

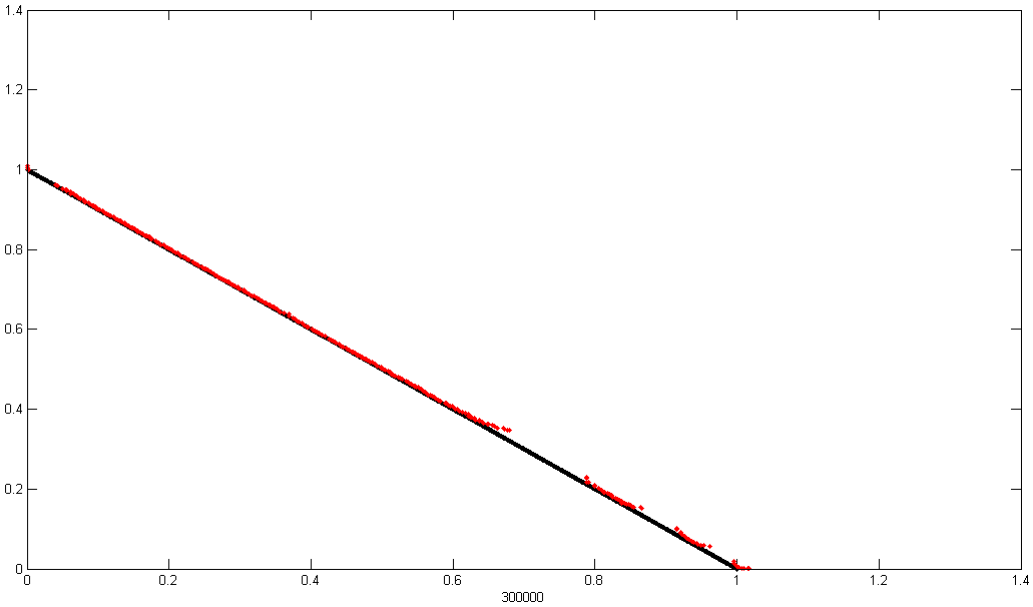

Figura 4-17 Fronteira Pareto obtida para o problema UF7 (30 dimensões)

Devido à diversidade dos problemas, percebe-se que não houve um padrão nos resultados. Entretanto, em todos os casos, o Fuzzy-DE mostrou desempenho computacional superior aos demais, além de ter resultados bastante competitivos, especialmente em relação ao NSGA-II. Além disso, em todos os casos, o Fuzzy-DE foi superior ao MOEA/D em pelo menos uma das métricas, além do tempo.

1.14.2 Problema TEAM22

Para comparar os resultados obtidos com o Fuzzy-DE, foram utilizados trabalhos encontrados na literatura que possuem características semelhantes, como a utilização de algoritmos evolucionários, além dos resultados de referência.

- ✓ (Campelo, et al. 2005) propõe um algoritmo de seleção clonal para tratar problemas de eletromagnetismo [CAMP05].
- ✓ (Soares, et al. 2009) propõe uma metodologia para se trabalhar com incertezas no problema [SOAR09].
- ✓ (Guimaraes, et al. 2006) utiliza uma nova abordagem com três objetivos e utiliza o algoritmo de seleção clonal multiobjetivo [GUIM06].
- ✓ (Takahashi, et al. 2003) utiliza uma nova versão do algoritmo Elipsoidal para o tratamento de restrições de igualdade e o aplica ao TEAM22 [TAKA03].
- ✓ (Alotto, Kuntsevitch, et al. 1996) apresenta o problema e apresenta os resultados obtidos pelo grupo de trabalho [IGTE96].
- ✓ (Alotto 1996) faz um comparativo entre diversos métodos evolucionários em uma abordagem mono-objetivo [ALOT96].

Os resultados foram separados para as versões com três e oito parâmetros.

1.14.2.1 Versão com três parâmetros

Para a versão com três parâmetros, os testes do Fuzzy-DE foram executados 50 vezes e os 10 melhores resultados foram selecionados. A Tabela 4.14 apresenta tais valores, além de um comparativo com outros métodos encontrados na literatura. Já a Figura 4-18 apresenta um diagrama do SMES com para o melhor resultado obtido pelo Fuzzy-DE.

Como pode ser observado, o Fuzzy-DE apresentou bons resultados, mesmo quando comparado a alguns do melhores resultados encontrados na literatura. O melhor resultado obtido ficou a 0,016% (0,0291) do valor de referência, 180MJ.

Tabela 4.14 Resultados obtidos para o TEAM22 com 03 parâmetros

	R_2	$h_2/2$	d_2	f_1	f_2
Fuzzy-DE	3,0586	0,7487	0,1661	180,0291	2,0389E-05
	3,1838	0,3939	0,2252	180,0320	9,7224E-07
	3,1222	0,3030	0,3013	179,9643	8,7194E-07
	3,0759	0,4889	0,2197	179,9632	4,8541E-06
	3,1591	0,3990	0,2305	180,0409	1,1210E-06
	3,1749	0,3738	0,2371	180,0419	9,4144E-07
	3,1663	0,4010	0,2268	179,9549	1,0425E-06
	3,0586	0,3024	0,3301	180,0460	1,2792E-06
	3,1394	0,3842	0,2429	179,9404	1,1066E-06
	3,2118	0,4101	0,2101	179,9329	9,9129E-07
CAMP05	3,1176	0,3009	0,3152	179,9100	8,8900E-04
IGTE96	3,080	0,239	0,394	180,0277	8,8960E-04
TAKA03	3,050	0,246	0,400	175,430	9,6420E-04
GUIM06	3,4000	0,2198	0,2945	184,53	2,56E-03
SOAR09	3,3790	0,2443	0,3168	180,36	1,38E-03

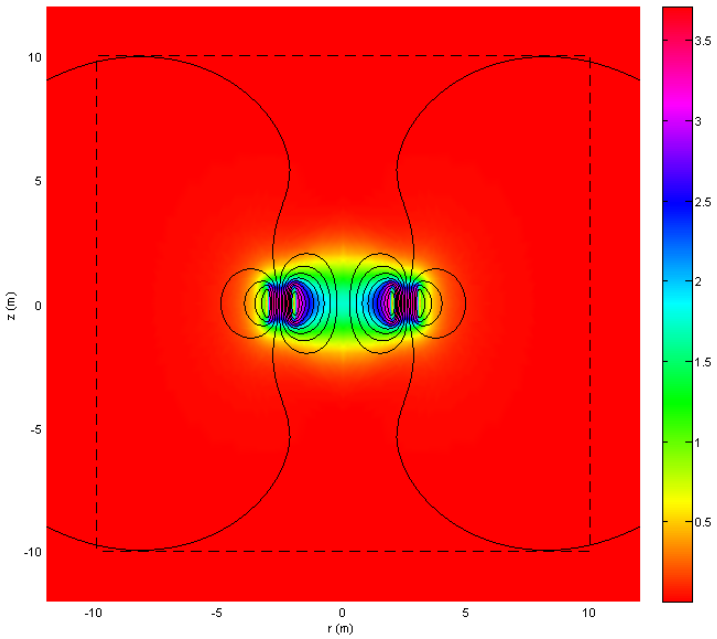

Figura 4-18 Campo de indução magnética do SMES com os resultados obtidos (em Tesla)

1.14.2.2 Versão com oito parâmetros

Para a versão mais complexa do TEAM22, os testes foram realizados em 30 execuções independentes e os melhores resultados estão apresentados na Tabela 4.15. A Figura 4-19 apresenta um diagrama do SMES com para o melhor resultado obtido pelo Fuzzy-DE.

Tabela 4.15 Resultados obtidos para o TEAM22 com 08 parâmetros

	R_1	$h_1/2$	d_1	J_1	R_2	$h_2/2$	d_2	J_2	f_1	f_2
Fuzzy-DE	4,3122	0,1086	0,5969	19,658	2,8128	1,404	0,2012	15,4109	180,0608	8,0602E-05
	2,0713	0,5332	0,5929	12,505	1,3103	1,3958	0,2671	22,9285	180,1603	2,3010E-06
	2,3288	1,419	0,1054	19,188	1,3144	1,4281	0,2995	20,4035	180,3121	1,8177E-07
	2,6512	0,4323	0,4139	11,358	1,3501	0,8315	0,6936	12,5712	180,3494	4,1755E-07
	1,9051	1,0029	0,2722	13,164	1,004	1,563	0,5802	13,4502	180,4383	1,8638E-07
	2,7209	0,7403	0,2953	15,417	1,9912	0,8377	0,7459	10,5043	179,4304	2,0685E-06
	2,543	0,8366	0,4638	10,133	1,3301	0,9362	0,2828	20,4207	181,0704	1,2619E-04
	2,7566	0,7152	0,2283	13,147	1,5383	0,9962	0,5033	13,5551	181,2151	5,3175E-07
	2,0457	0,7183	0,3246	13,953	1,1317	1,3793	0,5079	14,3242	178,5427	2,3443E-07
	3,3284	0,5643	0,175	18,806	1,9354	0,9489	0,2819	19,4271	181,5843	1,4374E-06
ALOT96	1,9390	1,1300	0,3990	22,5000	2,8230	1,1010	0,1950	22,5000	180,0800	1,3750E-05
	1,9900	1,2930	0,2900	26,6000	2,9310	0,9400	0,1880	26,6000	180,1200	1,2990E-05
	1,6940	1,6090	0,3230	20,9000	2,9070	0,8820	0,2070	20,9000	180,4100	1,8640E-05
ALOT	1,1070	0,7840	0,7870	14,2900	2,0230	1,1411	0,1780	-18,0100	179,6900	3,6200E-04
IGTE96	1,5702	0,7846	0,5943	17,3367	2,0999	1,4184	0,2562	-12,5738	179,9924	2,1913E-10

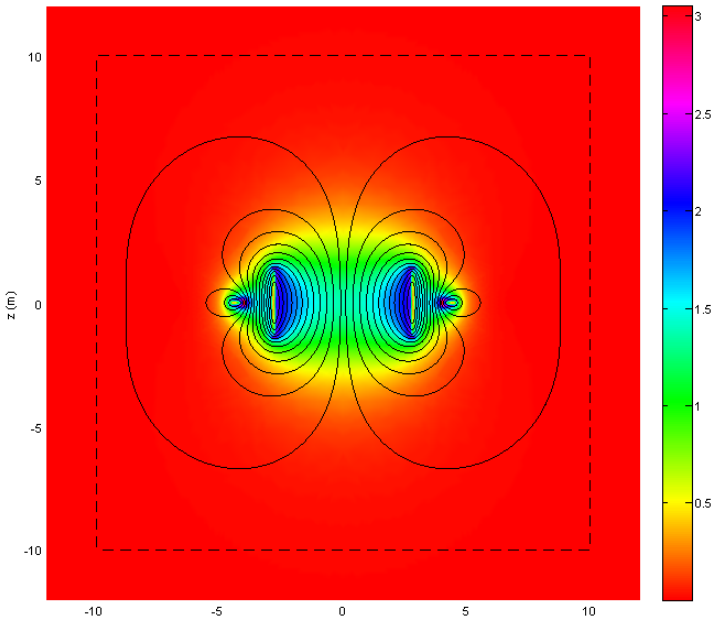

Figura 4-19 Campo de indução magnética do SMES com os resultados obtidos (em Tesla)

Novamente, os resultados do Fuzzy-DE foram bastante competitivos, ficando a 0,034% (0,0608) do valor de referência, mesmo para a versão mais complexa do problema, com oito parâmetros.

5 Conclusões e trabalhos futuros

O presente trabalho propôs o Fuzzy-DE, que possui como principais características:

- ✓ Utilização do método de Evolução Diferencial para a geração de novos indivíduos;
- ✓ Utilização de conceitos de Sistemas Nebulosos, especialmente s-normas, para o cálculo de *fitness*;
- ✓ Baixo custo computacional, quando comparado a métodos como NSGA-II e MOEA/D;
- ✓ Baixo número de parâmetros a serem ajustados;
- ✓ Mecanismo de busca dinâmico, partindo de uma busca global a buscas mais refinadas ao final da execução;
- ✓ Tratamento de problemas restritos e irrestritos sem modificações em sua estrutura.

Apesar de sua simplicidade e facilidade de implementação, o Fuzzy-DE mostrou-se bastante generalista ao apresentar bons resultados para problemas diversos, restritos e irrestritos, sem alterações em sua estrutura. Isso se deve, em parte, ao mecanismo de busca dinâmico obtido com a utilização de pesos dinâmicos ao longo da sua execução.

Além disso, a utilização de uma função de *fitness* simples facilita a sua modificação, de acordo com as necessidades do problema abordado. É possível, por exemplo, dar pesos diferentes para cada função objetivo ou restrição ou utilizar quaisquer outras características do indivíduo para esse cálculo. Para tal, é necessário apenas que tais valores estejam definidos no intervalo $[0,1]$.

Os testes mostraram, por exemplo, que o Fuzzy-DE obteve resultados para o problema TEAM22 que estão a 0,016% e 0,034% do ótimo global estabelecido, para as versões com três e oito parâmetros, respectivamente. Já para os problemas do CEC 2009, o mesmo ficou bem classificado, ocupando o 4º lugar geral, entre os 15 avaliados.

Além disso, a sua diferença para o melhor classificado foi pequena, apesar de ser aproximadamente 20 vezes mais rápido.

Para uma próxima etapa, pretende-se realizar análises de sensibilidade a ajustes em seus parâmetros, a fim de se identificar possíveis melhorias no algoritmo. Pretende-se ainda avaliar a sua eficiência para problemas dinâmicos, já que a diversidade da população pode ser facilmente controlada através dos pesos utilizados para o cálculo do *fitness* de cada indivíduo.

Outra possível abordagem a ser analisada é a utilização do Fuzzy-DE para problemas mono-objetivo, especialmente os restritos e dinâmicos, sem alterações em sua estrutura.

Referências bibliográficas

Alotto, P. "Multiobjective Optimization in Magnetostatics: A Proposal for Benchmark Problems." *IEEE Trans. on Magnetics*, 1996.

Alotto, P., A. V. Kuntsevitch, C. Magele, and G. Molinari. "SMES Optimization Benchmark." *TEAM Workshop Problem 22.* 1996. (accessed 05 09, 2010).

Alotto, P., et al. "SMES Optimization Benchmark Extended: Introducing Pareto Optimal Solutions Into TEAM22." *IEEE Transactions on Magnetics* 44 (june 2008): 1066 -1069.

Campelo, F., F. Guimaraes, H. Igarashi, and J. Ramirez. "A clonal selection algorithm for optimization in electromagnetics." *IEEE Transactions on Magnetics* 41 (may 2005): 1736-1739.

Chakraborty, U. K. *Advances in Differential Evolution.* 2008.

Coello, C. "Evolutionary multi-objective optimization: a historical view of the field." *IEEE Computational Intelligence Magazine* 1 (2006).

Coello, C., D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algortihms for Solving Multi-Objective Problems.* Kluwer Academic, 2002.

Cox, E. "The Fuzzy Systems Handbook." 1994.

de Castro, L. N., and F. Von Zuben. *Recent Developments In Biologically Inspired Computing.* 2004.

De Jong, K. "Evolutionary Computation - a Unified Approach." 2006.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. "A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2000).

Eiben, A., and J. Smith. "Introduction to Evolutionary Computing." 2003.

Fogel, D. B., T. Back, and Z. Michalewics. *Handbook of evolutionary computation.* 1997.

Fogel, L. J., A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* 1966.

Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning.* 1989.

Guimarães, F. G. "Algoritmos de evolução diferencial para otimização e aprendizado de máquina." *IX Congresso Brasileiro de Redes Neurais.* 2009.

Guimaraes, F. G., F. Campelo, R. R. Saldanha, H. Igarashi, R. H. Takahashi, and J. A. Ramirez. "A multiobjective proposal for the TEAM benchmark problem 22." *IEEE Transactions on Magnetics* 42 (2006).

Jang, J. R. *Neuro-Fuzzy and Soft Computing.* 1997.

Lacerda, A. S. M., and W. M. Caminhas. "FuGA - Fuzzy-based Genetic Algorithm." *I Escola Luso-Brasileira de Computação Evolutiva.* 2009.

Miettinen, K. "Nonlinear Multiobjective Optimization." 1999.

Osyczka, A. "Multicriteria optimization for engineering design." 1985.

Pareto, V. "Cours DEconomie Politique, volume I." 1896.

Pedrycz, W., and F. Gomide. *Fuzzy Systems Engineering: Toward Human-Centric Computing.* 2007.

Price, K. "Differential Evolution vs. the Functions of the 2nd ICEO." 1997.

Price, K., R. Storn, and J. Lampinen. *Differential Evolution A Practical Approach to Global Optimization.* 2005.

Ross, T. "Fuzzy Logic with Engineering Applications." 2004.

Sandri, S., and C. Correa. "Lógica Nebulosa." *V Congresso Brasileiro de Redes Neurais.* 2001.

Schwefel, H. "A survey of Evolution Strategies." 1991.

Schwefel, H. "Evolutionsstrategie und numerische Optimierung." 1975.

Soares, G. L., R. Adriano, C. A. Maia, L. Jaulin, and J. A. Vasconcelos. "Robust Multi-Objective TEAM 22 Problem: A Case Study of Uncertainties in Design Optimization." *IEEE Transactions on Magnetics* 45 (march 2009).

Srinivas, N., and K. Deb. "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms." *Evolutionary Computation* 2 (1994).

Storn, R., and K. Price. "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces." 1995.

—. "Minimizing the Real Functions of the ICEC96 Contest by Differential Evolution." 1996.

Takahashi, R. H., R. R. Saldanha, W. Dias-Filho, and J. A. Ramirez. "A new constrained ellipsoidal algorithm for nonlinear optimization with equality constraints." *IEEE Transactions on Magnetics* 39 (may 2003).

Tsoukalas, L., and R. Uhrig. *Fuzzy and Neural Approaches in Engineering.* 1996.

Van Veldhuizen, D. "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations." 1999.

Zadeh, L. A. "Fuzzy Sets." *Information and Control*, 1965.

Zhang, Q., A. Zhou, S. Zhao, P. Suganthan, W. Liu, and S. Tiwari. "Multiobjective optimization test instances for the CEC 2009 special session and competition." 2009.

Zhang, Q., and H. Li. "A multi-objective evolutionary algorithm based on decomposition." *IEEE Transactions on Evolutionary Computation* 2007 (2007).

Zitzler, E., and L. Thiele. "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach." 1999.

Zitzler, E., K. Deb, and L. Thiele. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results." *Evolutionary Computation* 8 (2000).

Zitzler, E., M. Laumanns, and L. Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm." techreport, 2001.