ESCOLA POLITÉCNICA DE
PERNAMBUCO

Trabalho de Conclusão de Curso

Engenharia de Computação

MOPSO-CDR com Especiação (MOPSO-CDRS)

Autor: Péricles Barbosa Cunha de Miranda

Orientador: Carmelo José Albanez Bastos Filho

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA DE PERNAMBUCO

PÉRICLES BARBOSA CUNHA DE MIRANDA

MOPSO-CDR COM ESPECIAÇÃO (MOPSO-CDRS)

Recife, Dezembro de 2010.

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco

Dedico este trabalho à Família, amigos e amor.

Agradecimentos

Agradeço a Deus, por me dar forças para a realização deste trabalho e para a conclusão do curso. Agradeço a meus pais, Antônio Pádua e Mércia, que sempre me dão força e apoio em minhas decisões, e ao meu irmão Filipe pelo seu companheirismo. Agradeço a toda a minha família sempre acolhedora e amável. Agradeço pelo amor, compreensão, companheirismo e ajudinhas no *Word* do meu amor, Mouriely Rodrigues, que me atura a 4 anos. Agradeço a todos os amigos que conquistei em minha vida. Agradeço ao Professor e Orientador Carmelo Bastos que me acolheu como seu aluno de Iniciação Científica e me orientou neste trabalho.

Resumo

O sucesso da heurística de otimização por enxame de partículas (PSO) como otimizador de único objetivo tem motivado a sua aplicação em outras áreas. Uma dessas é a de Problemas com Múltiplos Objetivos (MOP). Problemas de otimização com múltiplos objetivos são bastante comuns em diversas áreas do conhecimento. Devido a isso, muitas propostas para melhorar a convergência das soluções, têm sido propostas. Muitas variações no algoritmo MOPSO padrão foram propostas, visando alterações na equação de atualização de velocidade do PSO e inserção de fatores de mutação. Outras propostas visaram modificações na estrutura do Arquivo Externo e no processo de seleção de líderes cognitivos e sociais. Neste trabalho, uma nova técnica para o MOPSO é proposta, com o propósito de melhorar a convergência das soluções, através da criação de um *Gerenciador de Arquivo Externo e Executor de Decisões*. A busca por melhores soluções sem que o enxame fique estagnado em regiões onde se encontram soluções sub-ótimas é também um dos principais propósitos desta técnica, visando melhorar a capacidade de convergência de exploração do enxame e fornecer maior qualidade à solução obtida. Comparações com as técnicas já existentes também são realizadas, cujos resultados das simulações demonstram que a técnica proposta apresenta, em diversos casos, melhores resultados que as técnicas encontradas na literatura como MOPSO-CDR e MOPSO. Por fim, estudos foram realizados, analisando o comportamento da técnica no início das iterações, percebendo-se a sua alta capacidade de convergência.

Abstract

The success of Particle Swarm Optimization (PSO) heuristic as single-objective optimizer has motivated its application in other areas. One of them is Multi-Objective Optimization Problems (MOP). Optimization problems are very common in many areas of knowledge, thus many proposals to improve the convergence of External Archive' solutions have been proposed. Many variations on MOPSO standard algorithm were proposed focusing in changes on PSO velocity update equation and the insertion of mutation factors. Other proposals focused in modifications on External Archive structure and in the leaders selection process. In this work, a new MOPSO technique is proposed to improve the solutions' convergence, through the creation of an External Archive Manager and a Decision Executer. The search for best solutions avoiding stagnation is one of the main purposes of this technique, aimed to improve the exploration convergence capability of the swarm and provide high quality solutions. Comparisons with well known techniques were performed, where the simulations' results have shown that our proposal out performed in many cases well known techniques such as MOPSO-CDR and MOPSO. At last, we performed, an analysis in the beginning of the search process and we checked a superior convergence capacity of our proposal.

Sumário

CAPÍTULO 1 INTRODUÇÃO 1

1.1. Objetivos 2

1.2. Estrutura do Documento 3

CAPÍTULO 2 INTELIGÊNCIA DE ENXAMES E PSO 4

2.1. Otimização por Enxame de Partículas 4

2.2. Peso de Inércia e Fator de Constrição 7

2.3. Topologias Básicas 9

CAPÍTULO 3 OTIMIZAÇÃO MULTI-OBJETIVO 11

3.1. Conceitos Básicos 11

3.2. Métricas para Cálculo de Desempenho 14

CAPÍTULO 4 ABORDAGENS DE PSO PARA MOP 17

4.1. PSO aplicado a Problemas Multi-Objetivo 17

4.2. Trabalhos Relacionados 19

4.3. Detalhamento do MOPSO-CDR 23

CAPÍTULO 5 A NOVA ABORDAGEM, MOPSO-CDR COM ESPECIAÇÃO .. 27

5.1. Alterações Propostas 27

5.2. Seleção do Líder Social 29

5.3. Seleção do Líder Cognitivo 30

5.4. Analisando o Arquivo Externo e Tomando Decisões 30

5.5. Algoritmo 34

CAPÍTULO 6 EXPERIMENTOS 36

6.1. Arranjo Experimental 36

6.2. Resultados 42

CAPÍTULO 7 CONCLUSÕES E TRABALHOS FUTUROS 49

7.1. Contribuições 49

7.2. Conclusão 50

7.3. Trabalhos Futuros 51

BIBLIOGRAFIA 52

Índice de Figuras

Figura 1. Vetores que influenciam o movimento das partículas. 5

Figura 2. Fluxograma do algoritmo Clássico do PSO. 7

Figura 3. Topologias (a) Estrela e (b) Anel. 10

Figura 4. Relação de dominância em um espaço bi-objetivo. 12

Figura 5. Conjunto de soluções do *Pareto Front.* 13

Figura 6. Retângulos necessários para cálculo do Hypervolume. 14

Figura 7. Pseudo-código do algoritmo genérico do PSO em MOP. 19

Figura 8. Estratégia de escolha do (a) $gbest1$ e (b) $gbest2$. Figura adaptada de Chiu[14]. 20

Figura 9. Distribuição das soluções no Arquivo Externo. 22

Figura 10. Pseudo-código do algoritmo do MOPSO-CDR. 23

Figura 11. Casos possíveis para o AE. 25

Figura 12. Ilustração do processo de Especiação do enxame. 28

Figura 13. Seleção do líder social com *Especiação.* 29

Figura 14. *Fitness* do $pbest(t)$ e *fitness* da posição atual da $partículai$. 30

Figura 15. O *Executor* decide entre os estados: (1) MOPSO-CDR ou *Básico*, e (2) *Especiação* para fazer a seleção dos líderes. 31

Figura 16. No estado *Básico*, o *Gerenciador* tem o objetivo de analisar o *pareto* quanto ao *spreading* de modo a auxiliar na sua convergência. (a) Representa o *Pareto* com pouco espalhamento, já (b) mostra o *Pareto* com melhor grau de convergência em relação ao *spreading.* 32

Figura 17. No estado *Especiação*, o *Gerenciador* tem o objetivo de analisar o *pareto* quanto ao *spacing* de modo a auxiliar na sua convergência. (a) Representa o *Pareto* com espaçamento irregular entre as soluções, já (b) mostra a regularidade de espaçamento entre as soluções do *Pareto,* ou seja, melhor taxa de *spacing.* . 33

Figura 18. Pseudo-código do algoritmo do MOPSO-CDRS. 34

Figura 19. Representação gráfica da função ZDT1. 37

Figura 20. Representação gráfica da função ZDT2. 38

Figura 21. Representação gráfica da função ZDT3. 39

Figura 22. Representação gráfica da função ZDT4. 40

Figura 23. Representação gráfica da função ZDT6. 40

Índice de Tabelas

Tabela 1. Resultado da simulação para a função ZDT1 com 200.000 chamadas..... 42

Tabela 2. Resultado da simulação para a função ZDT2 com 200.000 chamadas..... 43

Tabela 3. Resultado da simulação para a função ZDT3 com 200.000 chamadas..... 43

Tabela 4. Resultado da simulação para a função ZDT4 com 200.000 chamadas..... 44

Tabela 5. Resultado da simulação para a função ZDT6 com 200.000 chamadas..... 44

Tabela 6. Resultado da simulação para a função ZDT1 com 100.000 chamadas..... 45

Tabela 7. Resultado da simulação para a função ZDT2 com 100.000 chamadas..... 45

Tabela 8. Resultado da simulação para a função ZDT3 com 100.000 chamadas..... 46

Tabela 9. Resultado da simulação para a função ZDT4 com 100.000 chamadas..... 46

Tabela 10. Resultado da simulação para a função ZDT6 com 100.000 chamadas..... 46

Tabela 11. Tempo de execução, em segundos, dos algoritmos MOPSO-CDRS e MOPSO-CDR para cada função objetivo usando-se 100.000 chamadas. 47

Tabela 12. Tempo de execução, em segundos, dos algoritmos MOPSO-CDRS usando 100.000 chamadas e MOPSO-CDR usando 200.000 chamadas para cada função objetivo..... 48

Tabela de Símbolos e Siglas

CD – *Crowding Distance*

CDR – *Crowding Distance and Roulette Wheel*

CSS-MOPSO – *Cross-Searching Strategy for Multi-Objetive* PSO

MOO – *Multiple Objective Optimization*

MOP – *Multi-Objective Problem*

MOPSO – *Multi-Objective Particle Swarm Optimization*

MOPSO-CDLS – *Multi-Objective Particle Swarm Optimization with Crowding Distance and Local Search*

MOPSO-CDR - *Multi-Objective Particle Swarm Optimization with Crowding Distance and Roulette Wheel*

MOPSO-CDRS - *Multi-Objective Particle Swarm Optimization with Crowding Distance and Roulette Wheel with Speciation*

PSO – *Particle Swarm Optimization*

RW – *Roulette Wheel*

Capítulo 1

Introdução

Encontrar soluções para problemas com múltiplas variáveis sempre foi uma tarefa árdua. Por esse mesmo motivo, existe uma atenção relevante da comunidade científica para desenvolver e aperfeiçoar algoritmos matemáticos complexos com o intuito de tentar resolver problemas de maneira rápida e precisa.

Uma das técnicas mais conhecidas e usadas nos campos de otimização e busca é o *Particle Swarm Optimization* (PSO) [1]. Esta técnica foi desenvolvida por James Kennedy e Russel Eberhart [2] em 1995 e foi inspirada no comportamento social de bandos de pássaros em busca de alimento. Basicamente, o algoritmo resume-se à modelagem da atualização das posições e velocidade de cada indivíduo do bando em modelos pré-definidos de comunicação.

A popularidade do PSO foi alcançada devido a sua simplicidade e a sua capacidade de gerar soluções precisas de forma rápida. O seu sucesso como otimizador de funções de único objetivo tem motivado a extensão e aplicação desta técnica bio-inspirada em outros tipos de problemas [3]. Devido à sua rápida convergência, o PSO tem sido bastante aplicado no contexto de problemas multi-objetivos.

Em muitos casos, o processo de otimização apresenta mais que um objetivo e esses objetivos podem ser conflitantes [1]. No caso de Problemas Multi-Objetivos (MOP, do inglês *Multi-Objective Problems*), a solução considerada ótima ou adequada é um vetor de soluções onde cada solução responde a um objetivo.

Um dos grandes desafios é definir o conjunto de soluções que são consideradas adequadas para os problemas em questão. O processo que determina a melhor condição de uma solução em relação à outra é fundamentado no conceito matemático de *dominância*. Uma solução *x* é considerada melhor que

y se *x* domina *y*, ou seja, para cada objetivo a ser atendido, *x* apresenta as melhores soluções para cada um deles.

Diante dos desafios que o campo de MOP proporciona, vários trabalhos vêm sendo desenvolvidos propondo estratégias com o objetivo de tornar o comportamento das soluções mais distribuído e contínuo.

1.1. Objetivos

Este trabalho de conclusão de curso tem como objetivo o desenvolvimento de uma nova técnica inspirada no algoritmo MOPSO-CDR, desenvolvido por Santana, Pontes e Bastos-Filho [4].

O MOPSO-CDR com Especiação (MOPSO-CDRS) utiliza uma abordagem alternativa ao CDR no processo de escolha do líder social e cognitivo.

A técnica desenvolvida apresenta um processo de análise das soluções não dominadas no Arquivo Externo (AE). Este processo estuda a convergência das soluções do AE, de modo a tentar identificar anomalias e tomar as decisões necessárias para saná-las.

A escolha dos líderes depende da análise feita no AE. Dependendo do sintoma avaliado após a aplicação das métricas no arquivo externo, pode-se tomar a seguintes decisões:

1. Manter a escolha dos líderes proposto pelo MOPSO-CDR;
2. Dividir o enxame em grupos ou sub-enxames e atribuir a cada sub-enxame tarefas específicas.

Espera-se que com essa melhoria possam-se obter melhores resultados quanto ao espalhamento e espaçamento das soluções no arquivo externo.

1.2. Estrutura do Documento

Este trabalho esta organizado em capítulos, detalhados a seguir:

O Capítulo 2 introduz o conceito da técnica de otimização por enxame de partículas. Será discutida a teoria fundamental da proposta, sua estrutura clássica, e também as principais variações na equação de atualização da velocidade.

O Capítulo 3 apresenta os conceitos básicos da área de otimização multi-objetivo, mostra as funções de teste e as métricas de desempenho que foram implementadas neste trabalho.

O Capítulo 4 aborda a aplicação da técnica PSO em problemas com múltiplos objetivos. Além disso, serão apresentadas algumas técnicas, encontradas na literatura, que implementam esta combinação, uma delas sendo o MOPSO-CDR, base deste trabalho, que será vista com mais detalhes.

O Capítulo 5 detalha sobre a proposta do trabalho, o MOPSO-CDR com Especiação, enfatizando as mudanças realizadas e o algoritmo desenvolvido, incluindo a nova abordagem de seleção dos líderes cognitivo e social.

O Capítulo 6 mostra a configuração, conjunto de funções de teste e métricas, necessários para a realização das simulações; os resultados obtidos, suas análises e comparação com outras técnicas.

No Capítulo 7 são apresentadas as conclusões, contribuições e listados alguns possíveis trabalhos futuros.

Capítulo 2

Inteligência de Enxames e PSO

Neste capítulo serão abordados os conceitos básicos da técnica PSO e algumas das principais equações de atualização de velocidade.

2.1. Otimização por Enxame de Partículas

A idéia dos algoritmos de otimização baseados em enxames de partículas iniciou-se com estudos a respeito do comportamento de bandos de pássaros, quais regras os regiam, e como poderiam mudar o curso de forma tão repentina e sincronizada [5]. O PSO foi proposto após a simulação de modelos sociais simplificados baseados em observações feitas nos bandos de aves à procura de alimento.

Um enxame pode ser definido com um conjunto de indivíduos que interagem localmente entre si, regidos por um comportamento global, buscando a solução para problemas de forma distribuída [6].

Atualmente, na maioria das implementações do PSO, as partículas movem-se no espaço de busca sendo guiadas por uma combinação entre a melhor posição encontrada pela partícula e a melhor posição encontrada pela vizinhança em que ela está inserida. Esta vizinhança é definida como um conjunto de partículas com as quais a partícula em análise pode se comunicar, podendo este conjunto se estender para todo o enxame ou não. O que define esta comunicação entre as partículas é a topologia de comunicação entre elas; este capítulo mostrará, de forma breve, duas topologias consideradas base de todas as outras: Anel e Estrela.

Cada partícula está sob influência de três forças que podem ser representadas matematicamente como vetores, a saber:

a) Vetor inércia: representa o movimento atual da partícula, ou seja, a velocidade atual. Na equação (1) está representado por $\overrightarrow{v_{idt}}$;

b) Vetor cognitivo: representa a componente cognitiva da partícula, uma relação entre a posição atual e a melhor posição encontrada por aquela partícula. Na equação (1) é representado pelo termo $c_1\epsilon_1(\vec{p}_i - \overrightarrow{x_i})$;

c) Vetor social: representa a influência do enxame em uma determinada partícula. É uma relação entre a melhor posição encontrada pelo enxame e a posição atual da partícula. Na equação (1) é representado pelo termo $c_2\epsilon_2(\vec{p}_g - \overrightarrow{x_i})$.

Um esquema gráfico pode ser observado na Figura 1, onde $\overrightarrow{x_{i\,(t)}}$ representa a posição atual da partícula e $\overrightarrow{x_{i\,(t+1)}}$ representa a posição da partícula após o processo de atualização da velocidade, que é determinante na atualização da posição.

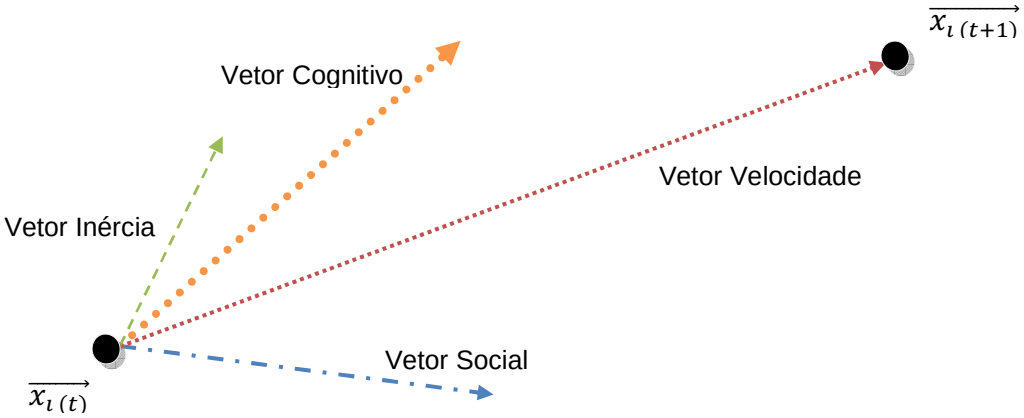

Figura 1. Vetores que influenciam o movimento das partículas.

O algoritmo do PSO apresenta como passo inicial a inicialização aleatória das posições e velocidades das partículas. Cada partícula *i* é representada por três vetores:

a. Sua posição em um espaço de busca *D*-dimensional: $\vec{x} = (x_{i1}, x_{i2}, \dots, x_{iD})$;
b. A melhor posição que a partícula i encontrou durante o processo de busca: $\overrightarrow{p_i} = (p_{i1}, p_{i2}, \dots, p_{iD})$;

c. A sua velocidade atual: $\overrightarrow{v_i} = (v_{i1}, v_{i2}, \dots, v_{iD})$.

As partículas então se movem no espaço de busca à procura da melhor solução possível. A cada iteração o algoritmo atualiza as posições e velocidades das partículas usando as equações (1) e (2).

$$\overrightarrow{v_{id(t+1)}} = \overrightarrow{v_{idt}} + c_1\epsilon_1(\overrightarrow{p_{idt}} - \overrightarrow{x_{idt}}) + c_2\epsilon_2(\overrightarrow{p_{gdt}} - \overrightarrow{x_{idt}}), \quad (1)$$

$$\overrightarrow{x_{id(t+1)}} = \overrightarrow{x_{idt}} + \overrightarrow{v_{idt}}. \quad (2)$$

No algoritmo original, c_1 e c_2 são constantes com valor igual a 2,0, ϵ_1 e ϵ_2 são números aleatórios entre 0 e 1, a cada iteração para cada partícula e cada dimensão. O vetor $\vec{p}_g$ é a melhor posição encontrada pela melhor partícula no enxame.

O algoritmo PSO define a existência de uma condição de parada para a execução do processo, sendo esta determinada pela quantidade de iterações que o algoritmo deve executar ou através de um limiar de aceitação, ou seja, se o enxame chegou até um ponto cujo desempenho não melhore significativamente. Além da definição da condição de parada, deve-se pré-definir o número de partículas do enxame, a literatura costuma utilizar de 20 a 50 partículas. Para cada intervalo *t* e para cada partícula *i* do enxame, o algoritmo deve avaliar a velocidade e posição usando as equações (1) e (2); atualizar o vetor de posição $\vec{x}_i$ da partícula *i* com os valores encontrados; calcular o *fitness* (desempenho) da partícula *i* e atualizar os valores de $\vec{p}_g$ e $\vec{p}_i$.

Este algoritmo pode ser facilmente entendido observando-se a Figura 2. No início do algoritmo, o enxame é inicializado, e com o passar das iterações, a posição, velocidade, *pbest* e *gbest* das partículas são atualizados, de modo que no fim do algoritmo cada partícula apresenta uma solução para o problema proposto.

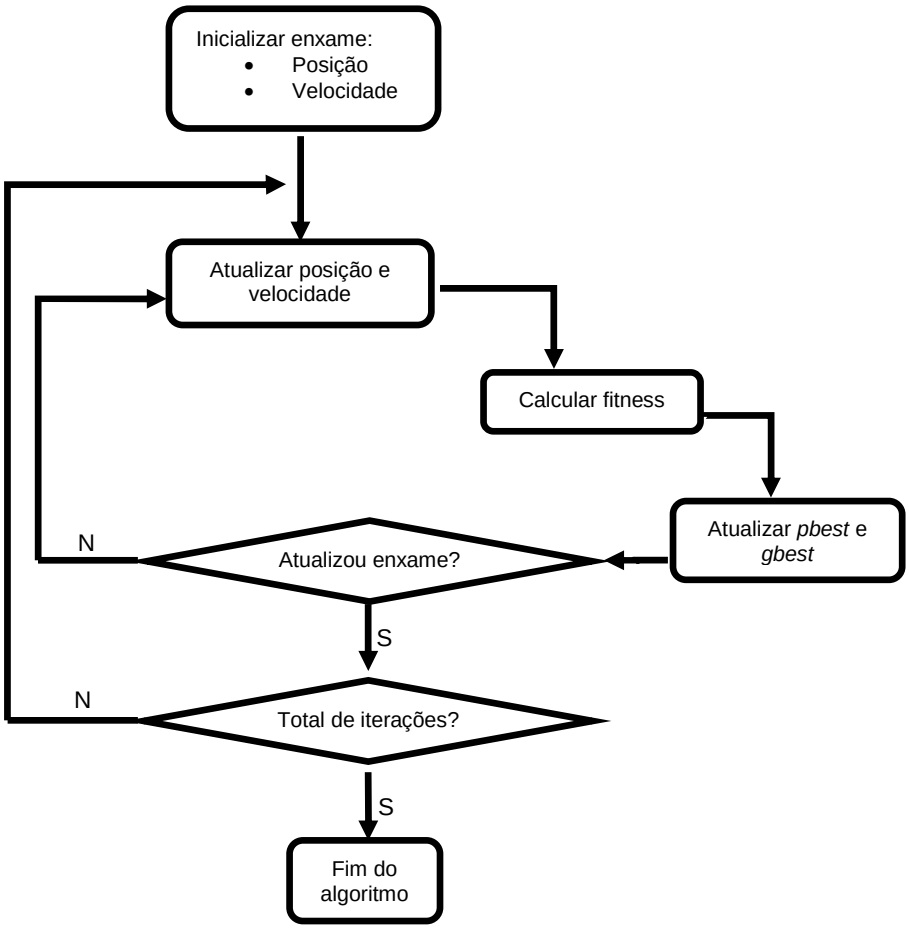

Figura 2. Fluxograma do algoritmo Clássico do PSO.

2.2. Peso de Inércia e Fator de Constrição

Um fenômeno freqüentemente observado nos enxames que utilizavam o algoritmo clássico é a “explosão” de velocidades. Facilmente uma partícula adquiria uma velocidade muito alta muito rapidamente, o que a levava a oscilar entre os extremos do espaço de busca.

Foi proposto, por Eberhart e Kennedy [7], um mecanismo que estabelece um limite para a velocidade das partículas $\overrightarrow{vmax}$. Foi observado, no entanto, que a determinação do valor $vmax$ não era nada trivial, e a escolha errada para este valor poderia implicar em uma diminuição do desempenho. Por exemplo, espaços de busca maiores demandavam valores maiores que $\overrightarrow{vmax}$ para garantir a

exploração e espaços de busca menores exigiam menores valores para evitar o problema da “explosão” de velocidades.

O peso de inércia e fator de constrição [8] foram introduzidos com a intenção de se remover completamente o conceito de velocidade limite, tentando-se evitar a “explosão” da velocidade das partículas. A idéia foi de ponderar a velocidade anterior da partícula no processo de atualização de sua velocidade. A equação de velocidade pode ser reescrita na forma da equação (3) para comportar o conceito de peso de inércia (representado por ω):

$$\overrightarrow{v_{id(t+1)}} = \omega\overrightarrow{v_{idt}} + c_1\epsilon_1(\overrightarrow{p_{idt}} - \overrightarrow{x_{idt}}) + c_2\epsilon_2(\overrightarrow{p_{gdt}} - \overrightarrow{x_{idt}}). \quad (3)$$

Diversos estudos foram realizados sobre os efeitos dos valores de ω sobre a atualização de velocidade. Os resultados obtidos com a variação da inércia com o passar do tempo demonstram uma convergência mais rápida do que os mesmos obtidos com valores de inércia estáticos.

Sugere-se que durante o processo de otimização, ω iniciando com valores altos próximos a 1,0 encoraja as partículas a uma exploração maior do espaço de busca (exploração em amplitude) e eventualmente esses valores vão decrescendo (abaixo de 1,0), focando assim os esforços do enxame na melhor área encontrada durante a exploração (exploração em profundidade) [9].

O fator de constrição, conceito similar ao peso de inércia, consiste na introdução de um novo parâmetro χ, derivado das constantes existentes na equação de velocidade. A escolha destas constantes influencia na velocidade de convergência e na capacidade do algoritmo de encontrar a solução ótima. Pelo fato de cada problema a ser solucionado necessitar de constantes específicas, a idéia do fator de constrição é balancear a influência das constantes, independente do problema. O parâmetro χ é calculado de acordo com a equação (4),

$$\chi = \frac{2}{\left\|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right\|}, \quad (4)$$

$$\varphi = c_1 + c_2. \quad (5)$$

Foi demonstrado por Clerc e Kennedy [8] que para valores de c_1e c_2 tal que $\varphi < 4$, o enxame converge lenta e espiralmente para a solução. Ao passo que

quando $\varphi > 4$ a convergência é rápida e garantida para um ótimo local. Por simplicidade, assume-se o valor de $\varphi = 4{,}1$ e conseqüentemente c_1 e c_2 iguais a 2,05 para assegurar convergência. Para aplicar o fator de constrição à equação de atualização de velocidade, esta precisa ser reescrita na forma da equação (6).

$$\overrightarrow{v_{\imath d(t+1)}} = \chi\left[\overrightarrow{v_{\imath dt}} + c_1\epsilon_1\left(\overrightarrow{p_{\imath dt}} - \overrightarrow{x_{\imath dt}}\right) + c_2\epsilon_2\left(\overrightarrow{p_{gdt}} - \overrightarrow{x_{\imath dt}}\right)\right]. \quad (6)$$

Outro fator muito importante e responsável por grande parte do sucesso ou fracasso no processo de otimização é a forma como as partículas se comunicam. A seção 2.3 trata do conceito de estruturas de comunicação entre partículas ou topologias.

2.3. Topologias Básicas

Algoritmos inteligentes baseados em enxames, e conseqüentemente semelhantes ao PSO, são algoritmos que tentam mapear comportamentos sociais em um ambiente computacional controlado. As partículas que compõem o enxame precisam de alguma forma, propagar as informações que conseguem coletar, caso contrário, os referidos algoritmos não poderiam se apoiar no conceito de sociedade, pois as partículas estariam “voando” pelo espaço de busca à procura de soluções sendo influenciadas apenas por sua própria experiência.

Neste cenário, as topologias, que definem as regras de como as partículas devem se comunicar, desempenham um papel importantíssimo, influenciando completamente o comportamento do enxame.

As topologias mais conhecidas e utilizadas são as topologias estrela e anel [2] apresentadas na Figura 3.

A topologia estrela, Figura 3 (a), foi a primeira a ser proposta. Nesta topologia cada partícula pode se comunicar com qualquer outra partícula do enxame. Conseqüentemente, uma partícula é influenciada por todas as outras partículas, pois estaria recebendo informações de todo o enxame, este modelo também é conhecido $gbest$.

Na topologia anel, Figura 3 (b), também conhecida como $lbest$, cada partícula se comunica apenas com seus vizinhos diretos. Uma decorrência direta desta diferença entre os modelos de comunicação é que a topologia global apresenta um desempenho superior à topologia em anel em problemas uni-modais (problemas com apenas uma solução ótima), enquanto que problemas multimodais (problemas com várias soluções ótimas ou sub-ótimas) são melhores tratados com a topologia em anel. Isso se deve ao fato de enxames com topologia em anel explorarem melhor o ambiente, não atraindo todas as partículas para uma solução sub-ótima rapidamente, o que é bastante interessante para problemas multimodais.

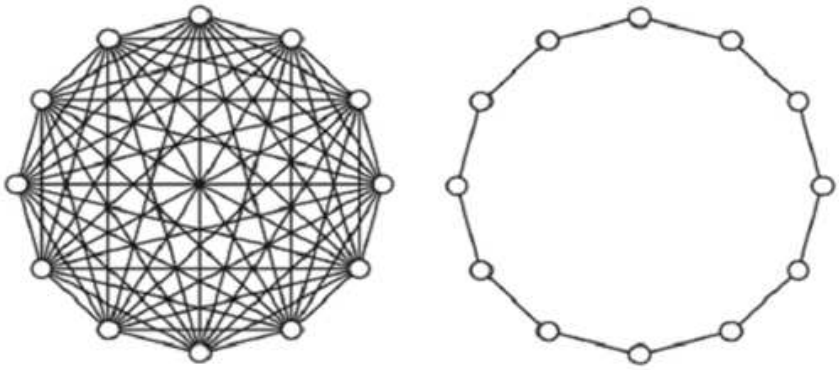

Figura 3. Topologias (a) Estrela e (b) Anel.

Vale salientar que na maioria das aplicações reais do PSO não se conhece bem o problema a ser resolvido, muito menos se sabe quantas soluções satisfatórias ele pode ter. Conseqüentemente, utilizar a topologia global pode ser muito arriscado se a solução ótima for necessária. No entanto, aplicar a topologia em anel geralmente leva a uma convergência mais lenta, pois num enxame de *N* partículas, uma partícula pode ter de esperar de uma a *N/2* iterações para indiretamente receber informações da melhor partícula do enxame, enquanto que na topologia global, após a primeira iteração todas as partículas já têm conhecimento da melhor solução obtida pelo enxame.

Capítulo 3

Otimização Multi-Objetivo

Este capítulo introduz alguns conceitos básicos, da área de otimização com múltiplos objetivos, considerados fundamentais para o melhor entendimento deste trabalho. Além disso, serão apresentadas algumas funções de teste (problemas a serem solucionados) e métricas de desempenho, que auxiliam na análise do retorno gerado pelo algoritmo.

3.1. Conceitos Básicos

Em contraste com a otimização de um único objetivo, a Otimização Multi-Objetivos (MOO, do inglês *Multi-Objetive Optimization*), diferentemente do PSO, possui vários objetivos a serem solucionados e sua principal meta é obter um conjunto de soluções bem distribuído e diverso. MOO pode ser definido como o problema de encontrar um vetor de variáveis de decisão que satisfazem restrições e otimizam um vetor de funções cujos elementos representam as funções objetivo. Um problema genérico na otimização multi-objetivo contendo um número de objetivos a serem minimizados como mostrados em (6), por exemplo, e restrições a serem satisfeitas como apresentados em (7)(8) pode ser escrito da seguinte forma:

$$\text{Minimizar } f_{(\vec{x})} \coloneqq [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})], \tag{7}$$

Sujeito a:

$$g_{i_{(\vec{x})}} \leq 0 \quad i = 1,2,\dots,m, \tag{8}$$

$$h_{j_{(\vec{x})}} = 0 \quad j = 1,2,\dots,p. \tag{9}$$

onde $\vec{x} = [x_1, x_2, \dots, x_n]^T$ é o vetor de variáveis de decisão, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \dots, k$ são as funções objetivo e $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \dots, m$, $j = 1, \dots, p$ são as funções de restrição do problema.

Em problemas de otimização multi-objetivo existe um conjunto de soluções equivalentes consideradas superiores ao restante das soluções e são consideradas incomparáveis na perspectiva de múltiplas funções objetivo. Estas soluções são chamadas de não dominadas ou soluções do *Pareto-ótimo*. Cada uma delas apresenta resultados, para pelo menos um dos objetivos, melhor do que os resultados das soluções restantes.

Para facilitar o entendimento do conceito de dominância seguem algumas definições:

Definição 1. Dados dois vetores $\vec{x}, \vec{y} \in \mathbb{R}^k$, se diz que $\vec{x} \leq \vec{y}$ se $x_i \leq y_i$ para todo $i = 1, \dots, k$, e que $\vec{x}$ ***domina*** $\vec{y}$ ($\vec{x} \prec \vec{y}$) se $\vec{x} \leq \vec{y}$ e $\vec{x} \neq \vec{y}$. A Figura 4 mostra um caso de relação de dominância em problemas de minimização com dois objetivos f_1 e f_2.

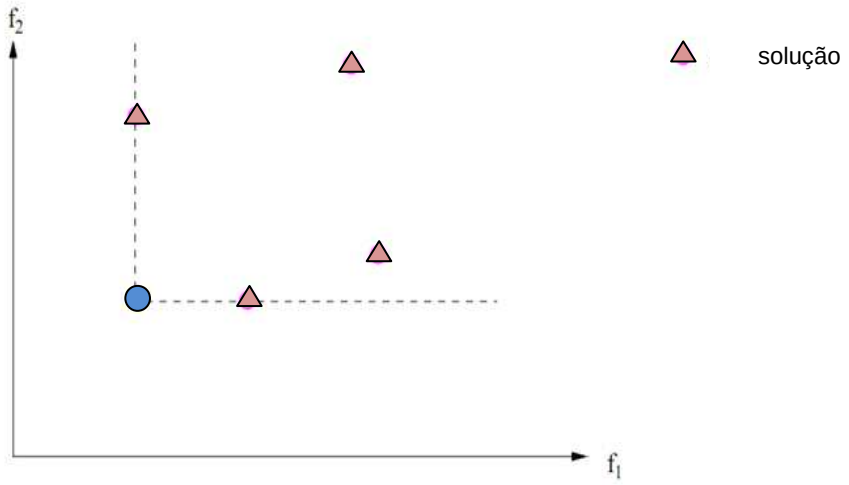

Figura 4. Relação de dominância em um espaço bi-objetivo.

Definição 2. Se diz que um vetor de variáveis de decisão $\vec{x} \in X \subset \mathbb{R}^n$ é **não dominado** com respeito a X, se não existe outro $\vec{x}' \in X$ tal que $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$.

Definição 3. Diz-se que um vetor de variáveis de decisão $\vec{x}^* \in F \subset \mathbb{R}^n$ (F é uma região alcançável) é considerado ***Pareto-ótimo*** se $\vec{x}^*$ não é dominado em relação a F.

Definição 4. O conjunto de soluções do ***Pareto-ótimo*** P^* é definido por:

$$P^* = \{\vec{x} \in F \mid \vec{x} \text{ é um } Pareto - ótimo\}.$$

Definição 5. O *Pareto Front* PF^* é definido por:

$$PF^* = \left\{\vec{f}(\vec{x}) \in \mathbb{R}^k \mid \vec{x} \in P^*\right\}.$$

A **Figura 5** mostra um caso particular de *Pareto Front* para problemas de dois objetivos.

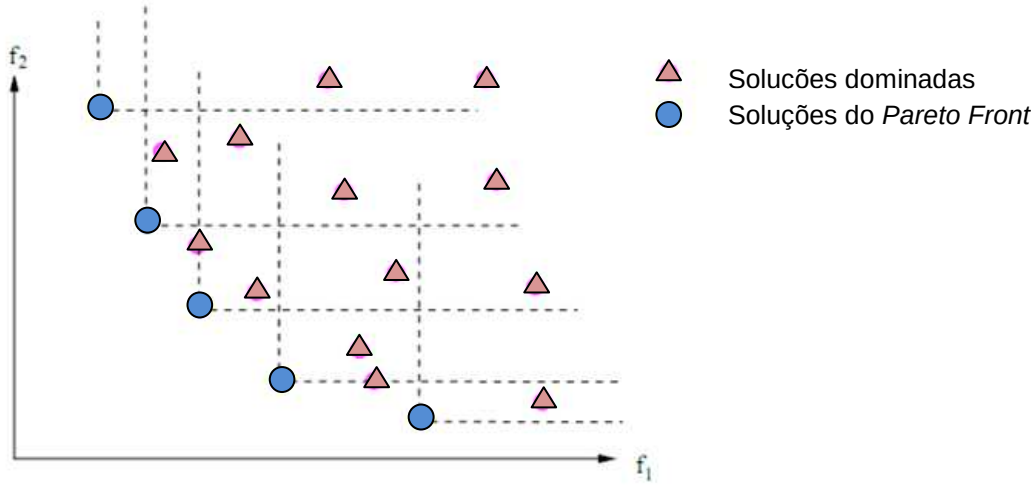

Figura 5. Conjunto de soluções do *Pareto Front.*

Cada problema de múltiplos objetivos apresenta um *Pareto-ótimo* associado, que representa o conjunto de soluções ótimas que melhor o soluciona. Deste modo, um processo de otimização multi-objetivo deve seguir duas premissas. A primeira é determinar o conjunto de soluções do *Pareto Front* a partir do conjunto F de todos os vetores de variáveis de decisão que satisfaçam (7) e (8); de modo que este se assemelhe com o *Pareto-ótimo* esperado, já que matematicamente, o *Pareto-ótimo* é a imagem do conjunto de soluções do *Pareto Front* no espaço de objetivos [1].

A segunda premissa consiste em encontrar soluções que apresentem diversidade, ou seja, que sejam bem distribuídas entre os objetivos. Na prática,

nem todo conjunto de soluções do *Pareto Front* é normalmente o desejável ou alcançável.

3.2. Métricas para Cálculo de Desempenho

As métricas são equações matemáticas com a função de quantificar a qualidade do *Pareto* gerado após a execução do algoritmo. Serão abordadas quatro métricas neste trabalho: *Hypervolume, Spacing, Maximum Spread* e *Coverage*. Cada uma destas métricas extrai diferentes aspectos do *Pareto* obtido.

3.2.1. Hypervolume

A métrica Hypervolume foi proposta por Zitzler e Thiele [10] e é definido pela área pela área ocupada pelo *Pareto Front* obtido ($\boldsymbol{PF}^*$) (área embaixo da curva). De modo a explicar o conceito, será adotado uma otimização com dois objetivos. Considerando um retângulo limitado pelo ponto $(\boldsymbol{f_1}(\vec{\boldsymbol{x}}); \boldsymbol{f_2}(\vec{\boldsymbol{x}}))$ que pertence ao *Pareto Front* e a origem. Supondo que cada ponto no *Pareto* gera um retângulo no espaço de objetivos, o *Hypervolume* corresponde à área formada pela união de todos os retângulos, como é mostrado na Figura 6.

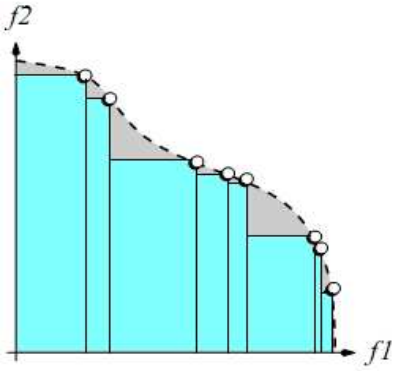

Figura 6. Retângulos necessários para cálculo do Hypervolume.

É possível generalizar a aplicação desta métrica para problemas com *n*-objetivos usando da equação:

$$HV = \left\{ \bigcup_i a_i \mid x_i \in PF^* \right\}, \tag{10}$$

onde x_i é um vetor de soluções não dominadas contido em PF^* e a_i é o *Hypervolume* determinado pelos componentes de x_i e a origem. Sendo *HV* a união de todos os *hypervolumes* calculados.

3.2.2. Spacing

O objetivo é medir a variância da distância entre as soluções não dominadas que são adjacentes e que pertencem ao *Pareto Front*. O cálculo desta métrica é realizado usando-se a seguinte equação

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\bar{d} - d_i\right)^2}, \tag{11}$$

onde $d_i = min_j\left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right), \quad i,j = 1,\dots,n,\ j \neq i;\ \bar{d}$ é a distância média entre todas as soluções adjacentes e n é o número de soluções não dominadas do *Pareto Front*. O valor igual a zero significa que todas as soluções do *Pareto Front* estão espaçadas de forma eqüidistante.

3.2.3. Coverage

Esta métrica foi proposta por Zitzler e Thiele [10] [11] [12] e realiza o mapeamento do par ordenado (A, B), sendo A e B dois *paretos*, no intervalo $[0, 1]$ de acordo com

$$C(A,B) = \frac{|\{b \in B;\ \exists a \in A; a \geq b\}|}{|B|}. \tag{12}$$

O valor de $C(A,B) = 1$ significa que todas as soluções do *Pareto Front* B são fracamente dominadas pelas soluções do *Pareto Front* A. Por outro lado,

$C(A, B) = 0$ significa que nenhuma das soluções do *Pareto Front* B é fracamente dominada pelas soluções do *Pareto Front* A. Esta métrica é interessante, pois permite a comparação entre *paretos* gerados por algoritmos diferentes, baseando-se na dominância entre suas soluções.

3.2.4. Maximum Spread

Proposta por Zitzler [11], esta técnica calcula a máxima extensão abrangida pelas soluções não dominadas do *pareto*. Usando problema com dois objetivos, o valor do *Maximum Spread* corresponde à distância euclidiana entre as duas soluções mais distantes. Esta métrica é calculada de acordo com

$$MS = \sqrt{\sum_{m=1}^{M} \left(max_{i=1}^{n} f_m^i - min_{i=1}^{n} f_m^i \right)^2}, \quad (13)$$

onde n é número de soluções no *Pareto Front* e M é o número de objetivos em um dado problema. Valores de *Maximum Spread* maiores indicam melhor desempenho e maior diversidade [4].

Capítulo 4

Abordagens de PSO para MOP

Este capítulo tem como objetivo mostrar algumas abordagens da técnica de otimização baseada em enxames em problemas de múltiplos objetivos. Serão apresentados algoritmos que obtiveram resultados significativos com esta nova idéia; entre eles o MOPSO-CDR que será estudado mais profundamente, pois, é nele que este trabalho é fundamentado.

4.1. PSO aplicado a Problemas Multi-Objetivo

Como foi visto no Capítulo 2, o PSO é uma técnica cujo objetivo é solucionar problemas de único objetivo; de modo que para aplicá-lo em MOP o seu formato original deve ser alterado.

No contexto de problemas com vários objetivos existem três pontos a serem alcançados [11]:

1. Maximizar o número de elementos pertencentes ao *Pareto*.
2. Minimizar a distância do *Pareto Front* produzido pelo algoritmo em relação ao verdadeiro *Pareto Front* do problema.
3. Maximizar o espalhamento das soluções encontradas, de modo a se obter um vetor de soluções uniforme e bem distribuído.

Ao se utilizar o PSO neste novo contexto, surgem algumas dúvidas técnicas referentes à adaptação que deve ser feita ao seu algoritmo [13]:

1. Como selecionar partículas para serem usadas como líderes, de modo a dar preferência às soluções não dominadas?
2. Como armazenar as soluções não dominadas encontradas durante o processo de busca, de modo a reportar as soluções não dominadas de todo o passado?

3. Como manter diversidade no enxame de modo a evitar convergência para uma única solução?

No PSO, o líder que cada partícula utiliza para atualizar sua posição é determinado através da relação de vizinhança entre as partículas. No caso de problemas de otimização multi-objetivo, cada partícula teria um conjunto de diferentes líderes a disposição, do qual apenas um seria selecionado para atualizar a sua posição. O conjunto de líderes é armazenado em um Arquivo Externo. O AE é um repositório das soluções não dominadas encontradas durante o processo de busca, e ao se chegar no ponto de parada do algoritmo, as soluções nele contidas são as soluções que compõe o *Pareto Front*.

A Figura 7 apresenta o pseudo-código do algoritmo genérico da aplicação de PSO em MOP. Os processos que foram adicionados ao algoritmo do PSO estão marcados em *itálico*.

Após o enxame ser inicializado, o conjunto de soluções não dominadas, proveniente do enxame, é inserido no arquivo externo. Baseado em alguma política de qualidade, como por exemplo, o *Crowding Distance* e roleta, é selecionado um líder para cada partícula do enxame, sendo utilizado com $gbest$ na equação (1), e este passo se repete a cada iteração.

Outro detalhe importante acontece na atualização do $pbest$ das partículas. No caso do PSO a atualização acontece da seguinte forma, no caso de problemas de minimização:

$$se\ fitness(x_i) < fitness(pbest_i).$$

Já nesta proposta, o $pbest$ só é atualizado se a nova posição dominar o $pbest$ armazenado. Após a atualização da posição e velocidade do enxame, o algoritmo se repete até o ponto de parada.

```
Início:
    Inicializar enxame
    Inicializar líderes no Arquivo Externo
    Qualificar líderes
    g = 0
    Enquanto g < gmax:
        Para cada partícula:
            Selecionar líder
            Atualizar Posição
            Calcular Fitness
            Atualizar pbest
        Atualizar líderes no Arquivo externo
        Qualificar líderes
        g++
    Reportar resultados
Fim
```

Figura 7. Pseudo-código do algoritmo genérico do PSO em MOP.

4.2. Trabalhos Relacionados

A área de MOP combinada com PSO possui, basicamente, dois campos que são bastante pesquisados e que apresentam bastante influência no resultado final [3]:

1. Seleção e atualização de líderes;
2. Geração de diversidade.

Nesta seção serão mostradas algumas técnicas que propõem idéias e melhorias nestas duas áreas citadas.

4.2.1. CSS-MOPSO

Este algoritmo foi proposto por Chiu [14] e apresenta uma peculiaridade, que é a ausência do componente cognitivo da equação (1) e a presença de dois componentes sociais.

A seleção do $gbest_1$ é realizada se baseando no ângulo θ entre a *datum line*, linha que conecta o membro do arquivo e um *datum point* c (ponto formado pela intersecção das linhas perpendiculares de duas soluções extremas provenientes do conjunto de soluções), e a linha que conecta a partícula e o membro do arquivo externo que está sendo verificado. O membro do arquivo externo que apresente o menor ângulo com a partícula será alocado como seu $gbest_1$. A Figura 8 (a) ilustra o processo de escolha do $gbest_1$ descrito anteriormente.

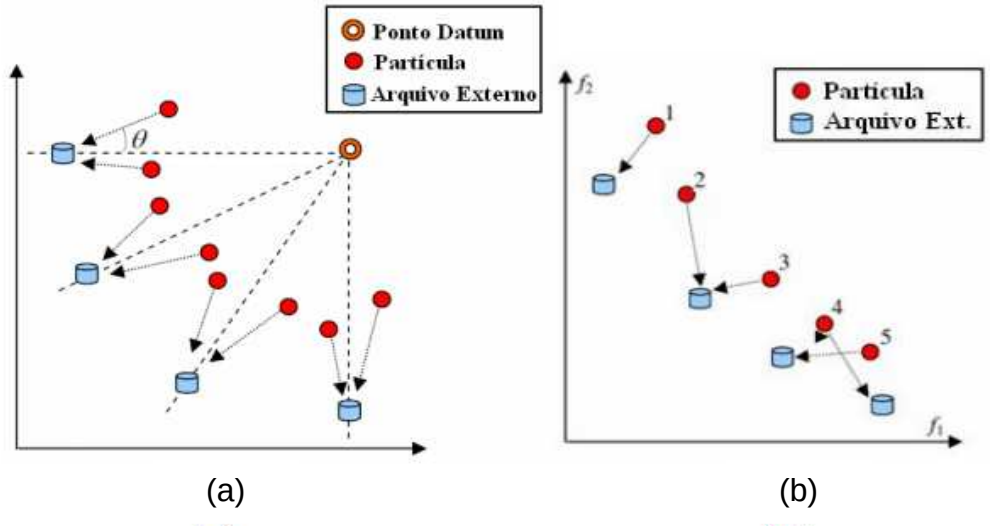

Figura 8. Estratégia de escolha do (a) $gbest_1$ e (b) $gbest_2$. Figura adaptada de Chiu [14].

A seleção do $gbest_2$ é feita de acordo com a escolha de uma das soluções do vetor de soluções para um objetivo selecionado aleatoriamente f_i em cada iteração. Todas as partículas são ordenadas pelo seu valor de *fitness de* f_i. Após isso, a cada partícula será dado um número serial. Para todas as partículas pares, o membro do arquivo mais próximo delas cujo valor de *fitness* de f_i é maior que o da partícula, será alocado como o $gbest_2$ da partícula. O mesmo acontece para as partículas ímpares, o membro do arquivo mais próximo delas cujo valor

de *fitness* de f_i é menor que o da partícula, será alocado como o $gbest_2$ da partícula. A Figura 8 (b) ilustra a estratégia de escolha do $gbest_2$ para o objetivo f_1.

4.2.2. MOPSO

Este algoritmo foi proposto por Coello [15], e é baseado na idéia de se ter um arquivo externo em que cada partícula depositará suas experiências a cada iteração. Nesta proposta, o espaço de objetivos explorado é dividido em hipercubos. Cada hipercubo recebe um valor de $fitness$ que depende do número de partículas inseridas nele.

A seleção do líder social de cada partícula é feita através de uma técnica chamada *Roulette Wheel*. Através dela, é escolhido o hipercubo, e dentro deste, escolhe-se uma partícula de forma aleatória, que esteja dentro do seu escopo para ser a líder. Nesta técnica também há o operador de mutação aplicado às posições das partículas.

4.2.3. MOPSO-CDLS

Proposta por Tsou [16] e baseada na proposta de Raquel [17], esta técnica utiliza o conceito de *Crowding Distance* (CD), utilizando-o como mecanismo para selecionar líderes do arquivo externo. O CD avalia o quão distante as partículas estão umas das outras, isso se torna importante na análise de como estão distribuídas as soluções. Na Figura 9, são mostradas regiões com partículas em seu interior. As partículas que se localizam em regiões mais populosas possuem CD maior, ou seja, uma maior quantidade de partículas próximas.

Existem duas possíveis situações na escolha dos líderes social e cognitivo: 1) o líder social é aleatoriamente escolhido entre 10% das soluções com maior *Crowding Distance,* se a partícula é dominada por pelo menos uma das soluções; caso contrário, o líder é escolhido aleatoriamente utilizando todas as soluções do arquivo externo.

2) O líder cognitivo de cada partícula é atualizado se a nova posição domina o atual $pbest$, se forem incomparáveis, uma das duas é escolhida aleatoriamente.

Esta proposta utiliza um mecanismo de busca local no Arquivo Externo com o objetivo de melhorar as habilidades de busca em amplitude e ainda aumentar a convergência.

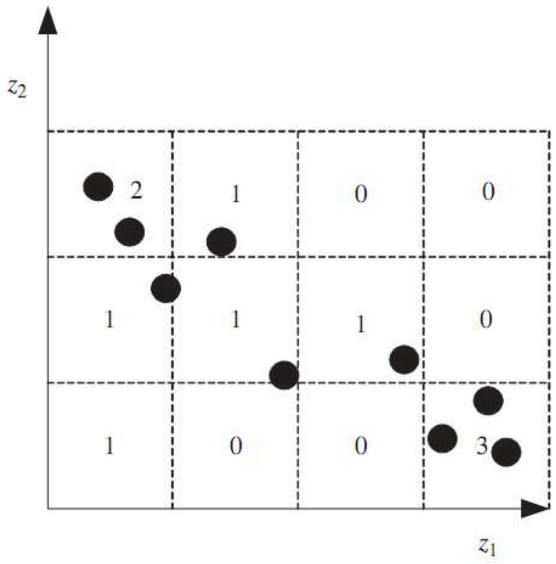

Figura 9. Distribuição das soluções no Arquivo Externo.

4.2.4. m-DNPSO

Este algoritmo foi proposto por Hu e Eberhart [18], e tem como característica a solução de um objetivo a cada passo usando um esquema similar à da ordenação lexográfica [19].

Os múltiplos objetivos são divididos em dois grupos: f_1 e f_2. f_1 é definido como o objetivo da vizinhança e f_2 como o objetivo de otimização. A seleção destes grupos é arbitrária.

Para selecionar o líder social, o algoritmo tem que calcular a distância da $partícula_i$ para todas as outras partículas considerando f_1. Usando esta informação, as m partículas mais próximas são selecionadas, onde m é o tamanho da vizinhança. Finalmente, o líder social é definido pela melhor solução em termos de f_2 valores entre m vizinhos.

O líder cognitivo é atualizado apenas quando uma nova solução domina sua atual posição. Nesta proposta o Arquivo Externo também é responsável pelo armazenamento de soluções não dominadas.

Esta apresenta dois pontos negativos: a ordenação lexográfica tende a ser útil apenas no escopo de apenas dois objetivos, além de que a ordem na escolha dos objetivos pode influenciar na performance do algoritmo.

Na próxima seção será abordado outro algoritmo, MOPSO-CDR, que também combina a técnica PSO com MOP, porém, diferente dos citados acima, este será estudado com mais detalhes, pois é nele que se fundamenta este trabalho.

4.3. Detalhamento do MOPSO-CDR

Este algoritmo foi desenvolvido por Santana, Pontes e Bastos-Filho [4], e é fundamentado no algoritmo MOPSO. Além disso, o mesmo incorpora técnicas como *Crowding Distance* e *Roulette Wheel*, vistas nas seções anteriores, para auxiliar nos processos de escolha do líder social ($gbest$) e para prevenir um número excessivo de soluções não dominadas no Arquivo Externo. Além disso, o MOPSO-CDR apresenta um novo procedimento na atualização do líder cognitivo ($pbest$). O pseudocódigo do algoritmo desta proposta é mostrado a seguir:

```
Início:
    Inicializar líderes no Arquivo Externo
    Qualificar Líderes usando Crowding Distance
    Enquanto Condição de parada não alcançada:
        Para cada partícula:
            Aplicar turbulência [3]
            Selecionar líder (usando-se Crowding Distance e roleta)
            Atualizar velocidade e posição
            Calcular Fitness
            Atualizar pbest (torneio binário)
        Atualizar líderes no Arquivo Externo
        Qualificar líderes aplicando Crowding Distance
    Reportar resultados
Fim
```

Figura 10. Pseudo-código do algoritmo do MOPSO-CDR.

4.3.1. Seleção do Líder Social

Em problemas com múltiplos objetivos a escolha apropriada do líder social é de fundamental importância, pois afeta a capacidade de convergência e distribuição das soluções ao longo do *pareto*.

Todas as soluções não dominadas estão presentes no Arquivo Externo, e estas são as possíveis candidatas a serem utilizadas como $gbest$ de alguma partícula do enxame. O MOPSO-CDR ordena, antes de cada iteração, as soluções presentes no arquivo externo utilizando como parâmetro o *CD.*

Como cada partícula do enxame realiza sua atualização de velocidade e posição, faz-se necessária a seleção do líder social, sendo esta realizada por meio da aplicação da *roleta*, onde as soluções do arquivo externo com menor CD apresentam mais chances de serem selecionadas.

4.3.2. Seleção do Líder Social Cognitivo

A regra que define a substituição do $pbest$ ou líder cognitivo também é bastante importante para a convergência e eficiência do algoritmo [4]. Foi desenvolvido no MOPSO-CDR uma nova estratégia para este processo. De acordo com a nova proposta, o $pbest$ só é atualizado se a posição atual da partícula dominá-lo; se a relação entre eles for incomparável, a escolha é feita utilizando-se o Arquivo Externo. O arquivo externo é necessário, pois, neste caso, a idéia é identificar a solução, presente nele, mais semelhante (aplica-se a distância euclidiana) do $pbest$ e mais semelhante do x_i. Deste modo, verifica-se qual das duas soluções apresenta menor *CD*; caso a solução mais próxima da posição atual x_i tenha sido a de menor *CD*, o $pbest$ tem seu valor atualizado para esta posição, caso contrário o $pbest$ é mantido.

4.3.3. Turbulência

PSO apresenta como principal característica o fato de apresentar alta velocidade de convergência. Porém, este comportamento pode ser prejudicial para otimização multi-objetivo, pois o algoritmo pode convergir para um falso

Pareto Front. O operador de mutação pode ajudar a evitar este problema através do aumento da habilidade exploratória das partículas [4].

O operador utilizado neste algoritmo é o mesmo utilizado no MOPSO [15] e é aplicado a cada iteração com influência limitada. No início, todas as partículas são afetadas, porém com o passar das iterações esta interferência diminui.

4.3.4. Arquivo Externo

Como foi visto nas seções anteriores, o principal objetivo do arquivo externo é atuar como repositório de soluções não dominadas que foram encontradas ao longo do processo de busca.

De modo a evitar excesso de soluções armazenadas no AE, decisões devem ser tomadas quanto à sua inserção e remoção. Deste modo, faz-se uso de um *Gerenciador* do processo de atualização do arquivo externo. A Figura 11 mostra todos os possíveis casos que podem ocorrer no repositório.

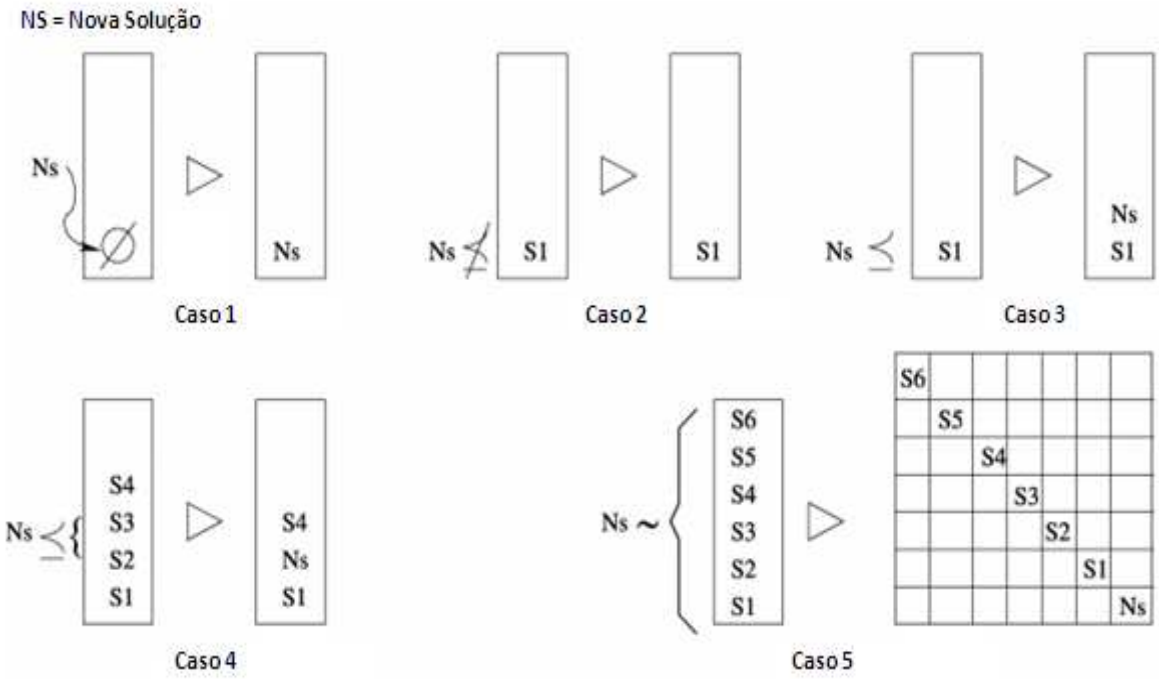

Figura 11. Casos possíveis para o AE.

O Caso 1 mostra o momento em que uma nova solução *Ns* entra no AE vazio. No Caso 2, o arquivo externo possui uma solução *S1*, e a nova solução *Ns*, pelo fato de ser dominada por *S1* não é adicionada ao repositório. Já o Caso 3 mostra que a solução candidata *Ns* é incomparável em relação *S1*, logo, *S1* é mantida no repositório e *Ns* adicionada. O Caso 4 mostra que a *Ns* é incomparável em relação às soluções *S1* e *S4,* e domina as soluções *S2* e *S3* retirando-as do arquivo externo. O último caso apresenta a situação em que *Ns* é incomparável a todas as soluções do arquivo externo, deste modo, foi inserida no arquivo externo, apresentado em duas dimensões para exemplificar o *Pareto* formado com as soluções presente.

O *Gerenciador*, ao receber uma demanda de soluções a serem adicionadas, deve analisar se estas não são dominadas pelas soluções do arquivo externo, as candidatas que forem dominadas não serão adicionadas, e as soluções presentes no Arquivo Externo que forem dominadas serão removidas.

O Arquivo Externo apresenta um limite quanto ao número de soluções que devem habitá-lo, deste modo, se o número exceder, aquelas que possuírem menor *CD* serão removidas do repositório.

Capítulo 5
A Nova Abordagem, MOPSO-CDR com Especiação

Este capítulo tem o objetivo de apresentar a proposta deste trabalho o MOPSO-CDR com especiação (MOPSO-CDRS), uma abordagem nova fundamentada no algoritmo MOPSO-CDR, analisado anteriormente.

5.1. Alterações Propostas

Para a construção do trabalho foi utilizada a base do algoritmo MOPSO-CDR. A filosofia deste algoritmo quanto à seleção de líderes social e cognitivo, utilizando conceitos de *Crowding Distance* e *roleta,* gerou resultados bastante positivos.

Com o objetivo de melhorar ainda mais os resultados obtidos pelo MOPSO-CDR, foram incorporados novos passos ao algoritmo.

Uma das mudanças propostas foi a inserção de um *Gerenciador* de arquivo externo. Este *Gerenciador* avalia como as soluções estão se comportando com o passar das iterações. Como foi visto no Capítulo 3, as métricas de cálculo de desempenho geram informação a respeito da distribuição e espaçamento das soluções presentes no Arquivo Externo.

Outra mudança proposta foi a criação de um tomador de decisões baseado no desempenho do *pareto*. Este é responsável pela análise das informações geradas pelo *Gerenciador do Arquivo Externo* e por tomar decisões de modo a tentar melhorar o desempenho na busca de melhores soluções para os objetivos.

Estas decisões estão relacionadas com a forma pela qual a seleção dos líderes será realizada, e nesta abordagem existem duas:

1. Seleção de líderes adotando forma *básica*, mesma que no MOPSO-CDR.

2. Seleção de líderes adotando a Especiação.

A primeira opção mantém a forma como é feita a seleção dos líderes social e cognitivo do MOPSO-CDR, citado nas seções 4.3.1 e 4.3.2 respectivamente.

A segunda opção divide o enxame, de modo que cada sub-enxame tenha uma responsabilidade específica. A proposta de particionamento do enxame tem o objetivo de dividir tarefas; supondo que haja n objetivos a serem solucionados, serão criados $n+1$ sub-enxames com o mesmo número de partículas. O sub-enxame adicional se comportará baseado no MOPSO-CDR, enquanto cada um dos n restantes será responsável em solucionar um objetivo. A Figura 12 mostra o processo de especiação do enxame para o caso de dois objetivos f_1 e f_2, com um total de 21 partículas. Como foi dito anteriormente, cada sub-enxame terá o mesmo número de partículas, logo, para este caso, cada sub-enxame apresenta 7 partículas.

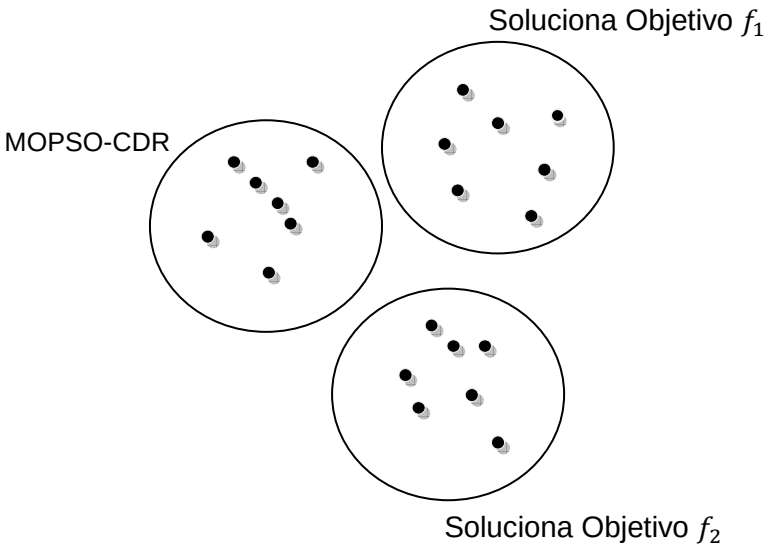

Figura 12. Ilustração do processo de Especiação do enxame.

Esta mudança interferiu diretamente na seleção dos líderes. Nas seções 5.2 e 5.3 serão mostradas, com detalhes, as alterações implementadas na escolha dos líderes no caso com Especiação e na seção 5.4 serão apresentados o *Gerenciador* do Arquivo Externo, e o processo para a tomada de decisões.

5.2. Seleção do Líder Social

Na sessão anterior, foi mencionado que o MOPSO-CDRS apresenta uma entidade responsável pela tomada de decisões: manter o funcionamento normal do MOPSO-CDR ou utilizar a especiação. Caso a primeira seja escolhida, o processo de seleção dos líderes social e cognitivo é mantido. Caso contrário, o líder social deixa de ser escolhido através de CD e roleta, e passa a utilizar especiação.

Lembrando que o $gbest$ é utilizado no processo de atualização de velocidade e posição das partículas; se a partícula em questão pertence a um agrupamento cujo objetivo é otimizar o objetivo f_1, será buscada, no Arquivo Externo, a solução que melhor soluciona f_1, sendo esta utilizada como líder social da partícula em questão. A Figura 13 mostra a idéia da seleção do líder social, de modo que a solução que apresenta o melhor resultado para grupo de partículas responsável pelo objetivo f_1 foi escolhida.

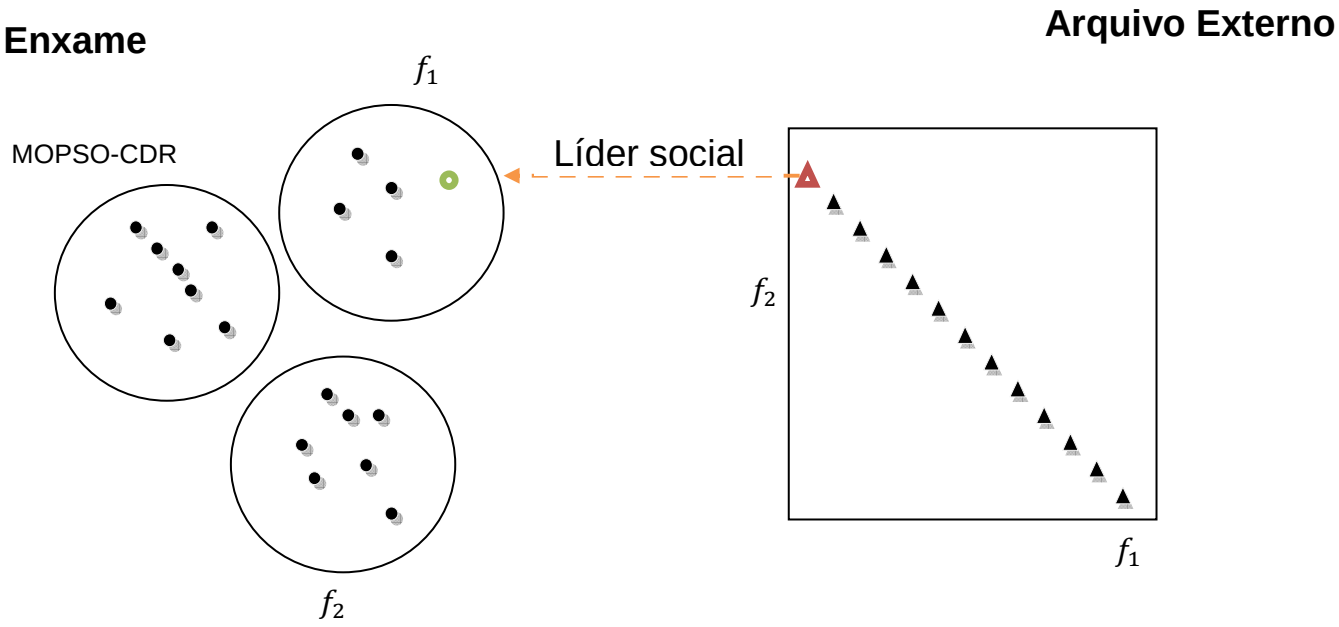

Figura 13. Seleção do líder social com *Especiação.*

5.3. Seleção do Líder Cognitivo

O líder cognitivo ou $pbest$ também é utilizado na atualização de velocidade e posição das partículas, porém, a forma como este deve ser atualizado é o diferencial. O MOPSO-CDRS propõe a seguinte mudança.

Caso o *fitness* do $pbest$ e o *fitness* da posição atual da partícula analisada sejam considerados incomparáveis, verifica-se qual das duas apresentam o menor *fitness* para o objetivo atribuído ao grupo em que a partícula está inserida. A Figura 14 mostra um exemplo onde se tem os valores de *fitness* do $pbest$ e da $partícula\ i$. Caso a partícula pertença a um sub-enxame cuja responsabilidade é solucionar o objetivo f_1, de acordo com a figura, como o *fitness* de seu $pbest$ apresenta um melhor valor para este objetivo, logo, este é mantido. Caso o objetivo a ser solucionado fosse o f_2, o *fitness* da $partícula_i$ é melhor que o do $\overrightarrow{pbest_{(t)}}$. Neste caso, a posição armazenada no $\overrightarrow{pbest_{(t)}}$ seria substituída pela posição atual da partícula.

	f_1	f_2		f_n
$\overrightarrow{pbest_{(t)}}$	2,0	3,4	...	5,7
$\overrightarrow{x_i}$	2,1	3,1	...	4,3

Figura 14. *Fitness* do $\overrightarrow{pbest_{(t)}}$ e *fitness* da posição atual da $partícula\ i$.

5.4. Analisando o Arquivo Externo e Tomando Decisões

Existe uma série de requisitos que devem ser atendidos ao se retornar um *Pareto Front* no final da execução do algoritmo. Com o uso das métricas de desempenho, torna-se fácil a quantização destas características. Deste modo, tentando-se aperfeiçoar as soluções do Arquivo Externo, foi desenvolvido o *Gerenciador de Arquivo Externo*, sendo este responsável por supervisionar o

comportamento das soluções através da aplicação das métricas de desempenho a cada iteração.

Além do *Gerenciador*, foi criado um *Executor ou Tomador de Decisões*. Esta aplicação recebe a análise feita pelo *Gerenciador* e verifica qual decisão deve ser tomada para que se possa melhorar a convergência do *pareto*.

Como foi dito na seção 5.1, o *Executor* é responsável pela escolha do estado em que o algoritmo deve estar em um dado momento; podendo migrar entre dois estados: 1) o estado *Básico* utilizado pelo MOPSO-CDR e 2) o estado de *Especiação.*

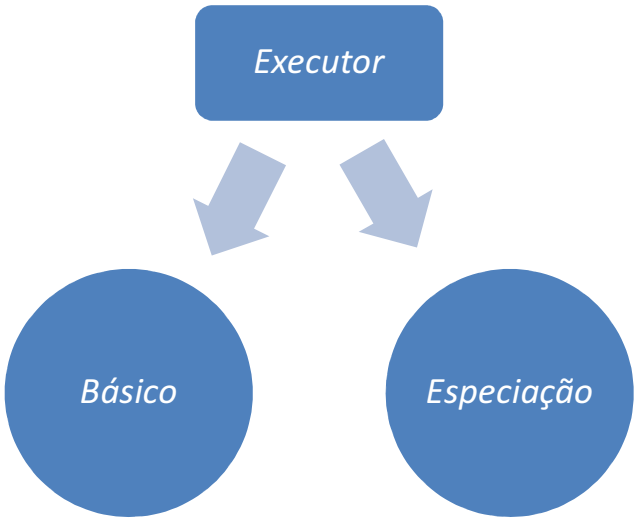

Figura 15. O *Executor* decide entre os estados: (1) MOPSO-CDR ou *Básico*, e (2) *Especiação* para fazer a seleção dos líderes.

A tomada de decisão do *Executor* é influenciada pelos valores obtidos através da aplicação das métricas *Spacing* e *Maximum Spread*, estudadas no Capítulo 3. Portanto, no caso do estado *Básico*, o *Gerenciador do Arquivo Externo* analisa o *spreading* do *pareto*; caso a variação do resultado desta métrica seja considerada insignificante, ou seja, não apresente melhora com o passar das iterações, o *Executor* muda para o estado de *Especiação*. Neste estado, o *Gerenciador* passa a analisar o *spacing* do *Pareto*, caso este, em um dado momento, também apresente estagnação na variação, o *Executor* muda para o estado *Básico*.

A Figura 16 mostra o aspecto que deve ser analisado pelo *Gerenciador* quando o algoritmo adota o estado *Básico*. No início da análise, o *Pareto* apresenta deficiências quanto ao *spreading* mostrado na Figura 16 (a), porém, com o passar das iterações o *Pareto* tende a um melhor espalhamento, mostrado na Figura 16 (b).

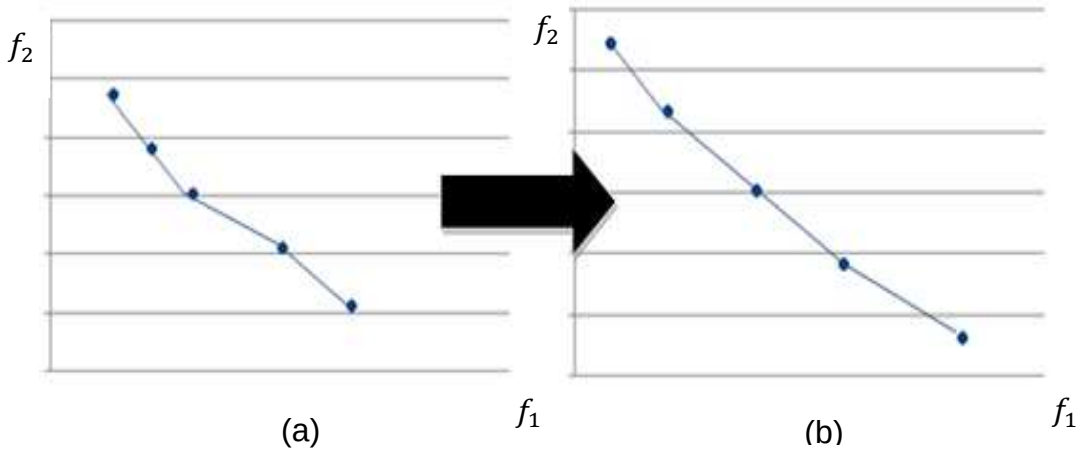

Figura 16. No estado *Básico*, o *Gerenciador* tem o objetivo de analisar o *pareto* quanto ao *spreading* de modo a auxiliar na sua convergência. (a) Representa o *Pareto* com pouco espalhamento, já (b) mostra o *Pareto* com melhor grau de convergência em relação ao *spreading.*

A Figura 17 encaixa-se no momento em que o algoritmo está no estado de *Especiação*. No início, o *Pareto* apresenta um espaçamento desuniforme entre as soluções, como mostra a Figura 17 (a). À medida que novas soluções vão sendo encontradas e adicionadas ao AE, a uniformidade entre as soluções se torna cada vez maior, aperfeiçoando cada vez mais o *Pareto* gerado. A Figura 17 (b) mostra um *Pareto* com soluções espaçadas uniformemente.

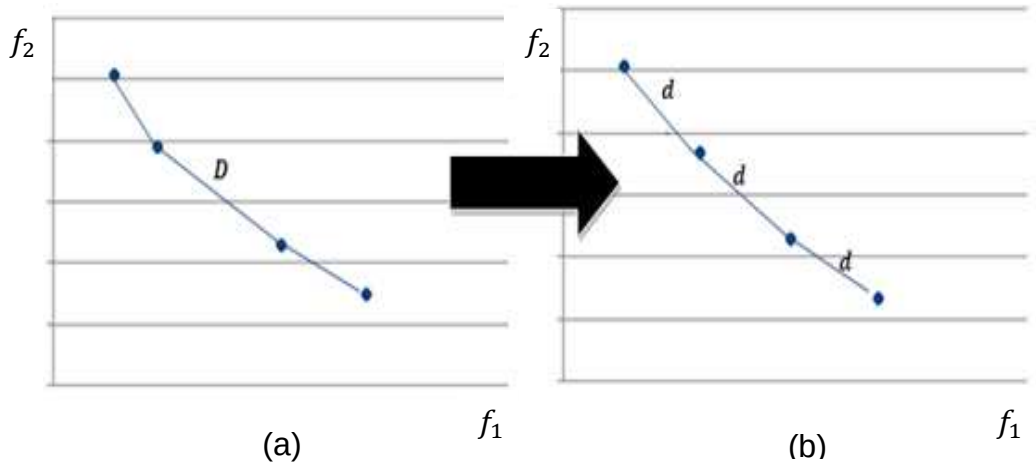

Figura 17. No estado *Especiação*, o *Gerenciador* tem o objetivo de analisar o *pareto* quanto ao *spacing* de modo a auxiliar na sua convergência. (a) Representa o *Pareto* com espaçamento irregular entre as soluções, já (b) mostra a regularidade de espaçamento entre as soluções do *Pareto,* ou seja, melhor taxa de *spacing.*

Para se definir o percentual de estagnação das soluções aplicou-se o desvio padrão sobre os resultados gerados pelo *Gerenciador* para uma dada métrica. De forma empírica, foi determinado o *limiar de estagnação* igual a 0,1%, ou seja, se o desvio padrão gerado for menor que este limiar, o *Executor* muda o estado do algoritmo.

Suponha o seguinte exemplo: O estado atual do sistema é *Básico* e este estado está sendo mantido a 1000 iterações. Como no estado *Básico* o *Gerenciador* é responsável pela análise do *Maximum Spreading*, para cada iteração foi gerado um valor que representa o *spreading* do *Pareto* naquele momento. O *Executor* aplica o desvio padrão sobre esses 1000 valores de *spreading* gerados e verifica se o resultado é menor que o *limiar de estagnação*, caso seja menor o *Executor* muda o estado do algoritmo para *Especiação*. Deste modo, o *Gerenciador* passa a analisar o *spacing* do *pareto* e o mesmo processo se repete.

5.5. Algoritmo

Como foi dito nas seções anteriores, novos processos foram adicionados à rotina do MOPSO-CDR. De acordo com a Figura 18, percebem-se as mudanças, em *itálico*, que foram efetuadas no algoritmo base.

Início:
Inicializar enxame
Inicializar líderes no Arquivo Externo
Qualificar líderes usando *CD* e roleta
Utilizar estado Básico (1)
Enquanto Critério de parada não alcançado:
Para cada partícula:
Aplicar turbulência
Selecionar líder social (de acordo com estado)
Atualizar velocidade e posição
Calcular Fitness
Atualizar pbest (de acordo com estado)
Atualizar líderes do Arquivo Externo
Qualificar líderes usando *CD*
Analisar pareto usando métrica
Atualizar estado
Reportar resultados
Fim

Figura 18. Pseudo-código do algoritmo do MOPSO-CDRS.

A primeira mudança a ser analisada é $Utilizar\ estado\ Básico$. O algoritmo necessita de um estado inicial para que a escolha do líder seja fundamentada, deste modo, foi definido como estado *default* o estado *Básico*. Outros dois pontos bastante importantes que sofreram mudanças foram: 1) $Selecionar\ líder\ social$ e 2) $Atualizar\ pbest$; estes dependem da escolha feita pelo *Executor* em relação a que estado será utilizado: *Básico* ou *Especiação.*

Outra novidade desta proposta é apresentada nos passos $Analisar\ pareto\ usando\ métricas$ e $Atualizar\ estado$. As métricas de análise de *pareto* costumam ser utilizadas no fim do algoritmo para verificar as características do *pareto* front gerado. No algoritmo proposto, a cada iteração, o *pareto* gerado é avaliado (aplicando-se métricas) de modo que este possa ser ajustado.

Supondo que o estado atual aplicado seja o *Básico*, deste modo, aplica-se a métrica respectiva, a cada iteração, e analisa-se o comportamento das soluções em relação ao *spreading*. O *Executor* supervisiona a melhora da taxa de *spreading* de forma incremental; ao perceber que esta taxa apresenta uma variação insignificante, ou seja, menor que 0,1%, o mesmo aciona a mudança de estado.

No estado *Especiação*, executam-se os mesmos passos citados no estado *Básico*, porém, o objetivo é analisar o *spacing* do *Pareto*. Caso a taxa de *spacing* sature, o *Executor* aciona a mudança de estado.

Capítulo 6

Experimentos

Este capítulo visa apresentar o arranjo experimental e a análise dos resultados obtidos utilizando o MOPSO-CDRS. Também é realizada comparação com outras técnicas, citadas na sessão 4.2. Serão descritos os detalhamentos dos arranjos experimentais, as funções de teste, métricas utilizadas para avaliação de desempenho do arquivo externo, parâmetros de simulação e por fim a apresentação e discussão dos resultados.

6.1. Arranjo Experimental

6.1.1. Funções de Teste

O trabalho desenvolvido por Deb [20] auxiliou na identificação de características que podem causar dificuldades no processo de convergência do *Pareto Front* e na manutenção da diversidade da população. Ao todo foram identificadas seis características: 1) multi-modalidade, 2) influência de mínimos e máximos locais, 3) ponto ótimo isolado, sendo estas três bastante conhecidas no campo de problemas de único objetivo, 4) convexidade ou não convexidade, 5) descontinuidade, 6) não uniformidade.

Para cada uma das seis características mencionadas, uma função de teste correspondente foi criada seguindo a orientação proposta por Deb [20].

De modo a facilitar a explicação das funções de teste, o estudo foi restrito a dois objetivos, já que este número de objetivos é suficiente para refletir os aspectos essenciais de otimização multi-objetivo. Além disso, apenas problemas de minimização são considerados.

As funções de teste são estruturadas obedecendo à mesma regra, sendo esta composta por três funções f_1, g, h [8]:

$$\text{Minimizar } \mathcal{T}(x) = [f_1(x_1), f_2(x)]; \tag{14}$$

Sujeito a $f_2(x) = g(x_2, \dots, x_m)h(f_1(x_1), g(x_2, \dots, x_m))$; (15)

onde $x = (x_1, \dots, x_m)$. (16)

A função f_1 é uma função da primeira variável de decisão apenas, g é uma função que utiliza as $m - 1$ variáveis restantes, e os parâmetros de h são as funções f_1 e g. As funções de teste diferem nestas três funções, no número de variáveis m e nos valores que as variáveis podem receber.

A primeira função de teste é a ZDT1. Sua característica é a presença de um *Pareto Front* com comportamento convexo como mostra a Figura 19.

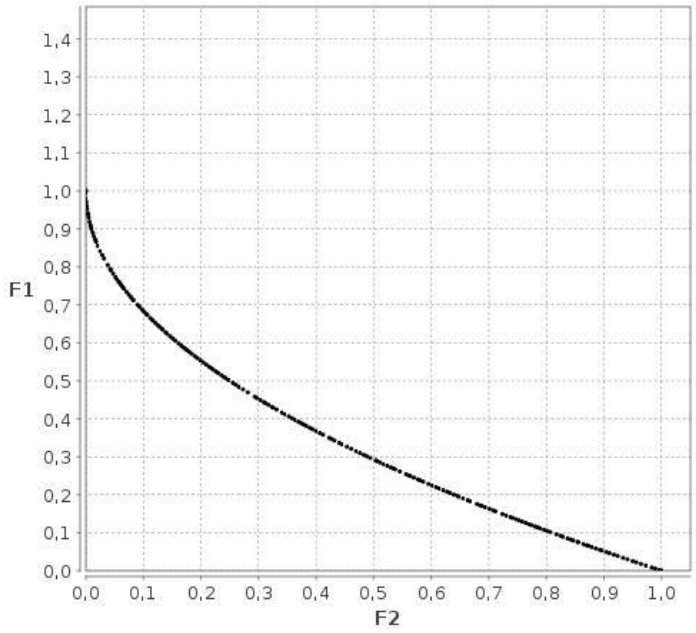

Figura 19. Representação gráfica da função ZDT1.

A ZDT1 é definida matematicamente pelas funções:

$$f_1(x_1) = x_1, \quad (17)$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^{m} x_i/(m-1), \quad (18)$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}. \quad (19)$$

As funções a seguir, exceto ZDT5 e ZDT6, apresentam $m = 30$ e $x_i \in [0, 1]$.

A segunda função é a ZDT2, e diferente da primeira apresenta um *Pareto* côncavo como mostra a Figura 20.

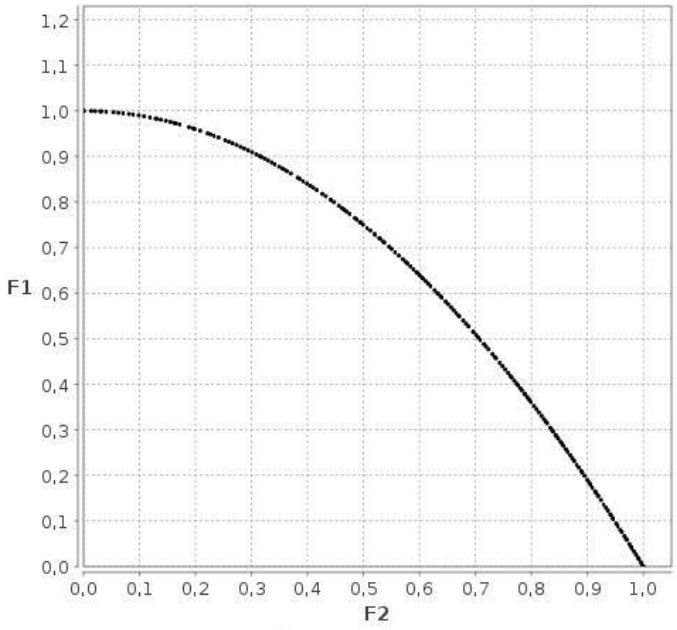

Figura 20. Representação gráfica da função ZDT2.

As funções que a define são mostradas a seguir:

$$f_1(x_1) = x_1, \tag{20}$$

$$g(x_2, \dots, x_m) = 1 + 9.\sum_{i=2}^{m} x_i/(m-1), \tag{21}$$

$$h(f_1, g) = 1 - (f_1/g)^2. \tag{22}$$

A característica de descontinuidade é representada pela ZDT3. Seu *Pareto Front* apresenta uma série de partes convexas não contínuas como mostra a Figura 21.

As funções que a define são mostradas a seguir:

$$f_1(x_1) = x_1, \tag{23}$$

$$g(x_2, \dots, x_m) = 1 + 9.\sum_{i=2}^{m} x_i/(m-1), \tag{24}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1). \tag{25}$$

A introdução da função $seno$ em h causa descontinuidade no *Pareto Front*.

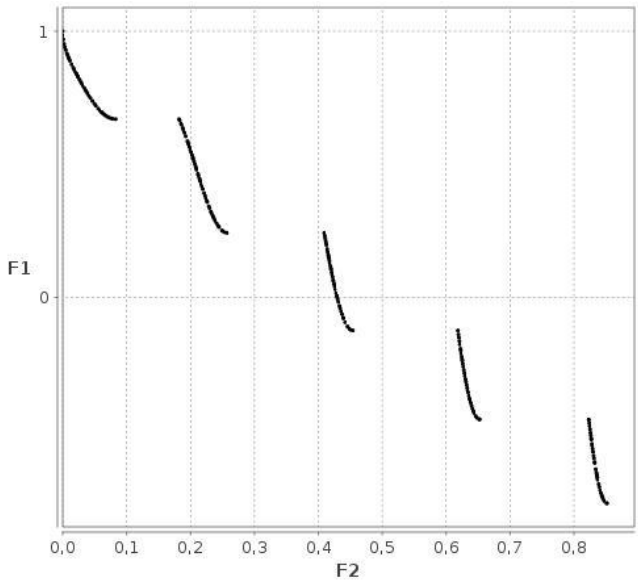

Figura 21. Representação gráfica da função ZDT3.

A função ZDT4 tem a característica de ser multimodal, apresentando 21^9 *paretos* front locais:

$$f_1(x_1) = x_1, \tag{26}$$

$$g(x_2, \dots, x_m) = 1 + 10(m-1) + \sum_{i=2}^{m}(x_i^2 - 10\cos(4\pi x_i)), \tag{27}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}. \tag{28}$$

onde $x_2, \dots, x_m \in [-5{,}5]$. O *Pareto-ótimo* é formado com $g(x) = 1$, e o melhor *Pareto Front* com $g(x) = 1{,}25$.

A Figura 22 mostra a disposição das soluções no contexto da função ZDT4.

A última função de teste é a ZDT6, e inclui dificuldade causada pela não uniformidade do espaço de busca:

$$f_1(x_1) = 1 - \exp(-4x_1)\, sin^6(6\pi x_1), \tag{32}$$

$$g(x_2, \dots, x_m) = 1 + 9.((\sum_{i=2}^{m} x_i)/(m-1))^{0.25}, \tag{33}$$

$$h(f_1, g) = 1 - (f_1/g)^2, \tag{34}$$

onde $m = 10, x_i \in [0, 1]$. O *Pareto Front* está formado com $g(x) = 1$ e apresenta comportamento côncavo como mostra na Figura 23.

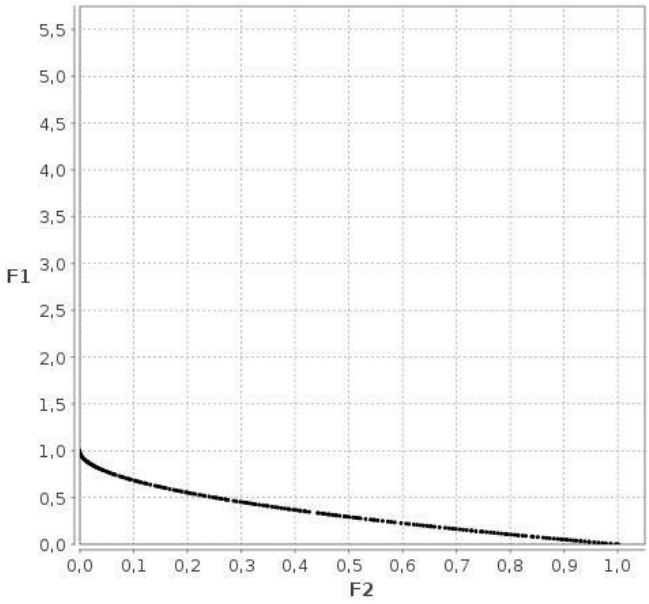

Figura 22. Representação gráfica da função ZDT4.

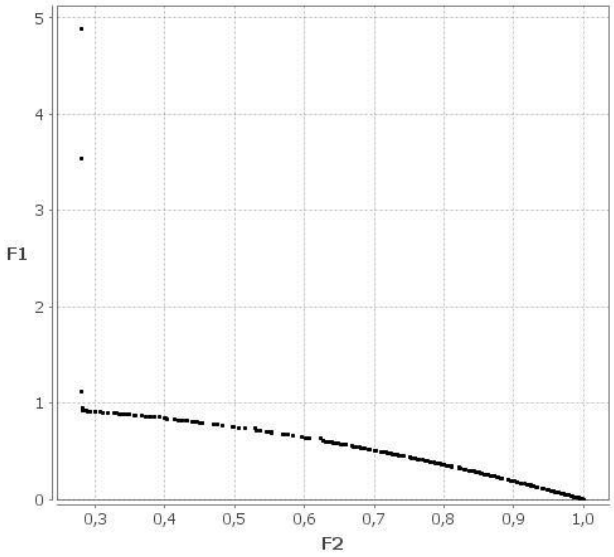

Figura 23. Representação gráfica da função ZDT6.

6.1.2. Métricas de Cálculo de Desempenho

As métricas utilizadas nesta experimentação são *Hypervolume, Spacing, Coverage*, e *Maximum Spread* citadas na sessão 3.2. Cada uma das técnicas experimentadas teve o seu *Pareto Front* analisado através da aplicação de cada uma das métricas citadas acima, possibilitando melhora na investigação das características de cada arquivo externo.

6.1.3. Parâmetros de Simulação

Cada técnica apresenta seu próprio conjunto de parâmetros. No caso do MOPSO, a taxa de mutação é 0,5, o número de divisões para o *grid* adaptativo é 30 e o fator de inércia diminui linearmente de 0,4 a 0,0 [15]. No MOPSO-CDLS o fator de inércia diminui linearmente de 0,9 a 0,4 [16]. No m-DNPSO o valor de $m = 10$ e o fator de inércia é gerado aleatoriamente em cada iteração no intervalo [0,5; 1,0] [4]. O CSS-MOPSO adota como desvio padrão para a mutação Gaussiana o valor 0,01 e o fator de inércia diminui linearmente de 0,9 a 0,4 [14]. No MOPSO-CDR é utilizado uma taxa de mutação de 0,5 e o fator de inércia diminui linearmente de 0,4 a 0,0. O MOPSO-CDRS apresenta a mesma configuração do MOPSO-CDR, porém, pelo fato de haver a análise estatística quanto à melhora significativa, em relação às métricas *Spacing* e *Maximum Spreading*, do *pareto*, adota-se como limite de saturação 0,1%.

Em todos os casos foram utilizadas 20 partículas no enxame e limite de 200 soluções no arquivo externo. 200.000 cálculos de *fitness* foram executados em cada simulação. As constantes de aceleração cognitiva e social igual a 1,49445, quando aplicadas. Para validar os resultados, cada simulação foi executada 30 vezes e os seus resultados são apresentados em termos da média e desvio padrão.

As simulações foram executadas em uma máquina com processador Intel Dual-Core, 2GB de memória RAM, rodando o sistema operacional Windows XP

6.2. Resultados

6.2.1. Comparação MOPSO-CDRS com demais Técnicas

Para realizar a comparação entre o MOPSO-CDRS e as demais técnicas foi necessário realizar simulações para cada técnica envolvida no contexto de todas as funções de teste. As simulações geram como saída o resultado da aplicação das métricas no *Pareto Front* de cada técnica.

A Tabela 1 mostra o valor das métricas de cada técnica para a função ZDT1. Como se pode observar tanto a métrica *Hypervolume* como *Spacing*, no caso do MOPSO-CDRS, é superior a todas com relação à média e apresentou um baixo desvio padrão. Já em relação à *Maximum Spread*, o valor é menor que o obtido nas demais técnicas, o que não significa dizer que o resultado é pior, e sim que devido a convergência, as partículas dos extremos terminaram se aproximando reduzindo o valor do *spreading*. Quanto à análise do *Coverage*, fica perceptível a grande vantagem da técnica proposta em relação às quatro primeiras técnicas, já em relação ao MOPSO-CDR, o MOPSO-CDRS apresenta 88% de dominância e é dominado em 3% dos casos.

Tabela 1. Resultado da simulação para a função ZDT1 com 200.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO	0,36 (0,002)	0,0046 (5E-4)	1,425 (0,005)	1,0 (0,0)	0,0 (0,0)
m-DNOPSO	0,713 (0,053)	0,0457 (0.014)	1,54 (0,065)	1,0 (0,0)	0,0 (0,0)
MOPSO CDLS	0,39 (0,003)	0,0042 (6E-4)	1,44 (0,005)	1,0 (0,0)	0,0 (0,0)
CSS MOPSO	0,34 (0,002)	0,0023 (1E-4)	1,42 (0,002)	0,99 (0,003)	0,0 (0,0)
MOPSO CDR	0,33 (3E-5)	0,0033 (2E-4)	1,41 (0,0)	0,88 (0,0034)	0,03 (0,023)
MOPSO CDRS	0,31 (2E-5)	0,0027 (1E-4)	1,38 (0,0)		

A Tabela 2 mostra o valor das métricas relacionadas à função ZDT2. MOPSO-CDRS, para o caso de *Hypervolume* e *Maximun Spread* apresentou um comportamento similar ao MOPSO-CDR, porém em *Spacing* alcançou melhor resultado. O MOPSO-CDRS superou todas as outras técnicas em *Coverage*. Em

relação ao MOPSO-CDR apresentou dominância de 72%, sendo dominado em apenas 19,5% dos casos.

Tabela 2. Resultado da simulação para a função ZDT2 com 200.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO	0,69 (0,001)	0,006 (0,001)	1,396 (0,015)	1,0 (0,0)	0,0 (0,0)
m-DNOPSO	0,94 (0,06)	0,054 (0,017)	1,29 (0,037)	1,0 (0,0)	0,0 (0,0)
MOPSO CDLS	0,716 (0,003)	0,006 (0,001)	1,39 (0,004)	1,0 (0,0)	0,0 (0,0)
CSS MOPSO	0,674 (0,001)	0,0035 (7E-4)	1,41 (8E-4)	0,978 (0,021)	0,0 (0,0)
MOPSO CDR	0,66 (3E-5)	0,0033 (2E-4)	1,41 (0,0)	0,72 (0,036)	0,195 (0,023)
MOPSO CDRS	0,656 (3E-5)	0,0029 (2E-5)	1,41 (0,0)		

A Tabela 3 apresenta os valores referentes à função ZDT3. Para as métricas *Hypervolume* e *Spacing* o MOPSO-CDRS apresentou melhora em relação à média e conseguiu reduzir bastante o desvio padrão em relação a todas as outras técnicas. No caso do *Maximun Spread* houve similaridade. Em relação ao *Coverage*, o MOPSO-CDRS obteve bons resultados, sendo dominado em 22% pela MOPSO-CDR.

Tabela 3. Resultado da simulação para a função ZDT3 com 200.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO	0,950 (0,004)	0,005 (4E-4)	1,976 (0,008)	1,0 (0,0)	0,0 (0,0)
m-DNOPSO	1,296 (0,088)	0,045 (0,016)	2,068 (0,146)	1,0 (0,0)	0,0 (0,0)
MOPSO CDLS	1,006 (0,009)	0,006 (9E-4)	1,988 (0,015)	1,0 (0,0)	0,0 (0,0)
CSS MOPSO	0,953 (0,008)	0,003 (7E-4)	1,983 (0,006)	0,999 (8E-4)	0,0 (0,0)
MOPSO CDR	0,920 (1E-4)	0,0033 (2E-4)	1,967 (2E-5)	0,69 (0,0056)	0,22 (0,034)
MOPSO CDRS	0,94 (6E-5)	0,0025 (0,0)	1,95 (9E-5)		

A Tabela 4 mostra os valores para a função ZDT4. O MOPSO-CDRS obteve resultados similares em relação ao MOPSO-CDR no caso *Hypervolume* e *Spacing*, porém obteve resultado satisfatório em relação ao *Maximum Spread*

reduzindo média e o desvio padrão, devido à redução da distância entre as soluções extremas. O MOPSO-CDRS obteve vantagem absoluta em relação ao *Coverage*, dominando quase por completo todas as outras técnicas.

Tabela 4. Resultado da simulação para a função ZDT4 com 200.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO	0,631 (0,526)	0,006 (0,0014)	1,54 (0,18)	0,68 (0,210)	0,2 (0,18)
m-DNOPSO	2,157 (0,935)	0,04 (0,037)	1,94 (0,29)	1,0 (0,0)	0,0 (0,0)
MOPSO CDLS	4,82 (0,2174)	0,005 (9E-4)	2,7 (0,46)	1,0 (0,0)	0,0 (0,0)
CSS MOPSO	5,38 (0,008)	0,005 (0,0012)	2,8 (0,525)	0,999 (8E-4)	0,0 (0,0)
MOPSO CDR	0,57 (0,26)	0,0033 (3E-4)	1,52 (0,109)	0,9 (3E-4)	0,015 (2E-3)
MOPSO CDRS	0,56 (0,012)	0,0025 (2E-4)	1,38 (2E-4)		

A Tabela 5 apresenta os valores referentes à função ZDT6. O MOPSO-CDRS apresenta uma melhora significativa em média e desvio padrão em relação ao MOPSO-CDR em todas as métricas, no caso do *Maximum Spreading* houve redução da distância das soluções extremas, resultando na diminuição do valor. Porém, apenas em *Spacing* obteve melhora em relação ao MOPSO. O *Coverage* do MOPSO-CDRS, para esta função, apresenta dominância razoável em relação às técnicas MOPSO e CSS-MOPSO, 48% e 41,7% respectivamente. As técnicas restantes são dominadas pela MOPSO-CDRS com valores acima de 88%.

Tabela 5. Resultado da simulação para a função ZDT6 com 200.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO	1,261 (0,386)	0,129 (0,122)	3,180 (1,4)	0,48 (0,102)	0,32 (0,0034)
m-DNOPSO	1,279 (0,506)	0,126 (0,108)	3,203 (1,732)	1,0 (0,0)	0,0 (0,0)
MOPSO CDLS	1,717 (0,519)	0,186 (0,145)	4,632 (1,816)	0,89 (2E-3)	0,09 (2E-4)
CSS MOPSO	2,051 (0,697)	0,234 (0,153)	5,571 (2,046)	0,417 (0,004)	0,002 (2E-3)
MOPSO CDR	1,670 (0,3)	0,088 (0,056)	4,636 (1,053)	0,88 (0,0034)	0,03 (0,023)
MOPSO CDRS	1,345 (0,46)	0,078 (2E-3)	3,233 (0,034)		

6.2.2. Comparação entre MOPSO-CDRS e MOPSO-CDR com número de iterações reduzido

Também foram realizadas simulações reduzindo-se de 200.000 chamadas da função fitness para 100.000 chamadas, com o objetivo de avaliar o desempenho das técnicas MOPSO-CDR e MOPSO-CDRS na primeira metade de uma simulação considerada completa. As tabelas a seguir seguem o mesmo padrão das que foram mostradas anteriormente.

Devido ao alto grau de convergência, o *Pareto* gerado apresenta redução da distância entre as soluções extremas, acarretando na redução do *spreading* para todas as funções.

A Tabela 6 mostra que o comportamento do *pareto* front das duas técnicas é bastante parecido, porém o *pareto* da MOPSO-CDRS domina o da CDR em 13,5% e é dominado em 0,5% dos casos.

Tabela 6. Resultado da simulação para a função ZDT1 com 100.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO CDR	0,36 (0,002)	0,0035 (0,002)	1,43 (0,001)	0,135 (0,0034)	0,005 (0,023)
MOPSO CDRS	0,36 (0,002)	0,0029 (0,002)	1,41 (0,001)		

A Tabela 7, da mesma forma que a anterior, apresenta grande similaridade entre os *pareto*s das duas técnicas. O MOPSO-CDRS obteve um percentual de dominância de 14,5% e não foi dominado pelo MOPSO-CDR.

Tabela 7. Resultado da simulação para a função ZDT2 com 100.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO CDR	0,76 (0,001)	0,0043 (0,007)	1,43 (0,005)	0,145 (0,0032)	0,0 (0,0)
MOPSO CDRS	0,72 (0,0023)	0,0042 (0,02)	1,42 (0,02)		

A Tabela 8 apresenta a superioridade do MOPSO-CDRS, em todas as métricas, em relação ao MOPSO-CDR. O *Coverage* obtido pela técnica mostra que esta dominou a concorrente em 74,5% dos casos, sendo dominada em apenas 1%.

Tabela 8. Resultado da simulação para a função ZDT3 com 100.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO CDR	0,93 (0,008)	0,0036 (0,032)	1,97 (0,019)	0,745 (0,001)	0,01 (0,034)
MOPSO CDRS	0,912 (0,034)	0,0027 (1E-4)	1,96 (0,004)		

A Tabela 9 mostra que MOPSO-CDRS atinge ótimo desempenho quando compete com MOPSO-CDR no início da simulação, para a função ZDT4. A técnica proposta superou a concorrente em todas as métricas, dominando-a 100%.

Tabela 9. Resultado da simulação para a função ZDT4 com 100.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO CDR	4,88 (2,61)	0,74 (0,56)	16,49 (8,79)	1,0 (0,0)	0,0 (0,0)
MOPSO CDRS	2,23 (1,32)	0,08 (0,05)	11,44 (5,39)		

A Tabela 10 mostra que, exceto em *Spacing*, a MOPSO-CDRS apresentou melhor desempenho em relação à concorrente. Seu *pareto* dominou em 67% e foi dominado em apenas 12%.

Tabela 10. Resultado da simulação para a função ZDT6 com 100.000 chamadas.

Algoritmo	Hypervolume	Spacing	Max. Spread	Cover CDRS,*	Cover *, CDRS
MOPSO CDR	2,05 (0,2)	0,17 (0,071)	6,03 (1,34)	0,67 (0,56)	0,12 (0,78)
MOPSO CDRS	1,8 (0,16)	0,189 (0,0017)	5,89 (0,59)		

De acordo com os resultados apresentados para o número de chamadas reduzido, no caso de funções mais simples, o MOPSO-CDRS e o MOPSO-CDR

obtiveram convergência similar, porém, para o caso de funções mais complexas como a ZDT4 e ZDT6 ficou clara a superioridade do *Pareto* gerado pela técnica proposta. Nestas duas funções, as soluções encontradas pelo MOPSO-CDR foram dominadas em 100% e 67% respectivamente. Além do *Coverage*, as demais métricas obtiveram uma melhora significativa em relação à obtida pelo MOPSO-CDR.

6.2.3. Comparação entre MOPSO-CDRS e MOPSO-CDR quanto ao tempo de execução

Além da análise em relação à eficácia, foi feito um comparativo entre o MOPSO-CDRS e o MOPSO-CDR quanto ao tempo gasto para a execução destes algoritmos para cada função objetivo. De acordo com a Tabela 11, que mostra a comparação com 100.000 chamadas, percebe-se um aumento considerável no tempo de execução do MOPSO-CDRS. Este aumento pode ser associado ao grande processamento realizado pelo *Gerenciador de arquivo externo* e pelo *Executor de decisões*. Estas duas entidades são responsáveis por muitas consultas ao arquivo externo, tornando a técnica proposta mais custosa que o MOPSO-CDR.

Tabela 11. Tempo de execução, em segundos, dos algoritmos MOPSO-CDRS e MOPSO-CDR para cada função objetivo usando-se 100.000 chamadas.

Algoritmo	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MOPSO CDR	37,681	35,566	42,683	2,372	4,381
MOPSO CDRS	112,250	107,887	117,475	30,969	33,053

O MOPSO-CDR também foi executado se utilizando 200.000 chamadas para verificar se, mesmo com um número de chamadas maior, o seu tempo de execução se mantém inferior ao do MOPSO-CDRS. A Tabela 12 mostra que mesmo com o aumento do número de chamadas, o MOPSO-CDRS apresenta um

tempo de execução superior, chegando-se a conclusão que é mais eficaz e eficiente a utilização do MOPSO-CDR com 200.000 chamadas do que o MOPSO-CDRS com 100.000 chamadas.

Tabela 12. Tempo de execução, em segundos, dos algoritmos MOPSO-CDRS usando 100.000 chamadas e MOPSO-CDR usando 200.000 chamadas para cada função objetivo.

Algoritmo	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MOPSO CDR	54,356	76,89	75,34	5,98	9,76
MOPSO CDRS	112,250	107,887	117,475	30,969	33,053

Capítulo 7

Conclusões e Trabalhos Futuros

Este trabalho apresenta uma nova técnica aplicada em problemas multi-objetivos (MOP), com o objetivo de melhorar o desempenho e comportamento das soluções presentes no *Pareto Front*, e contribuir com avanços na área de Otimização Multi-Objetivo.

Neste capítulo, serão feitas algumas considerações sobre os resultados dos experimentos realizados em relação à nova técnica proposta neste trabalho, como também sobre as contribuições aos estudos realizados na área de MOO e por fim os possíveis trabalhos futuros para extensão deste trabalho.

7.1. Contribuições

Fica claro que com o aparecimento de problemas cada vez mais complexos, é necessário fazer modificações nas técnicas de otimização a fim de se obter um melhor desempenho na busca de soluções adequadas. Este trabalho contribui com os estudos que vêm sendo realizados recentemente na busca por novas modificações e variações na heurística de otimização por enxame de partículas (PSO) aplicado a problemas com múltiplos objetivos.

Este trabalho estende o algoritmo MOPSO-CDR, propondo uma abordagem que agrega um analisador de arquivo externo, alterando a proposta do algoritmo de acordo com a análise realizada. Esta alternativa se torna uma alternativa eficiente apresentando um melhor desempenho em comparação a algumas técnicas presentes na literatura, citadas na sessão 4.2, assim contribuindo com avanços nas pesquisas de técnicas de otimização.

Tais contribuições foram provadas pelos experimentos realizados que demonstram os estudos comparativos entre a nova técnica MOPSO-CDRS e as técnicas existentes por meio de simulações envolvendo funções de teste, bastante utilizadas para avaliação de desempenho na literatura.

A técnica proposta, utilizando o número total de simulações adotado, apresenta uma sensível melhora de desempenho em relação à sua principal concorrente, o MOPSO-CDR; já em relação às técnicas restantes, a melhora é bastante superior.

Com a redução do número de iterações, utilizando-se metade do total, foi constatada a superioridade do MOPSO-CDRS em relação ao MOPSO-CDR. Observou-se que o MOPSO-CDRS forma um *Pareto Front* que apresenta um percentual de dominância significativo em relação ao seu concorrente, ou seja, no início da simulação o seu *Pareto* converge mais rapidamente.

7.2. Conclusão

Este trabalho apresenta uma nova técnica de otimização aplicada a problemas de busca e otimização com múltiplos objetivos. Esta técnica foi inspirada na estrutura algorítmica do MOPSO-CDR. De acordo com os resultados dos experimentos, mostrados no Capítulo 6, o MOPSO-CDRS obteve melhores soluções que os seus concorrentes, gerando um *Pareto Front* mais distribuído e compacto.

A criação do *Gerenciador do arquivo externo*, cuja função é analisar o comportamento das soluções, e do permutador de comportamento baseado em desempenho, que muda a proposta do algoritmo dinamicamente baseado no *Spacing* ou *Maximum Spreading*, contribuiu para a melhora da convergência do enxame e conseqüentemente no melhor desempenho do *Pareto*.

Os resultados apresentados no Capítulo 6 mostram que a técnica proposta, com tempo de iterações total, apresentou convergência superior a todas as outras citadas. Além disso, simulando-se com número de iterações reduzido à metade, para funções de teste mais complexas, o MOPSO-CDRS obteve maior grau de convergência, gerando um *Pareto Front* com percentual de dominância superior ao do MOPSO-CDR.

Por fim, pode-se concluir que novos mecanismos, o *Gerenciador* de arquivo externo e o permutador de comportamento baseado em desempenho, foram propostos com a introdução da técnica MOPSO-CDRS. Os estudos iniciais

validados pelos experimentos realizados neste trabalho demonstram que a técnica MOPSO-CDRS é uma alternativa eficaz para a resolução de problemas de busca e otimização por enxame de partículas com múltiplos objetivos, além de apresentar convergência satisfatória com um número menor de iterações. O ponto negativo da técnica proposta é o seu tempo de execução; embora o resultado gerado seja melhor que o do MOPSO-CDR, a mesma demora o dobro ou mais do tempo para finalizar a simulação.

7.3. Trabalhos Futuros

Como trabalho futuro, pode-se fazer uma análise em relação ao *limiar de saturação*. O MOPSO-CDRS utiliza o limiar fixo igual a 0,1% e quando este limiar é ultrapassado, o comportamento do algoritmo muda. Um estudo poderia ser feito na determinação de um valor ótimo para o limiar ou implementar uma proposta de limiar variável.

Outro possível trabalho futuro seria estudar a influência de outras métricas no desempenho do arquivo externo, de modo a introduzi-las para análise no *Gerenciador*.

O MOPSO-CDRS, quando está no modo *Especiação* utiliza o MOPSO-CDR como técnica em um dos agrupamentos gerados, um estudo futuro seria substituir o MOPSO-CDR por outra técnica, bem fundamentada na literatura, para verificar sua interferência no desempenho do *Pareto*.

De modo a tentar sanar o problema encontrado, o tempo de execução do algoritmo, pode-se refatorar o código desenvolvido ou aperfeiçoar os mecanismos de análise do arquivo externo e tomada de decisões.

Além das possibilidades acima, pode-se incorporar ainda mais objetivos e analisar o desempenho do MOPSO-CDRS em relação às demais técnicas.

Bibliografia

[1] NEDJA, N.; DOS SANTOS COELHO, L.; DE MACEDO DE MOURELLE, L. **Studies in Computational Intelligence**. Rio de Janeiro: Springer, v. 261, 2009. Multi-Objective Swarm Intelligent Systems.

[2] KENNEDY, J.; EBERHART, R. Particle Swarm Optimization, 4, 1995. 1942-1948.

[3] REYES-SERRA, M.; COELLO, C. A. C. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art, 2002. 34-54. Electrical Engineering Department, Computer Science Section.

[4] SANTANA, R. A.; PONTES, M. R.; BASTOS-FILHO, C. J. A. A Multiple Objective Particle Swarm Optimization Approach Using Crowding Distance And Roulette Wheel. Department of Computing Systems, UPE.

[5] REYNOLDS, C. W. Flocks, Herds And Schools: A Distributed Behavioral, 1987. 25-34. Computer Graphics.

[6] MILLONAS, M. M. **Swarms, Phase Transitions and Collective Intelligence**. 3. ed. Chicago: Artifial Life, 1994.

[7] EBERHART, R.; KENNEDY, J. A New Optimizer Using Particle Swarm Theory, 1995. 39-43.

[8] CLERC, M.; KENNEDY, J. The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space, v. 6, p. 58-73, 2002. IEEE Transactions on Evolutionary Computation.

[9] EBERHART., Y. S. A. R. C. A Modified Particle Swarm Optimiser, Maio 1998. International Conference of Evolutionary Computation.

[10] ZITZLER, E. **Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications**. ETH Zurich. Switzerland. 1999.

[11] ZITZLER, E.; THIELE, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and The Strength *Pareto* Approach, 1999. 257-271. IEEE Transactions on Evolutionary Computation.

[12] ZITZLER, E.; THIELE, L. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study, 1998. 292-271. Conference on Parallel Problem Solving from Nature.

[13] COELLO, C. A. C.; VAN VELDHUIZEN, D. A.; LAMONT, G. B. Evolutionary

Algorithms for Solving Multi-Objetive Problems, New York, Maio 2002. Kluwer Academic Publishers.

[14] CHIU, S. Y. et al. Cross-Searching Strategy for Multi-Objective Particle Swarm Optimization. **Congress on Evolutionary Computation**. 3135-3141. IEEE.

[15] COELLO, C. A. C.; PULIDO, G. T.; LECHUGA, M. S. Handling Multiple Objectives With Particle Swarm Opimization, Junho 2004. IEEE Transactions Evolutionary Computation.

[16] TSOU, C.; CHANG, S.; LAI, P. Using Crowding Distance to Improve Multi-Objective PSO with Local Search. In: TIWARI, F. T. S. C. A. M. K. **Swarm Intelligence, Focus on Ant and Particle Swarm Optimization**. Vienna, Austria: I-Tech Education and Publishing, 2007. p. 77-86.

[17] RAQUEL, C. R.; NAVAL JR., P. C. An Effective Use of Crowding Distance in Multi-Objective Particle Swarm Optimization. **Proceedings of the 2005 Conference on Genetic and Evolutionary Computation**, New York, USA, 2005. 257-264. GECCO'05.

[18] HU, X.; EBERHART, R.; SHI, Y. Particle Swarm With Extended Memory For Multi-Objective Opitmization. **In Swarm Intellingence Symposium**, 2003. 193-197. SIS'03.

[19] COELLO, C. A. C.; LAMONT, G. B.; VAN VELDHUIZEN, D. A. Evolutionary Algorithms for Solving Multi-Objective Problems. **Genetic and Evolutionary Computation**, New York, 2007.

[20] DEB, K. Multi-objective genetic algorithms: Problem Difficulties and Construction of Test Problems, 1999. 205-230.

[21] COELLO, C. A. C.; LAMONT, G. B. An Introduction to Multi-Objective Evolutionary Algorithms And Their Application, 1999. 14-26.

[22] KONAK, A.; COIT, D. W.; SMITH, A. E. Multi-Objective Optimization Using Genetic Algorithms: A Tutorial. **Reliability Engineering and System Safety**, p. 13-29, 2006. Information Sciences and Technology, Penn State Berks, USA.