OTIMIZADOR MULTIOBJETIVO BASEADO EM INTELIGÊNCIA DE ENXAMES UTILIZANDO FATOR DE DIVERSIDADE

Trabalho de Conclusão de Curso

Engenharia de Computação

Aluno: Dennis Rodrigo da Cunha Silva

Orientador: Prof. Dr. Carmelo José Albanez Bastos Filho

UNIVERSIDADE
DE PERNAMBUCO

Dennis Rodrigo da Cunha Silva

Otimizador multiobjetivo baseado em inteligência de enxames utilizando fator de diversidade

Monografia apresentada para obtenção do Grau de Bacharel em Engenharia de Computação pela Universidade de Pernambuco

Orientador:
Prof. Dr. Carmelo José Albanez Bastos Filho

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO
ESCOLA POLITÉCNICA DE PERNAMBUCO
UNIVERSIDADE DE PERNAMBUCO

Recife - PE, Brasil

Novembro de 2012

ESCOLA POLITÉCNICA
DE PERNAMBUCO

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 7 de 12 de 2012, às 14:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente DENNIS RODRIGO DA CUNHA SILVA, orientado pelo professor Carmelo José Albanez Bastos Filho, sob título OTIMIZADOR MULTI-OBJETIVO BASEADO EM INTELIGÊNCIA DE ENXAMES UTILIZANDO FATOR DE DIVERSIDADE, a banca composta pelos professores:

Cícero Garrozi

Carmelo José Albanez Bastos Filho

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 10,0 (DEZ)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

Cícero Garrozi

CÍCERO GARROZI

Carmelo José A. Bastos Filho

CARMELO JOSÉ ALBANEZ BASTOS FILHO

Resumo

O recente sucesso da área de inteligência de enxames, iniciado pela proposta de um algoritmo de otimização baseado em enxame de partículas, o PSO, fez com que abordagens semelhantes fossem propostas para outras aplicações. Uma delas é a otimização multiobjetivo e diversos algoritmos foram propostos na última década com o intuito de encontrar um conjunto de soluções que melhor resolva um problema com muitos objetivos conflitantes. Tais técnicas têm sido propostas visando a melhoria da qualidade dos resultados finais, velocidade de convergência e diversidade de soluções.

Neste trabalho de conclusão de curso é apresentado um novo otimizador multiobjetivo baseado em enxames, o MOPSO-DFR, visando a melhoria do espaçamento e espalhamento das partículas pelo espaço de busca. O mecanismo proposto utiliza um conceito chamado *Fator de Diversidade*, que mede a distribuição de cada partícula no espaço de objetivos, pertencente ao arquivo externo. O *Fator de Diversidade* é usado para definir os líderes cognitivos e sociais das partículas, bem como é aplicado como critério de ordenação do arquivo externo. Esse critério de ordenação é utilizado para efetuar a remoção das piores partículas do arquivo externo, uma vez que o número máximo é excedido.

Com a utilização do *Fator de Diversidade* e utilizando a nova proposta de seleção de líderes, foi possível encontrar um bom conjunto de soluções que resolve um problema específico a ser tratado. Para analisar a qualidade do algoritmo foi utilizado um conjunto funções de teste amplamente utilizado pela comunidade científica, intitulado DTLZ. Foram avaliados os valores das métricas *Coverage*, *Maximum Spread*, *Spacing*, e *Hypervolume*. Por fim, foram realizadas comparações com diversas técnicas existentes na literatura. O MOPSO-DFR apresentou resultados de *Maximum Spread* melhor, em alguns casos, que diversas técnicas bem conhecidas da literatura como MOPSO-CDR, SMPSO e NSGA-II. Os resultados obtidos de *Coverage* para a proposta apresentada foram superiores em relação aos resultados encontrados pelas outras técnicas, na maioria das as funções de teste utilizadas. Além da proposta ter conseguido gerar soluções que dominam as soluções geradas pelas outras técnicas, a Frente de Pareto gerada apresenta bons resultados *Maximum Spread*, *Spacing* e *Hypervolume*.

Abstract

Due to the recent evergrowing interesting in swarm intelligence, mainly due to the Particle Swarm Optimization (PSO), many applications on optimization have been proposed. One of these is known as multiobjective optimization and several algorithms have been proposed within the last decade in order to find a set of solutions that better describes the trade-off between several conflicting objectives. Several techniques have been proposed aiming to outperform the quality of final results, speed of convergence and diversity of the solutions.

In this monography, we present a new swarm-based multiobjective optimizer, called MOPSO-DFR, in order to improve spreading and spacing of the particles through the search space. The mechanism proposed uses a concept called *Diversity Factor*, that measures the distribution of each particle in the objectives space, belonging to the external archive. The *Diversity Factor* is used to define the cognitive and social leaders of all particles. Besides, it is used as a sorting criterion to rank the external archive as well. This sorting criterium is used in order to remove the worsts external archive particles, once the maximum number is reached.

Using the *Diversity Factor* and the new approach to select the leaders, it is possible to find a good set of solutions that solves a specific problem. In order the analyze the quality of MOPSO-DFR, we was used a set of function tests widely used by the scientific community, called DTLZ. The metrics Maximum Spread, Spacing, Hypervolume and Coverage have been evaluated. A comparison between MOPSO-DFR and several techniques in the literature is performed. In some cases, MOPSO-DFR has shown better Maximum Spread results when compared with several well-known techniques as MOPSO-CDR, SMPSO and NSGA-II. The Coverage results achieved by the proposed technique were better when compared to the results found by the other techniques, for most test functions. Beside this, the Pareto Fronts achieved by the proposal present good values for the metrics Maximum Spread, Spacing and Hypervolume results.

Agradecimentos

Primeiramente à minha mãe, Maria de Fátima da Cunha, por ter me ensinado como a educação é importante não só no caráter profissional como também no ético. Se não fosse por seus conselhos e palavras confortantes eu com certeza não teria chegado até aqui. Também à minha irmã, Débora Cunha, por ter sido bastante altruísta e compreensiva principalmente durante o último ano de graduação.

À minha companheira, namorada e amiga, Janaina Câmara, por ter me aturado e ajudado diretamente não só com este trabalho como também durante toda graduação. Por ter sido uma das grandes responsáveis pelo meu amadurecimento pessoal e profissional e ter me ensinado a dar valor às coisas importantes da vida.

Ao Professor Carmelo J. A. Bastos-Filho, por ter me orientado com excelência não só durante este trabalho de conclusão de curso, como também durante todos os anos de Iniciação Científica. E aos demais pesquisadores que compõem o grupo de estudo de Inteligência de Enxames e Inteligência Computacional da Universidade de Pernambuco.

Aos meus companheiros de classe, os quais contribuíram diretamente com o que me tornei hoje. Não só pelas contribuições acadêmicas e profissionais, como também por todas as horas que passamos jogando até tarde, saindo para comemorar as conquistas de cada um (e também do grupo como um todo) e pela companhia nos momentos difíceis que passamos. Todo esse companheirismo foi fundamental.

Aos meus grandes amigos da ETF, por renovar minhas energias todas as sexta-feiras. Eles foram fundamentais, fazendo com que esses 5 anos passassem depressa e que eu pudesse enfrentar de cabeça erguida qualquer desafio imposto. Também aos amigos da Coeus, pelas constantes discussões sobre ciência e pelo trabalho duro que damos todos os dias.

"(...) Todos os dias quando acordo
Não tenho mais o tempo que passou
Mas tenho muito tempo
Temos todo tempo do mundo."
Tempo perdido - Legião Urbana

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

ABC – *Artificial Bee Colony*
ACO – *Ant Colony Optimization*
AE – Arquivo Externo
APSO – *Adaptive Particle Swarm Optimization*
CEGA – *Co-Evolutionary Genetic Algorithm*
DF – *Diversity Factor*
EA – *Evolutionary Algorithms*
GA – *Genetic Algorithms*
MDFA – *Multi-Directional Fitness Assignment*
MOP – *Multi-Objective Problems*
MOPSO – *Multi-objective Particle Swarm Optimization*
MOPSO-CDR – *Multi-Objective Particle Swarm Optimization using Crowding Distance and Roulette Wheel*
MOPSO-DFR – *Multi-Objective Particle Swarm Optimization using Diversity Factor and Roulette Wheel*
MOPSO-LS – *Multi-Objective Particle Swarm Optimization using Local Search*
NSGA-II – *Non-dominated Sorting Genetic Algorithm II*
PSO – *Particle Swarm Optimization*
SMPSO – *Speed Constrained Particle Swarm Optimization*
SPEA-2 – *Strength Pareto Evolutionary Algorithm 2*

1 Introdução

"*Se eu vi mais longe,*
foi por estar de pé no ombro de gigantes."
– Isaac Newton.

Neste trabalho de conclusão de curso, foi desenvolvida uma nova meta-heurística multiobjetiva baseada em enxame de partículas. A escolha de desenvolver uma proposta baseada em enxames foi definida para se aproveitar do comportamento intrínseco desse tipo de algoritmos se comportar de forma adequada em espaços de busca contínuos.

Este capítulo apresenta a introdução desta monografia, e está organizado em 3 Seções. Na Seção 1.1, é apresentada a motivação para a realização deste trabalho bem como o problema abordado pelo mesmo. Em seguida, na Seção 1.2 são apresentados os objetivos gerais e específicos e a hipótese de como o problema poderá ser solucionado. Por fim, na Seção 1.3, é descrita a estrutura do restante da monografia.

1.1 Motivação e Caracterização do Problema

Encontrar uma solução para um problema com dois ou mais objetivos conflitantes sempre foi uma tarefa difícil de ser alcançada. Por esse motivo, diversos algoritmos foram propostos com o intuito de resolver esses problemas de maneira rápida e precisa. Otimização de problemas multiobjetivos (MOP, *Multi-Objective Problems*) é um tema que vem sido bastante discutido na comunidade científica com o passar dos anos e isso se deve principalmente ao fato do mundo real possuir diversos problemas com vários objetivos conflitantes [2].

Dentre os algoritmos de busca e otimização, uma das técnicas mais famosas é o Algoritmo de Otimização por Enxame de Partículas (PSO, *Particle Swarm Optimiza-*

tion) que foi proposto por Kennedy e Eberhart em 1995 [3]. Esta técnica foi inspirada no comportamento social de bandos de pássaros em busca de comida. Esse algoritmo, que apesar de simples e rápido, se mostrou capaz de encontrar resultados bons para problemas de alta dimensionalidade. Entretanto, a versão padrão do PSO otimiza problemas com apenas um objetivo. Além do PSO, diversas técnicas de busca e otimização bioinspiradas foram propostas nas últimas décadas: Algoritmos Genéticos (GA, *Genetic Algorithms*) [4], Otimização por Enxame de Formigas (ACO, *Ant Colony Optimization*) [5], Colônia de Abelhas Artificiais (ABC, *Artificial Bee Colony*) [6], dentre outras.

Dado que problemas reais geralmente apresentam mais de um único objetivo e estes são conflitantes, vários algoritmos evolucionários (EA, *Evolutionary Algorithms*) foram propostos, como por exemplo o SPEA-2 (*Strength Pareto Evolutionary Algorithm 2*) [7], o NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) [8], o CEGA (*Co-evolutionary Genetic Algorithm*) [9] e o MDFA (*Multi-Directional Fitness Assignment*) [9]. Algoritmos baseados em enxames são geralmente mais adequados a solucionar problemas cujo espaço de busca é contínuo graças à forma em que as partículas do enxame interagem. Devido a essa capacidade intrínseca, alguns algoritmos baseados em enxame também foram propostos para resolver problemas multiobjetivos com múltiplas variáveis contínuas como o MOPSO (*Multi-objective PSO*) [10], SMPSO (*Speed Constrained PSO*) [11] o MOPSO-LS (*Multi-Objective PSO using Local Search*) [12] e o MOPSO-CDR (*Multi-Objective PSO using Crowding Distance and Roulette Wheel*) [13].

O algoritmo APSO (*Adaptive Particle Swarm Optimization*), proposto por Zhan *et al* [14], contém um mecanismo de adaptação baseado no estado evolucionário do enxame. Na proposta original do APSO, o fator evolucionário é calculado para definir qual estado evolucionário o enxame está associado. Este fator considera a distância mínima entre uma partícula e as demais, a distância máxima entre uma partículas e as demais e a distância do líder em relação as demais partículas. A partir de um estado evolucionário selecionado, alguns parâmetros do algoritmo são atualizados para garantir seu comportamento adaptativo. Além disso, o APSO contém alguns mecanismos introduzidos com o intuito de melhor a proposta do PSO original como: uma estratégia de aprendizado elitista, que permite que o líder do enxame se comporte de forma mais satisfatória, e uma proposta de adaptação do parâmetro do fator inercial da partícula [14].

O MOPSO-CDR é uma técnica baseada no algoritmo proposto por Tsou *et al* [12]. A técnica utiliza o mecanismo de *crowding distance* e *roulette wheel* com o intuito de selecionar o líder do enxame e prevenir um número excessivo de soluções no arquivo externo. Da mesma forma que o MOPSO, o MOPSO-CDR utiliza um arquivo externo para armazenar as melhores soluções encontradas e com o passar das iterações, as melhores soluções são adicionadas a esse arquivo externo. Com isso, é calculado o valor de *crowding distance* de uma solução, que representa uma estimativa da densidade das soluções em comparação com as outras soluções [8]. Além disso, o MOPSO-CDR introduziu um novo procedimento para atualizar a melhor posição cognitiva da partícula que considera o *crowding distance*.

A técnica desenvolvida neste trabalho de conclusão de curso utiliza uma abordagem similar ao fator evolucionário proposto por Zhan *et al* [14]. É utilizada uma variação desse fator evolucionário, intitulado fator de diversidade, para selecionar os líderes sociais e cognitivos do enxame, assim como para realizar a manutenção do conjunto de partículas pertencentes à Frente de Pareto.

1.2 Hipóteses e Objetivos

Existem diversas propostas multiobjetivas de algoritmos evolucionários que são capazes de convergir e encontrar um bom conjunto de soluções para um dado problema. O campo de inteligência de enxames ainda não possui uma técnica capaz de convergir e encontrar um conjunto de soluções suficientemente bom. Entretanto, as técnicas da inteligência de enxames possuem características intrínsecas adequadas à otimização multiobjetiva.

Devido a esta hipótese, este trabalho de conclusão de curso tem como objetivo desenvolver uma nova meta heurística capaz de resolver MOPs de forma precisa e rápida, visando obter melhorar a convergência de técnicas baseadas em enxame. Espera-se encontrar como resultados uma Frente de Pareto próxima à real. A técnica proposta por este trabalho consiste em utilizar uma abordagem similar ao algoritmo MOPSO-CDR, utilizando o Fator de Diversidade como mecanismo de seleção e ordenação do conjunto de soluções. O algoritmo desenvolvido foi testado para um conjunto de funções *benchmark* propostos por Deb *et al* [1] e houve uma comparação com outros algoritmos da literatura, para comprovar a qualidade da técnica proposta.

1.3 Organização do Documento

Este trabalho está organizado em 6 capítulos. No Capítulo 2 serão abordados aspectos relativos à inteligência de enxames e a descrição do PSO. Em seguida, no Capítulo 3, será realizada uma explanação de técnicas multiobjetivas. No Capítulo 4 será apresentada a contribuição deste trabalho de conclusão de curso: o MOPSO-DFR (*Multi-Objective Particle Swarm Optimization using Diversity Factor and Roulette Wheel*). Em seguinda, no Capítulo 5, os experimentos e resultados serão apresentados. Por fim, no Capítulo 6, serão discutidas as principais conclusões desse trabalho, como também propostas para trabalhos futuros.

2 *Inteligência de Enxames e Otimização por Enxame Partículas.*

"Os problemas significativos que enfrentamos não podem ser resolvidos no mesmo nível de pensamento em que estávamos quando os criamos."
– Albert Einstein.

Neste capítulo são apresentados os conceitos básicos da técnica de otimização por enxame de partículas (PSO, *Particle Swarm Optimization*), e diversos avanços propostos para esta técnica. Além disso, será apresentada a técnica otimização adaptiva por exame de partículas (APSO, *Adaptive Particle Swarm Optimization*), que consiste em uma técnica adaptativa baseada em enxame de partículas.

Otimização baseada em enxames (*Swarm Intelligence*) é um ramo da Inteligência Computacional que utiliza o conceito de partículas em comunicação para otimizar problemas. Dentro desse conceito, uma partícula é uma estrutura simples com pouca memória. Essas partículas se comunicam entre si trocando o pouco de informações que elas podem armazenar. É a partir dessa comunicação que emerge a "Inteligência" desse tipo de técnica. Na Subseção 2.1 será apresentado o mais famoso algoritmo de área de otimização baseada em enxames, o PSO.

2.1 Otimização por enxame de partículas

O algoritmo de otimização baseado em enxame de partículas foi proposto em 1995 pelos pesquisadores Kennedy e Eberhart [3]. A ideia do algoritmo deu-se a partir de observações de bandos de pássaros e de uma análise do seu comportamento. Foi realizado um estudo sobre as regras que guiavam o bando, como também sobre a forma em que voavam em sincronia. Além disso, notou-se que os bandos mudavam de direção de forma repentina sem perder as características individuais de cada pássaro.

O PSO foi proposto após ser feita uma simulação de um modelo social simplificado baseado nas observações e estudos realizados sobre os bandos de pássaros que procuravam por comida. Desta forma, definiu-se que um enxame seria composto por um conjunto de entidades, conhecidas como partículas, que interagem localmente entre si, resultando em um comportamento global emergente, buscando a solução de problemas de forma distribuída.

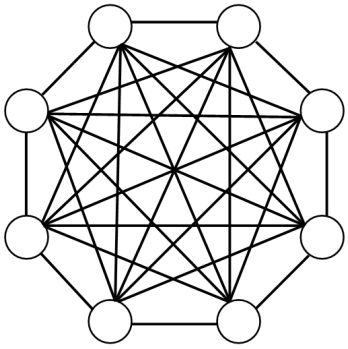
Atualmente, existem diversas implementações de algoritmos baseados em PSO. As partículas do enxame voam pelo espaço de busca sendo guiadas por uma combinação de entre a velocidade no instante anterior, a melhor posição encontrado pela partícula em questão e a melhor posição encontrado pela partícula em sua vizinhança. A vizinhança de uma partícula significa o conjunto de partículas pertencentes ao enxame com as quais a partícula em questão pode se comunicar. Em alguns casos, a vizinhança em que uma partícula está inserida pode compreender todo o enxame.

A vizinhança de uma partícula é definida pela topologia de comunicação entre as partículas. A topologia de comunicação normalmente não está relacionado com a distância entre as partículas e sim com o índice de cada uma delas. Entre as topologias propostas na literatura [15] [16] podem ser citadas: topologia global, onde cada partícula se comunica com todas as outras (ver Figura 2.1a); topologia em anel, onde cada partícula apenas se comunica com seus vizinhos adjacentes (ver Figura 2.1b); topologia de Von Neumann, onde as partículas são arranjadas na forma de um cristal (ver Figura 2.1c).

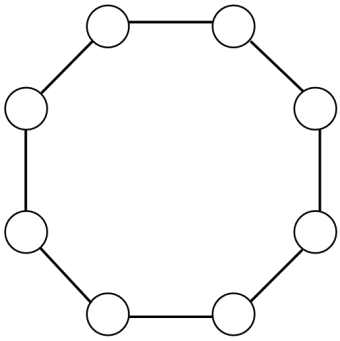
A velocidade de cada partícula é obtida através de uma soma vetorial de três componentes, que são:

- Vetor inércia: representa o movimento da partícula no instante anterior;
- Vetor cognitivo: representa a porção cognitiva da partícula, ou seja, o que ela foi capaz de aprender no processo de otimização até então;
- Vetor social: representa a componente social da partícula. Consiste na melhor posição encontrada até então, pela vizinhança em que a partícula está inserida.

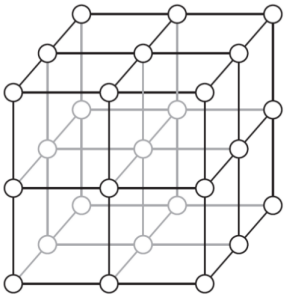
Para facilitar o entendimento da atualização da velocidade da partícula, um esquema gráfico pode ser visto na Figura 2.2. O esquema representa a posição da partícula antes e depois da atualização, a velocidade resultante, além da influência dos vetores inércia, cognitivo e social.



(a) Topologia estrela.



(b) Topologia em anel.



(c) Topologia Von Neumann.

Figura 2.1: Ilustração das topologias de comunicação mais usadas.

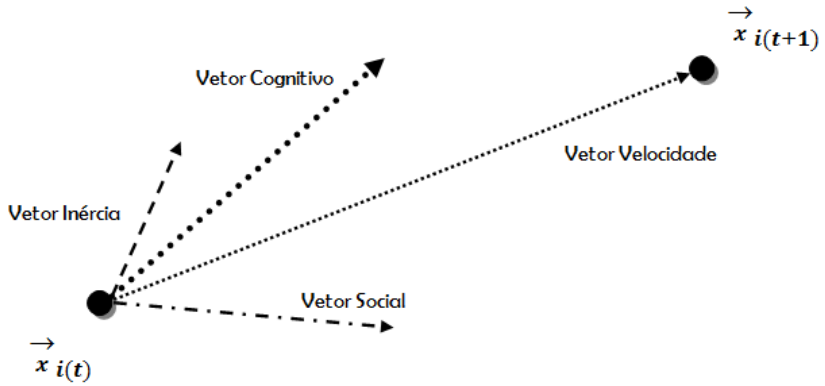


Figura 2.2: Influência dos vetores inércia, cognitivo e social na atualização da posição da partícula.

No PSO, cada partícula representa um ponto no domínio da função *fitness*. Cada partícula i é composta por quatro vetores: sua posição atual no espaço de busca D-dimensional, $\vec{x}_i = (x_{i1}, x_{i2}, ..., x_{id})$, sua melhor posição encontrada até então, ou seja, sua memória cognitiva, $\vec{p}_i = (p_{i1}, p_{i2}, ..., p_{id})$, a melhor posição encontrados pelos seus vizinhos até então $\vec{n}_i = (n_{i1}, n_{i2}, ..., n_{id})$ e sua velocidade $\vec{v}_i = (v_{i1}, v_{i2}, ..., v_{id})$.

As partículas do enxame percorrem o espaço de busca, à procura da melhor posição possível. A posição e velocidade de cada partícula são atualizadas de acordo

com as equações (2.1) e (2.2):

$$\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}_i) + c_2 \cdot r_2 \cdot (\vec{n}_{best} - \vec{x}_i), \tag{2.1}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1), \tag{2.2}$$

em que c_1 e c_2 são os coeficientes de aceleração cognitivo e social, respectivamente, r_1 e r_2 são dois números aleatórios, no intervalo $[0,1]$, gerados a cada iteração do algoritmo, para cada partícula, em cada dimensão, x_i representa a posição atual da partícula i, p_{best} consiste na melhor posição encontrada pela partícula i até então e por fim, n_{best} representa a melhor posição encontrada na vizinhança de i.

O PSO apresenta como passo inicial a inicialização aleatória das posições e velocidades de toda partícula do enxame. Além disso, é necessário definir uma condição de parada do processo de otimização. Essa condição de parada deve ser definida pela quantidade de iterações que o algoritmo deve executar, por um limiar de aceitação, ou seja, caso o enxame chegou a um ponto onde o valor de *fitness* não melhore significativamente ou se o algoritmo já atingiu um valor de *fitness* previamente definido. Além da condição de parada, outros parâmetros devem ser definidos no algoritmo como: os valores das constantes c_1 e c_2, a quantidade de partículas no enxame, os valores máximos que a velocidade da partícula pode atingir e os limites do espaço de busca. Na abordagem original, c_1 e c_2 são constantes com valor igual a 2, a quantidade de partículas do enxame varia entre 20 e 50 e os valores da velocidade e dos limites do espaço de busca variam de acordo com o problema [3].

No PSO, após sua inicialização, o algoritmo entra em um laço que é executado até que um dos critérios de parada definido seja atingido. Em cada iteração t, cada partícula i deve atualizar sua velocidade e posição de acordo com as Equações (2.1) e (2.2), respectivamente, calcular o valor de *fitness* para a nova posição encontrada, além de atualizar o $\vec{p}_{best}$ e $\vec{n}_{best}$.

O pseudocódigo do PSO pode ser visto em Algoritmo 1.

Em algumas abordagens, após o cálculo da velocidade ser realizado, é necessário checar se essa velocidade está entre os limites mínimos e máximo pré-definidos. Isto é feito para evitar o estado de explosão, ou seja, que a velocidade atinja valores superiores ao tamanho do espaço de busca. Algumas abordagens utilizam como valor máximo 20% do espaço de busca, outras 50% [17]. Além disso, é fundamental para a dinâmica do enxame ter um controle da variação máxima de posição entre duas

Algoritmo 1: Pseudocódigo do PSO.

```
Inicialize as partículas do enxame com posição e velocidade aleatórias;
enquanto critério de parada não for alcançado faça
    para cada partícula faça
        Localizar melhor partícula na vizinhaça;
        Atualizar velocidade utilizando Equação (2.1);
        Atualizar posição utilizando Equação (2.2);
        Avaliar posição levando em conta função objetiva do problema;
        se posição atual melhor que memória cognitiva então
            Atualizar memória cognitiva;
Retorne a melhor memória cognitiva do enxame.
```

iterações consecutivas.

Após a nova posição ser calculada, é necessário verificar se a partícula está dentro do espaço de busca definido. Um estratégia simples é deixar a partícula entrar no espaço infactível sem calcular o valor da função objetivo na próxima iteração. Espera-se que a partícula retorne ao espaço definido, uma vez que seu valor de *fitness* permanece inalterado e existe uma atração da partícula pelo $\vec{n}_{best}$. Outra alternativa consiste em atualizar o valor da posição da partícula para o limite definido.

2.2 Inércia e Fator de Constrição

A implementação original, do PSO foi capaz de resolver problemas de forma eficaz e rápida, entretanto foi observado um comportamento negativo na dinâmica do algoritmo, notou-se que frequentemente o enxame entrava em um estado de “explosão” de velocidades. A partícula em dados instantes de tempo adquiria uma velocidade muito alta, fazendo com que oscilasse entre os extremos do espaço de busca.

A primeira e mais famosa proposta de mudança do algoritmo original foi proposta por Kennedy e Eberhart, onde foi adicionado um fator de aceleração na equação de atualização da velocidade [18]. Este fator foi introduzido para evitar que o enxame entrasse no estado de “explosão”. Entretanto, foi observado que definir o valor deste fator não seria uma tarefa fácil de ser realizada e definir valores fixos para esse fator impactavam de forma negativa na dinâmica do algoritmo.

Devido a isso, foi proposto uma decaimento linear do coeficiente de aceleração, mostrado na Equação (2.3),

$$\omega = \omega_{max} - [(\omega_{max} - \omega_{min})\frac{g_{atual}}{g_{final}}], \tag{2.3}$$

onde g_{atual} é a iteração atual e g_{final} é o número total de iterações a ser realizado, e ω_{max} e ω_{min} são valores máximo e mínimo do coeficiente de aceleração, respectivamente.

A equação de velocidade com coeficiente de aceleração incorporado pode ser visto na Equação (2.4):

$$\vec{v}_i(t+1) = \omega \cdot \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}_i) + c_2 \cdot r_2 \cdot (\vec{n}_{best} - \vec{x}_i). \tag{2.4}$$

A estratégia de decrescimento do coeficiente de aceleração permite balancear a capacidade do enxame de mudar de busca em amplitude para busca em profundidade com o passar das iterações. Essa estratégia garante que nas iterações inicias o enxame realize uma busca em amplitude, permitindo explorar de forma abrangente o espaço de busca, já que o algoritmo ainda não encontrou uma região próxima ao ponto mínimo da função objetivo. Além disso, provê a capacidade da busca em profundidade maior nas iterações finais, onde o algoritmo provavelmente já encontrou uma área próxima ao ponto mínimo da função objetiva e está apenas refinando as soluções encontradas.

Além do coeficiente de aceleração, outra contribuição importante para o PSO foi feita por Clerc, que realizou uma análise nos parâmetros do PSO com a finalidade de investigar as propriedades de convergência e estabilidade [19] [20]. Como resultado da análise, foi determinado um "fator de constrição". Clerc provou que quando este coeficiente é incorporado à equação de velocidade do PSO, o enxame converge para uma região de um ótimo local após um número de iterações. Além disso, o ajuste da velocidade das partículas do enxame ocorre de forma estável, evitando o estado de explosão. Com o uso do fator de constrição não é mais necessário impor limites de velocidade para as partículas do enxame.

O fator de constrição é calculado através das Equações (2.5) e (2.6). A nova equação de atualização de velocidade, com o fator de constrição incorporado pode ser visto na Equação (2.7).

$$\varphi = c_1 + c_2, \tag{2.5}$$

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \tag{2.6}$$

$$\vec{v}_i(t+1) = \chi \cdot \{\vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}_i) + c_2 \cdot r_2 \cdot (\vec{n}_{best} - \vec{x}_i)\}. \tag{2.7}$$

Clerc [20] demonstrou que só existe garantia de convergência quando $\varphi > 4$. Quando $\varphi <= 4$, as partículas movem de forma lenta executando movimento em espiral e não há garantia de convergência para uma área de um mínimo local. Para valores de $c_1 = c_2 = 2,05$, tem-se $\varphi = 4,1$ e $\chi \approx 0,7298$. É importante notar que nas análises realizadas por Clerc o peso da inércia, ω, foi desconsiderado.

2.3 Otimização adaptativa por enxame de partículas

O algoritmo PSO adaptativo (APSO, *Adaptive PSO*) foi proposto por Zhan *et al* [14] com o intuito de super as seguintes deficiências do PSO original: tempo significativamente alto para atingir a convergência e capacidade limitada de escapar de mínimos locais. O APSO apresenta um esquema de adaptação sistemático de parâmetros como também uma estratégia de aprendizado elitista. Os passos que são executados a cada geração do APSO consistem em:

- Calcular o fator evolucionário;
- Classificar o enxame em um dos estados evolucionários;
- Adaptar os parâmetros de aceleração evando em conta o estado em que o enxame se encontra;
- Adaptar o parâmetro de inércia.

Cada tópico mostrado acima é explicado nas seções subsequentes.

2.3.1 Estimativa do estado evolucionário

O estado evolucionário é determinado pelo valor do fator evolucionário. O modo como os parâmetros são atualizados depende do estado evolucionário que o enxame se enquadra no momento. O cálculo da estimativa do estado evolucionário é realizado em dois passos.

O primeiro passo consiste no cálculo da distância média de cada partícula para todas as outras, de acordo com a Equação (2.8).

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2}, \tag{2.8}$$

em que N é a quantidade de partículas do enxame e D é a quantidade de dimensões. Após o cálculo da distância média de todas as partículas, é calculado o fator evolucionário do enxame, dado pela Equação (2.9):

$$f_{evol} = \frac{d_g - d_{min}}{d_{max} - d_{min}}, \tag{2.9}$$

em que f_{evol} é o fator evolucionário, d_g é a distância média entre a melhor partícula (*i.e.* o líder) e as demais partículas do enxame, d_{min} é a distância mínima entre uma partícula e as demais partículas do enxame e d_{max} é a distância máxima entre uma partícula e as demais partículas do enxame. É necessário o uso de ao menos 3 partículas no enxame devido a normalização do fator evolucionário, para evitar que $d_{min} = d_{max}$.

Quanto mais próximo de "0"for f_{evol}, significa que as partículas do enxame estão mais próximas à melhor partícula, entretanto quanto mais próximo de "1" significa que as partículas estão mais distantes da melhor partícula.

2.3.2 Classificação do enxame em um dos estados evolucionários

Na proposta original do APSO [14], a classificação do enxame em um dos estados evolucionários é feito através de um processo de classificação *fuzzy*. Existem quatro possíveis estados evolucionários: convergência, busca em profundidade, busca em amplitude e escape. Para classificar o fator evolucionário funções de pertinência são utilizadas.

Estado de convergência: ocorre quando o valor de f_{evol} é próximo de "0", em outras palavras quando a distância do líder do enxame para as demais partículas do enxame é mínima. No caso desse estado, o algoritmo está refininando os valores encontrados em uma região ótima. Calcula-se o valor *fuzzy* do estado utilizando a Equação (2.10)

$$E_{converge(f_{evol})} = \begin{cases} 1, & \text{se } 0,0 \leq f_{evol} \leq 0,1 \ , \\ -5f_{evol}, & \text{se } 0,1 < f_{evol} \leq 0,3 \ , \\ 0, & \text{se } 0,3 < f_{evol} \leq 1,0 \ . \end{cases} \tag{2.10}$$

Estado de busca em profundidade: ocorre quando o valor de f_{evol} é pequeno, ou seja, neste caso a distância entre a melhor partícula e as demais é pequena. O algoritmo busca soluções satisfatórias em uma região limitada. Para calcular o valor *fuzzy* para esse estado, é utilizada a Equação (2.11).

$$E_{profundidade(f_{evol})} = \begin{cases} 0, & \text{se } 0,0 \leq f_{evol} \leq 0,2 \ , \\ 5f_{evol} - 2, & \text{se } 0,2 < f_{evol} \leq 0,3 \ , \\ 1, & \text{se } 0,3 < f_{evol} \leq 0,4 \ , \\ -5f_{evol} + 3, & \text{se } 0,4 < f_{evol} \leq 0,6 \ , \\ 0, & \text{se } 0,6 < f_{evol} \leq 1,0 \ . \end{cases} \tag{2.11}$$

Estado de busca em amplitude: ocorre quando o valor de f_{evol} é médio ou alto, ou seja, neste caso a distância entre a melhor partícula e as demais é média ou alta. O algoritmo está em busca de uma região ótima, onde estão inseridos os mínimos. Para realizar o cálculo do valor *fuzzy* desse estado, é utilizada a Equação (2.12).

$$E_{amplitude(f_{evol})} = \begin{cases} 0, & \text{se } 0,0 \leq f_{evol} \leq 0,4 \ , \\ 5f_{evol} - 2, & \text{se } 0,4 < f_{evol} \leq 0,6 \ , \\ 1, & \text{se } 0,6 < f_{evol} \leq 0,7 \ , \\ -10f_{evol} + 8, & \text{se } 0,7 < f_{evol} \leq 0,8 \ , \\ 0, & \text{se } 0,8 < f_{evol} \leq 1,0 \ . \end{cases} \tag{2.12}$$

Estado de escape: ocorre quando o valor de f_{evol} é próximo de "$1,0$", ou seja, neste caso a distância entre a melhor partícula e as demais é grande. Neste caso, o líder saiu de uma região onde se encontrava um mínimo local para uma região com um ótimo mais promissor. Para realizar o cálculo do valor *fuzzy* do estado de escape, é utilizada a Equação (2.13).

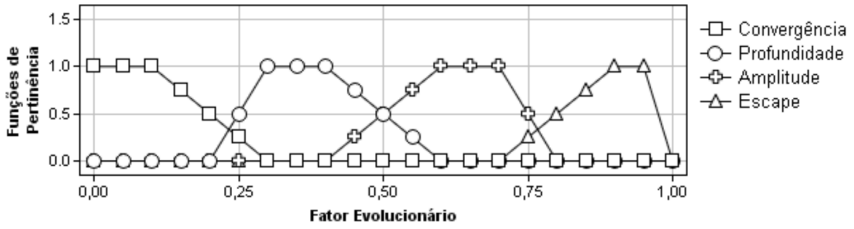


Figura 2.3: Funções de pertinência *fuzzy* utilizadas no APSO.

$$E_{escape(f_{evol})} = \begin{cases} 0, & \text{se } 0,0 \leq f_{evol} \leq 0,7\ , \\ 5f_{evol} - 3,5, & \text{se } 0,7 < f_{evol} \leq 0,9\ , \\ 1, & \text{se } 0,9 < f_{evol} \leq 1,0. \end{cases} \tag{2.13}$$

A Figura 2.3 exibe um esquema gráfico das funções de pertinências utilizadas na proposta original do APSO [14].

Os valores para cada estado são reiniciados com valor 0 em toda iteração. Uma vez calculado os estados, aquele que obtiver o maior valor *fuzzy* é selecionado como o estado evolucionário atual do enxame.

2.3.3 Adaptação dos parâmetros de aceleração

Na proposta original do APSO, os coeficientes de aceleração c_1 e c_2 são inicializados com valor igual a $2,0$ e, durante sua execução, são atualizados adaptativamente em função do estado evolucionário conforme a Tabela 2.1.

Tabela 2.1: Estratégias de controle dos coeficientes de aceleração.

Estado evolucionário	c_1	c_2
Busca em amplitude	incrementar	decrementar
Busca em profundidade	incrementar levemente	decrementar levemente
Convergência	incrementar levemente	incrementar levemente
Escape	decrementar	incrementar

A importância do estado de busca em amplitude é descobrir o maior número possível de ótimos no espaço de busca. Essa estratégia permite que as partículas explorarem individualmente e obtenham suas próprias posições ótimas, ao invés de se associar a uma suposta melhor partícula que pode estar em um mínimo local.

No estado de busca em profundidade as partículas fazem uso da informação local

e se "voam" ao redor de possíveis ótimos locais, levando em conta $\vec{n}_{best}$. Espera-se que os ajustes de c_1 e c_2 o estado de profundidade deve conduzir de modo apropriado o algoritmo do estado de busca em amplitude para o estado de convergência.

No estado de convergência o enxame possivelmente encontrou uma região que possui um ótimo global. A influência social deve aumentar com o intuito de atrair as partículas para a melhor partícula da vizinhança, já a influência cognitiva deve diminuir. Entretanto, foi observado durante os testes do APSO, que realizar esse procedimento leva o algoritmo a convergir prematuramente, uma vez que os parâmetros saturam rapidamente para os limites mínimos e máximos. Como alternativa, decidiu-se incrementar levemente os dois coeficientes de aceleração.

No estado de escape acontece quando a melhor partícula deixa um ótimo local para outro melhor. Após a descoberta dessa nova região, as outras partículas devem seguir a melhor partícula do enxame. Sendo assim, é dada uma maior importância ao novo líder social em detrimento da sua experiência cognitiva de cada partícula. Isso faz com que a partícula seja guiada para uma região promissora.

O passo de incremento ou decremento dos coeficientes de aceleração é chamado de taxa de aceleração (δ). Na proposta original do APSO, o valor da taxa de aceleração (δ) é obtido de uma distribuição uniforme entre $0,05$ e $0,10$. Nas estratégias onde é necessário um incremento ou decremento leve, usa-se $0,5 \cdot \delta$.

Após a etapa de atualização dos coeficientes de aceleração é necessário normalizá-los de acordo com a Equação (2.14).

$$c_i = \frac{c_i(c_{min} + c_{max})}{c_1 + c_2}. \tag{2.14}$$

2.3.4 Estratégia de aprendizado elitista

Com a incorporação dos procedimentos mostrados nas seções anteriores, é esperado que o PSO atinja a convergência de forma mais rápida. Entretanto, quando o algoritmo está no estado de convergência, a partícula $\vec{n}_{best}$ não possui um líder para ser seguido. Desta forma o mecanismo de aprendizado padrão não ajuda o $\vec{n}_{best}$ a escapar de ótimo atual, no caso deste ser local.

Deste modo, uma estratégia de aprendizado elitista foi desenvolvida na proposta original do APSO com o intuito de ajudar o $\vec{n}_{best}$ a se movimentar para uma região

possivelmente melhor. Essa estratégia consiste em escolher uma dimensão da partícula $\vec{n}_{best}$ de forma aleatória, representada por $\vec{n}_{bestd}$, em que d representa a dimensão escolhida. Então, uma perturbação Gaussiana é gerada de acordo com a Equação (2.15).

$$n_{bestd}(t+1) = n_{bestd}(t) + (X^d_{max} - X^d_{min})G(\mu, \sigma^2), \tag{2.15}$$

em que (X^d_{min}, X^d_{max}) representam os limites do espaço de busca, $G(\mu, \sigma^2)$ é um número aleatório obtido de um distribuição gaussiana com média (μ) zero e o desvio padrão (σ) variando no tempo, e pode ser calculado através da Equação (2.16). Além disso, a nova posição só é atualizada se for melhor do que a atual posição da melhor partícula do enxame. Caso contrário, esta solução será usada para substituir a pior partícula do enxame.

$$\sigma = \sigma_{max} - (\sigma_{max} - \sigma_{min})(\frac{g_{atual}}{g_{final}}), \tag{2.16}$$

em que σ_{max} e σ_{min} são os limites máximo e mínimo para a taxa de aprendizado elitista, g_{atual} é a iteração atual e g_{final} é o total de iterações a serem realizadas. Foi demonstrado que $\sigma_{max} = 1,0$ e $\sigma_{min} = 0,1$ geram boas soluções. Uma alta taxa de aprendizado elitista provê ao líder a possibilidade de deixar um ótimo local, uma baixa taxa de aprendizado elitista permite que o líder refine a melhor solução encontrada até então.

2.3.5 Adaptação do parâmetro de inércia

No APSO, o fator de inércia ω é obtido em função do fator evolucionário utilizando a Equação (2.17). Lembrando que o ajuste do fator de inércia ω é usado para balancear a capacidade de busca em amplitude e busca em profundidade.

$$\omega(f) = \frac{1}{1 + 1,5e^{2,6f}} \in [0,4; 0,9], \forall f \in [0,1]. \tag{2.17}$$

2.3.6 Pseudo código do algoritmo APSO

O pseudo código do algoritmo APSO é apresentado no Algoritmo 2. Na linha 8, a classificação do enxame em um estado evolucionário é feita utilizando o estado que

apresenta o maior valor de pertinência.

Algoritmo 2: Pseudocódigo do APSO.

```
Iniciar as partículas do enxame com posição e velocidade aleatórias;
enquanto critério de parada não for alcançado faça
    para cada partícula faça
        Calcular distância média utilizando a Equação (2.8);
    Calcular fator evolucionário utilizando a Equação (2.9);
    Calcular funções de pertinências utilizando Equações (2.10),
    (2.12), (2.11) e (2.13);
    Classificar enxame em um dos estados evolucionários;
    Adaptar os coeficientes de acordo com a Tabela 2.1;
    Normalizar os coeficientes de aceleração utilizando a Equação (2.14);
    Atualizar o parâmetro de inércia utilizando a Equação (2.17);
    se classificado em estado de convergência então
        Gerar uma nova posição utilizando as Equações (2.15) e (2.16);
        se nova posição é melhor que memória cognitiva então
            Atualizar posição e memória cognitiva da melhor partícula;
        Atualizar posição e memória da pior partícula;
    para cada partícula faça
        Localizar melhor partícula na vizinhaça;
        Atualizar velocidade utilizando Equação (2.1);
        Atualizar posição utilizando Equação (2.2);
        Avaliar posição levando em conta função objetiva do problema;
        se posição atual melhor que memória cognitiva então
            Atualizar memória cognitiva;
Retorne a melhor memória cognitiva do enxame.
```

3 *Otimização multiobjetivo*

"*Deve-se mais aprender em profundidade,*
do que em largura."
– Quintiliano.

Neste capítulo serão abordados os aspectos teóricos referentes à otimização multiobjetivo. Na Seção 3.1 são definidos problemas multiobjetivo. Em seguida, são apresentadas na Seção 3.2 técnicas baseadas no conceito de enxames de partículas e que são utilizadas na resolução de problemas multiobjetivo. Por fim, serão apresentadas na seção 3.3 algumas métricas que servem para avaliar a qualidade das soluções encontradas por técnicas de otimização multiobjetivo.

3.1 Conceitos de otimização multiobjetivo

Um problema multiobjetivo consiste em um problema no qual existem dois ou mais objetivos conflitantes a serem resolvidos, em outras palavras, melhorar um objetivo implica negativamente na qualidade dos demais. Como exemplo prático para esse tipo de problema, pode-se levar em consideração que uma pessoa no processo de compra de um novo celular, deseja comprar o dispositivo que tenha o menor custo com a melhor configuração possível. Percebe-se que existe um conflito entre melhorar a configuração do celular e diminuir seu custo, portanto esse problema se caracteriza como multiobjetivo. Aplicações reais na área de engenharia, geralmente possuem diversos problemas de objetivos conflitantes, e para isso é interessante a aplicação de técnicas multibjetivas.

A otimização de um problema multiobjetivo consiste na busca do melhor conjunto de parâmetros de entrada para o problema específico, tal que esses parâmetros encontrem um conjunto de solucões bem distribuído e diverso. Formalmente, uma otimização multiobjetivo é definada como:

Otimização Multiobjetivo Um problema de otimização multiobjetivo é definido como a minimização ou maximização de $F(\vec{x}) = \{f_1(\vec{x}), ..., f_k(\vec{x})\}$, sujeita às restrições $g_i(\vec{x}) \leq 0,\ i = 1, ..., m$ e $h_j(\vec{x}) = 0,\ j = 1, ..., p$ e $\vec{x} \in D$. Uma solução do problema minimiza (ou maximiza) os componentes do vetor $F(\vec{x})$, onde $\vec{x}$ é uma variável representada por um vetor n-dimensional, $\vec{x} = \{x_1, ..., x_n\}$, de algum universo *D*. Note que $g_i(\vec{x}) \leq 0$ e $h_j(\vec{x}) = 0$ representam restrições que precisam ser satisfeitas durante a minimização (ou maximização) de $F(\vec{x})$ e *D* contém todos os possíveis $\vec{x}$ que podem ser usados para satisfazer uma avaliação de $F(\vec{x})$ [21].

Em problemas de otimização multiobjetivo existe um conjunto de soluções equivalentes consideradas não-dominadas na perspectiva de múltiplas funções objetivo. Essas soluções são chamadas de não-dominadas ou soluções pertencentes à Frente de Pareto [21]. Cada uma dessas soluções apresenta resultado melhor que as demais soluções em pelo menos um objetivo e o seu conjunto representa as solucões ótimas que melhor soluciona um problema. Para facilitar o entendimento do conceito de dominância, segue a definição dominância para um problema de minimização:

Dominância [21] Um vetor $\vec{u} = (u_1, ..., u_k)$ **domina** um vetor $\vec{v} = (v_1, ..., v_k)$ (representado por $\vec{u} \succ \vec{v}$) se e somente se:

$$\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i. \tag{3.1}$$

Deste modo, um processo de otimização multiobjetivo consiste em determinar o conjunto de soluções da Frente de Pareto a partir do conjunto de todos os vetores de variáveis de decisão. Deste modo, espera-se que o conjunto de soluções encontrado se assemelhe com o Pareto-ótimo esperado. Além disso, um algoritmo ideal deve encontrar soluções que sejam bem distribuídas ao longo da Frente de Pareto.

3.2 Algoritmos baseados em enxame para otimização multiobjetiva

Devido a capacidade de solucionar problemas de único objetivo, os algoritmos baseados em enxame de partículas foram adaptados para serem aptos a resolver problemas com múltiplos objetivos conflitantes [10] [22].

Para fazer a adaptação para solucionar problemas multiobjetivo é necessário alcançar três pontos: maximizar o número de solucções pertencentes à Frente de Pa-

reto, aproximar a Frente de Pareto encontrada da verdadeira Frente de Pareto do problema e maximizar o espalhamento das soluções encontradas, visando-se obter um conjunto de soluções uniforme e bem distribuído ao longo da Frente de Pareto.

Para que o PSO possa ser adaptado para solucionar problemas multiobjetivo surgem as seguintes questões a serem analisadas:

- Como selecionar partículas para serem usadas como líderes, de modo a dar preferência às soluções não dominadas?
- Como armazenar as soluções não dominadas encontradas durante o processo de busca, de modo a reportar as soluções não dominadas de todo o passado?
- Como manter diversidade no enxame de modo a evitar convergência para uma única solução?

Em resposta a essas questões, foi proposto por Coello Coello e colaboradores o algoritmo de Otimização Multiobjetiva por Enxame de Partículas (MOPSO, *Multi-Objective Particle Swarm Optimization*) [10] apresentado adiante na Subseção 3.2.1.

Neste trabalho foi desenvolvida uma técnica de otimização multiobjetivo baseada no PSO. Nas seguintes subseções serão apresentados a primeira versão do algoritmo baseado em PSO para solucionar problemas multiobjetivo, como também outra técnica proposta mais recentemente, o MOPSO-CDR.

3.2.1 *Multi-Objective Particle Swarm Optimization* (MOPSO)

O MOPSO foi a primeira meta heurística baseada em PSO capaz de solucionar problemas multiobjetivo [10]. Para desenvolver esse algoritmo, foi necessário adicionar à estrutura original do PSO um arquivo externo capaz de armazenar as melhores soluções encontradas durante o processo de otimização. Em contraste com o PSO original, o líder ($\vec{n}_{best}$) de cada partícula será escolhido dentre as soluções pertencentes ao arquivo externo. O objetivo dessa mudança é acelerar a convergência do algoritmo.

Nesta abordagem, para fazer a escolha do líder de cada partícula, o arquivo externo é dividido em hipercubos. Cada hipercubo recebe um valor de *fitness* que depende do número de partículas associadas ao hipercubo. Uma seleção por roleta é usada para escolher o líder de cada partícula do enxame. Uma vez que um hipercubo seja escolhido, uma das partículas dentro do hipercubo é aleatoriamente escolhida

para ser o líder. Dessa forma, esse método força as partículas do enxame a explorar as regiões menos densas, gerando diversidade entre as soluções resultantes.

Porém, a diversidade gerada por esse método não é suficiente para que o algoritmo seja capaz de escapar de mínimos locais. Com o intuito de resolver esse problema, um operador de turbulência foi criado, para provocar uma pertubação Gaussiana no vetor velocidade de algumas partículas do enxame escolhidas aleatoriamente a cada iteração. A principal desvantagem desse algoritmo consiste na dependência do número de hipercubos utilizados com o problema. Desta forma, se este parâmetro não for escolhido de forma satisfatória, a eficácia do algoritmo pode ser comprometida.

3.2.2 *Multiple Objective Particle Swarm Optimization Approach using Crowding Distance and Roulette Wheel* (MOPSO-CDR)

O MOPSO-CDR consiste em uma técnica de otimização multiobjetivo proposta por Santana *et al* [22]. Da mesma forma que o MOPSO original, o MOPSO-CDR utiliza um arquivo externo para armazenar as melhores soluções encontradas durante o processo de otimização. Esse arquivo é iniciado vazio e todas as soluções encontradas não-dominadas que são encontradas ao longo do processo são adicionadas a esse arquivo externo. Uma nova solução encontrada pelo enxame só será incorporada ao arquivo externo caso seja melhor que as demais ou não-dominadas, seguindo o conceito de dominância.

As soluções presentes no arquivo externo possuem um valor de *fitness* associado a cada partícula, que é calculado levando em conta o seu *crowding distance*. O valor de *crowding distance* de uma soluções consiste em uma estimativa da densidade das soluções em relação às demais [8]. A Figura 3.1 exibe o cálculo do *crowding distance* do ponto *i*, que consiste em estimar o tamanho do maior cubóide em torno de *i* sem que qualquer outro ponto seja incluído.

Para determinar o líder social de cada partícula, leva-se em consideração o *fitness* usando o método da roleta (do inglês, *Roulette Wheel*). Da mesma forma que o MOPSO, a seleção do líder tende a escolher regiões menos densas.

A atualização da memória cognitiva da partícula ($\vec{P}_{best}(t)$) também leva em conta o *crowding distance*. A atualização é realizada toda vez que a nova posição da partícula dominar o valor atual do $\vec{P}_{best}(t)$. No caso das soluções serem não-dominadas, então é procurado no arquivo externo a solução mais próxima da nova posição e a solução

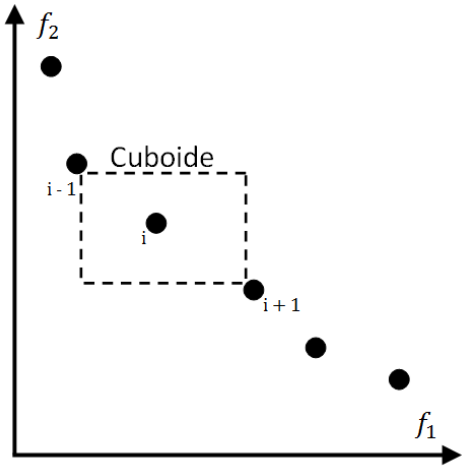


Figura 3.1: Cálculo do *crowding distance*.

mais próxima do $\vec{P}_{best}(t)$ atual. Para calcular a distância entre as partículas é utilizado o cálculo Distância Euclidiana entre dois vetores. Caso a solução mais próxima da nova posição tenha um melhor valor de *crowding distance*, em outras palavras, esteja em uma região menos densa do que a solução mais próxima do $\vec{P}_{best}(t)$ atual, então a nova posição se torna o líder cognitivo ($\vec{P}_{best}(t)$). Caso contrário, o líder cognitivo é mantido.

Além disso, o *crowding distance* é utilizado para realizar a poda do arquivo externo. Quando o tamanho máximo do arquivo externo é excedido, ou seja, quando a quantidade de soluções pertencentes à Frente de Pareto for maior que um valor pré-determinado, as soluções com os menores valores de *crowding distance* serão descartadas do arquivo externo. A utilização do *crowding distance* tanto para realizar a manuntenção do arquivo externo, quanto para selecionar os líderes cognitivos e social, visa produzir uma Frente de Pareto uniformemente distribuída, para que a maioria das regiões do espaço de objetivos sejam bem representadas pelas soluções encontradas pelo algoritmo.

Com o intuito de aumentar a diversidade no processo de busca e otimização, o MOPSO-CDR também utiliza o operador de turbulência proposto por Coello Coello [10], no MOPSO original. Nas primeiras iterações do algoritmo, todas as partículas do enxame recebem uma perturbação Gaussiana, de acordo com o passar do processo de otimização a influência deste operador se torna cada vez menos expressiva. O Algoritmo 3 representa o pseudocódigo do MOPSO-CDR.

Algoritmo 3: Pseudocódigo do MOPSO-CDR.

```
Inicialize o enxame;
Determine os líderes iniciais do arquivo externo;
Qualifique os líderes considerando crowding distance;
enquanto critério de parada não for alcançado faça
    para cada partícula faça
        Aplique o operador de turbulência utilizado no MOPSO;
        Selecione líder usando crowding distance e seleção por roleta;
        Atualize velocidade e posição;
        Avalie a qualidade da partícula;
        Atualize P⃗best(t) usando torneio binário;
    Atualize líderes do arquivo externo;
    Qualifique os líderes por crowding distance;
Retorne o Arquivo Externo.
```

3.3 Métricas para avaliação de soluções

Como o resultado de uma otimizaçao multiobjetivo consiste em um conjunto com as melhores soluções, fica inviável realizar uma comparação qualitativa considerando apenas o conjunto com as soluções obtidas. Com o intuito de analisar a qualidade das soluções obtidas através do processo de otimização multiobjetivo, foram propostas diversas técnicas para avaliar sua eficácia, tendo como foco: o grau de convergência e a diversidade das soluções.

O grau de convergência consiste em exibir o quanto as soluções encontradas pelo processo de otimização se aproximam da Frente de Pareto ideal verdadeiro. Deste modo, quanto maior for a eficácia da técnica em encontrar as soluções para resolver o problema em questão, maior será o grau de convergência encontrado.

Uma boa diversidade significa que o conjunto de soluções encontrado está bem distribuído ao longo da Frente de Pareto. Este fator é considerado importante pois o conjunto irá conter uma boa variedade de possibilidades de escolha e o tomador de decisões será capaz de escolher uma solução que melhor se adeque às condições externas e à realidade.

Diversas métricas foram propostas para mensurar as duas características citadas anteriormente e desta forma avaliar a qualidade de técnicas multiobjetivas. As seguintes subseções detalham as seguintes métricas: *Coverage Set*, *Hypervolume*, *Spacing* e *Maximum Spread* [21].

3.3.1 *Coverage Set* (C)

A métrica *Coverage* é utilizada para avaliar o grau de convergência do algoritmo. Como o seu resultado é obtido de forma comparativa, dois paretos são necessários para calcular essa métrica. A Equação 3.2 mostra como é realizado o cálculo do *coverage set* para dois conjuntos Frente de Pareto A e B (leia-se *Coverage* de A em relação a B):

$$C(A,B) = \frac{|\{b \in B; \exists a \in A : a \succ b\}|}{|B|}, \tag{3.2}$$

em que |B| representa a quantidade de soluções na Frente de Pareto B.

Nota-se que o valor de $C(A,B)$ pertence ao intervalo [0,1]. Caso o valor de $C(A,B) = 1$ todas as soluções de B são dominadas pelas soluções de A, ou seja, a Frente de Pareto A tem um grau de convergência maior do que o B. Entretanto, caso $C(A,B) = 0$ nenhuma das soluções em B são dominadas por A. Observa-se também que $C(B,A)$ não necessariamente é igual à $1 - C(A,B)$.

3.3.2 *Hypervolume* (HV)

A fim de explicar este conceito considere um problema de otimização com dois objetivos. O *Hypervolume* é definido pela área no espaço de objetivos coberta pela Frente de Pareto ($\mathscr{P}_a^*$) (a área abaixo da curva).

Considere o retângulo delimitado pelo ponto $(f_1(\vec{x}); f_2(\vec{x}))$ que pertence à Frente de Pareto e a origem. Suponha que cada ponto na Frente de Pareto gera um retângulo no espaço de objetivos. O HV corresponde pela área formada pela união de todos os retângulos.

É também possível generalizar este conceito para problemas com n-objetivos, usando a seguinte equação:

$$HV = \left\{ \bigcup_i a_i \,|\, x_i \in \mathscr{P}_a^* \right\}, \tag{3.3}$$

onde x_i é um vetor não-dominado em ($\mathscr{P}_a^*$) e a_i é o *Hypervolume* determinado pelos componentes de x_i e a origem.

3.3.3 *Spacing* (S)

O *Spacing* foi uma métrica proposta por Schott [23] com o intuito de valorar o espalhamento (distribuição) de soluções não-dominadas ao longo da Frente de Pareto. Essa métrica consiste em medir a variância das distâncias entre a solução não-dominada adjacente e a distância média entre todas as soluções adjacentes, como mostra a Equação (3.4).

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}, \tag{3.4}$$

em que $d_i = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|)$, $i, j = 1, ..., n$, $\bar{d}$ é a distância média entre todas as soluções adjacentes e n é o número de soluções não-dominadas na Frente de Pareto.

Um valor igual de *Spacing* a zero significa que todas as soluções na Frente de Pareto são equidistantes.

3.3.4 *Maximum Spread* (MS)

Esta métrica foi proposta pro Zitzler *et al* [24] em 1999. Nessa métrica a distância euclidiana entre as duas soluções mais afastadas da Frente de Pareto é calculada através da Equação (3.5),

$$MS = \sqrt{\sum_{m=1}^{M}(max_{i=1}^{n} f_m^i - min_{i=1}^{n} f_m^i)^2}, \tag{3.5}$$

em que n é o número de soluções da Frente de Pareto e M é o número de objetivos de um determinado problema.

Quanto maior o valor do *Maximum Spread* for, maior será a qualidade da técnica utilizada, já que altos valores de MS significa as soluções cobrem uma área significativamente grande do espaço de objetivos, tendendo a crer, desta forma, que o conjunto de soluções tem uma alta diversidade. Entretanto, apenas valores altos de MS não significa necessariamente uma boa diversidade, apenas um possível indicativo.

4 A Nova Abordagem: MOPSO-DFR

"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original."
– Albert Einstein

Neste capítulo será apresentada a proposta deste trabalho, o MOPSO-DFR (do inglês, *Multi-Objective Particle Swarm Optimization using Diversity Factor and Roulette Wheel*), uma abordagem multiobjetiva adaptativa baseada nos algoritmos MOPSO-CDR [13] e APSO [14].

4.1 Alterações propostas

O algoritmo proposto por este trabalho tem como objetivo melhorar a convergência em uma técnica baseada em enxames multiobjetiva, visando encontrar uma Frente de Pareto mais espaçada e espalhada. Para tanto, algumas mudanças foram propostas no algoritmo MOPSO-CDR. O MOPSO-CDR foi escolhido como algoritmo base por se tratar de uma metaheurística capaz de encontrar bons resultados quando comparada às outras metaheurísticas baseadas em enxame, por utilizar as abordagens *Crowding Distance* e *Roulette Wheel* para selecionar os líderes cognitivos e sociais.

Com o objetivo de melhorar os resultados obtidos pelo MOPSO-CDR, algumas modificações foram introduzidas no algoritmo original. A primeira contribuição deste trabalho consiste em uma abordagem chamada Fator de Diversidade (DF, *Diversity Factor*), que consiste em analisar a distribuição das partículas ao longo da Frente de Pareto. O DF é inspirado no Fator Evolucionário, proposto originalmente por Zhang *et al* [14], na técnica APSO (*Adaptive Particle Swarm Optimization*), apresentada no Capítulo 2 deste trabalho.

Para realizar o cálculo do DF é necessário calcular a distância média de uma

partícula para as demais. Para auxiliar a explicação desta abordagem, considere a Frente de Pareto na Figura 4.1:

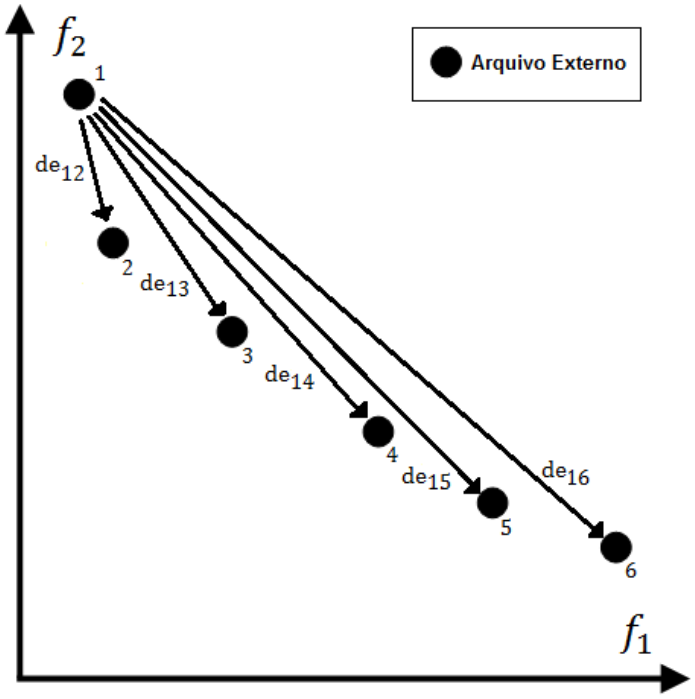


Figura 4.1: Ilustração de como é realizado o cálculo do DF.

A distância média da partícula i para as demais partículas da Frente de Pareto é dada pela Equação (4.1):

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} \sqrt{\sum_{k=1}^{M} (x_i^k - x_j^k)^2}, \tag{4.1}$$

em que N é a quantidade de partículas da Frente de Pareto e M é a quantidade de objetivos do problema em questão.

A partir da distância média calculada na Equação (4.1), é possível calcular o DF de cada partícula pertencente à Frente de Pareto. O cálculo do DF é dado pela Equação (4.2).

$$DF_i = \frac{d_i - d_{min}}{d_{max} - d_{min}}, \tag{4.2}$$

em que DF_i é o DF da partícula i, d_i é a distância média da partícula i para as demais partículas, d_{max} e d_{min} são as distâncias médias máxima e mínima, respectivamente, de uma partícula para as demais.

Acredita-se que com a utilização do DF, o conjunto de soluções encontrado terá melhores espaçamento e espalhamento uma vez que leva-se em a conta distância

média das partículas no espaço de objetivos. As partículas mais próximas às extremidades terão mais probabilidade de serem escolhidas como líder, uma vez que possuem valores mais altos de DF. É provável que essa dinâmica promova o aumento de diversidade do enxame. Além disso, o *Crowding Distance* consiste em uma abordagem local, uma vez que só é analisada a distância para os dois vizinhos mais próximos da partícula atual. Por se tratar de uma análise de toda a Frente de Pareto, garante ao DF um comportamento global, capaz de representar mais fielmente o estado em que o enxame se encontra e de definir os líderes de maneira mais eficaz.

Usando o cálculo do DF, combinado com o operador de Mutação introduzido por Coello Coello [21], é possível manter a diversidade das soluções não-dominadas do Arquivo Externo (AE). Este mecanismo é utilizado na seleção dos líderes globais e cognitivos, como também para ordenar e remover partículas não-dominadas pertencentes ao AE quando seu limite de partículas é atingido. A forma como esta abordagem é utilizada é explicada nas próximas seções.

4.2 Seleção do Líder Cognitivo

O processo de seleção do líder cognitivo ($\vec{P}_{best}$) é crucial para a convergência e eficiência do algoritmo. Para tanto, neste trabalho foi proposta uma solução similar à estratégia utilizada no MOPSO-CDR original. Na proposta de atualização do líder cognitivo deste trabalho, o $\vec{P}_{best}$ só é atualizado, caso a solução atual o domina. Caso eles sejam não-dominadas, a escolha é feita utilizando o AE. O uso do AE se torna necessário, pois é preciso identificar as soluções do AE mais próximas da partícula atual e do $\vec{P}_{best}$. Feito isso, é verificado qual das duas soluções apresenta maior DF; caso a solução mais próxima da posição atual tenha o maior valor de DF, o $\vec{P}_{best}$ tem seu valor atualizado para esta posição, caso contrário seu valor não é atualizado.

Utilizando essa abordagem é esperado que as partículas da população interna (*i.e.* enxame), sejam atraídas para regiões menos populadas uma vez que o líder cognitivo da partícula em questão possui um valor maior de DF. Além disso, como as partículas são guiadas também pelo termo cognitivo da equação de velocidade, essa seleção do líder cognitivo ajuda a impedir que as partículas se afastem das regiões menos populosas, resultado, deste modo, em um enxame com maior diversidade.

Para facilitar o entendimento da seleção do líder cognitivo, pode ser observado o esquema da Figura 4.2.

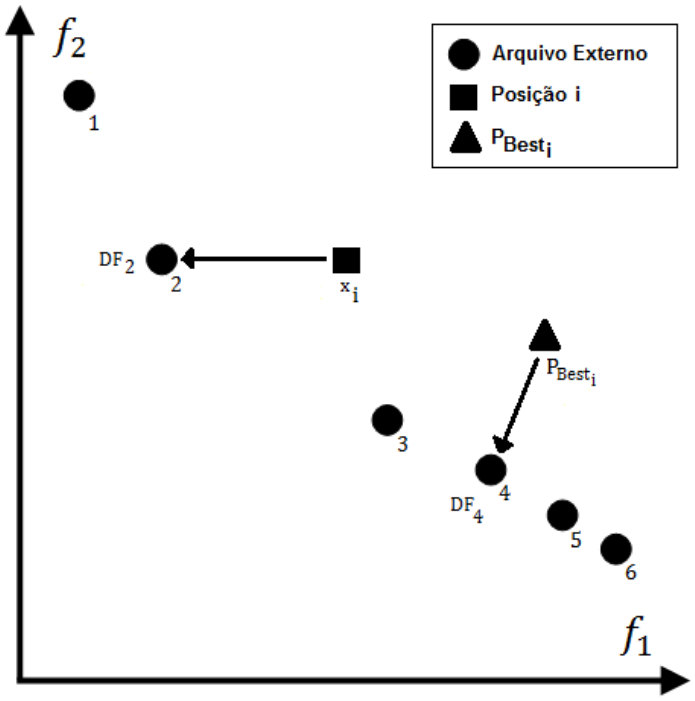


Figura 4.2: Ilustração da dinâmica da seleção do líder cognitivo.

Pode-se observar na Figura 4.2 que a foram encontradas as partículas da Frente de Pareto mais próximas da partícula i (cuja posição é representada por x_i) e $\vec{P}_{Best_i}$. Além disso x_i e $\vec{P}_{Best_i}$ são não-dominadas, ou seja, será feita uma análise para saber se $\vec{P}_{Best_i}$ será ou não atualizado. A partícula da Frente de Pareto mais próxima a x_i possui um valor DF_2 e a partícula mais próxima a $\vec{P}_{Best_i}$ possui valor DF_4. Como o valor de DF_4 é maior que DF_2, o valor de $\vec{P}_{Best_i}$ é atualizado para x_i.

4.3 Seleção do Líder Social

Em problemas multiobjetivo é fundamental escolher o líder social de forma apropriada, pois essa seleção afeta diretamente a capacidade de convergência do algoritmo e distribuição das soluções ao longo da Frente de Pareto. Como mencionado anteriormente, todas as soluções não dominadas estão presentes no AE, e estas são as possíveis candidatas a serem escolhidas como líder social da população interna. Para realizar essa seleção, o algoritmo MOPSO-DFR ordena a cada iteração o conjunto de soluções pertencentes ao AE utilizando como critério de ordenação o DF de cada partícula (quanto maior o DF, melhor). Para realizar a atualização da velocidade de cada partícula do enxame é necessário selecionar o líder social. O MOPSO-DFR seleciona

o líder social através da aplicação da roleta, onde as soluções do AE com maior DF possuem mais chances de serem escolhidas.

A Figura 4.3 exibe várias partículas da população interna e as partículas pertencentes ao AE.

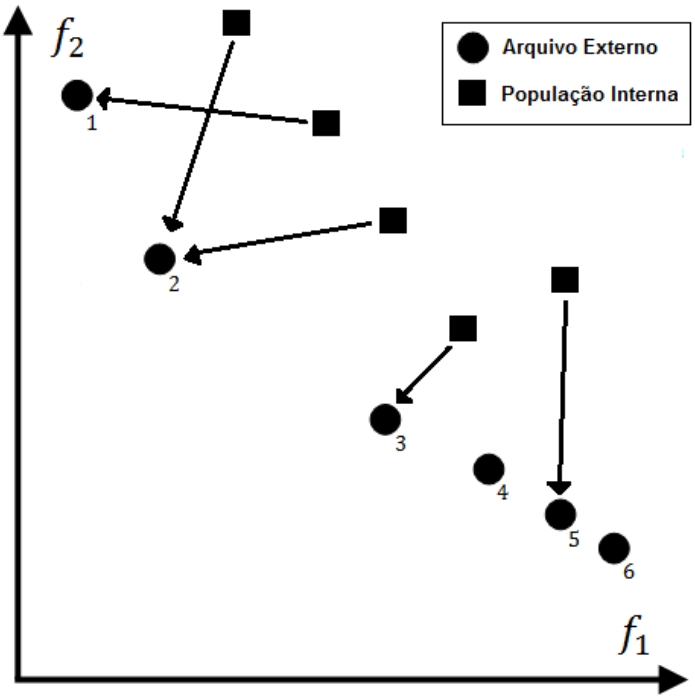


Figura 4.3: Dinâmica da seleção do líder social.

Além disso, podem ser vistas também na Figura 4.3 setas em direção à Frente de Pareto mostrando os líderes encontrados por cada partícula. Pode-se notar que mesmo partículas com valores mais baixos de DF podem ser escolhidos. Isso acontece devido à utilização da roleta: partículas com valores de DF mais altos tem maior probabilidade de serem escolhidos como líder de uma partícula.

Esse comportanto é bastante intessante, uma vez que o líder social está presente em um dos termos da equação de velocidade e desta forma influencia diretamente na movimentação do enxame para regiões menos populadas. Desta forma, garante-se ainda mais a tentativa de explorar a região de forma mais voltada para a melhoria do espaçamento e espalhamento do enxame. Com isso, espera-se que a proposta apresentada apresente uma melhoria na convergência, obtendo um melhor conjunto de soluções quando comparado com o MOPSO-CDR.

4.4 Arquivo Externo

Como foi visto durante os capítulos anteriores, o principal intuito na utilização do arquivo externo consiste em atuar como banco de soluções não-dominadas que foram encontradas durante todo o processo de otimização. Com o objetivo de evitar o excesso de soluções presentes no AE, faz-se necessário tomar algumas decisões quanto à inserção e remoção de partículas. Devido a este fato, normalmente limita-se a quantidade de soluções pertencentes ao AE e caso esse número seja atingido, as soluções serão ordenadas com base em algum critério e as últimas colocadas nesse *ranking* serão removidas do AE, mantendo a quantidade de máxima de partículas.

O algoritmo proposto neste trabalho adota como critério de ordenação o DF das partículas, ou seja, as partículas com os maiores valores de DF serão as primeiras colocadas no processo de ordenação. Após a ordenação ser realizada, as m piores partículas retornadas serão removidas do AE, onde m é a quantidade de partículas excedentes à quantidade máxima suportada pelo AE. Esse processo é também conhecido como poda do AE e é fundamental para funcionamento do algoritmo, uma vez que esse mecanismo irá evitar que o AE fique com uma quantidade de soluções maior que o necessário e acabe prejudicando a dinâmica do algoritmo.

A dinâmica de atualização (adição e remoção de partículas) do arquivo externo é exibida na Figura 4.4.

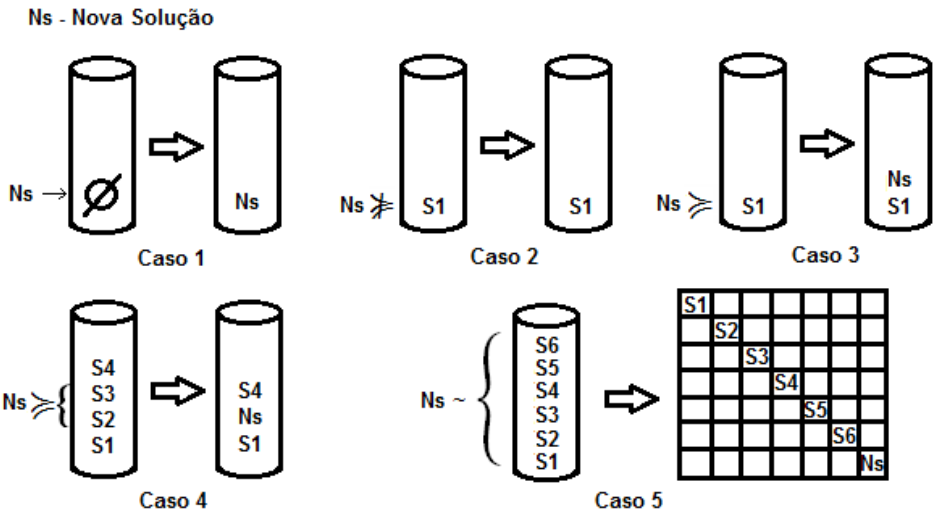


Figura 4.4: Possíveis casos de atualização do Arquivo Externo.

É possível ver no Caso 1 a adição de uma partícula nova (N_S) no arquivo externo ainda vazio. Já no Caso 2, a solução S_1 já pertence ao AE e devido ao fato da nova

solução N_S ser dominada por S_1, ela não é adicionada ao AE. O Caso 3 exibe uma solução candidata N_S que é incomparável em relação a S_1. Deste modo N_S é adicionada ao banco e S_1 é mantida. O Caso 4 exibe uma situação onde a N_S é incomparável em relação às S_1 e S_4 e além disso domina as soluções S_2 e S_3. Quando essa situação ocorre, as soluções dominadas (S_2 e S_3) são retiradas do AE. Por fim, o Caso 5 mostra que N_S é imconparável às demais soluções presentes no arquivo externo e, por isso, é adicionada ao banco junto com as demais soluções. Para este último caso, a Figura 4.4 exibe uma matriz simbolizando a Frente de Pareto em um problema de duas dimensões e todas as soluções presentes no AE são exibidas.

4.5 Nova equação de velocidade

Com o intuito de encontrar um enxame mais espaçado e espalhado é proposto neste trabalho uma nova equação de velocidade para o algoritmo MOPSO-DFR. Como o MOPSO-CDR já encontrou resultados significativos em termos de espaçamento e espalhamento, é proposto a adição de um novo termo na equação de velocidade do algoritmo MOPSO-DFR, um segundo termo social onde o líder do enxame levará em conta os valores de *crowding distance* das partículas do arquivo externo. A nova equação proposta pode ser vista na Equação (4.3).

$$\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}_i) + c_2 \cdot r_2 \cdot (\vec{n}_{best_{DF}} - \vec{x}_i) + c_3 \cdot r_3 \cdot (\vec{n}_{best_{CD}} - \vec{x}_i), \qquad (4.3)$$

em que c_3 é uma constante de aceleração associada ao segundo líder social, r_3 é um valor aleatório entre 0 e 1 usado no cálculo da velocidade de cada dimensão, a cada iteração e $\vec{n}_{best_{CD}}$ é segundo líder social encontrado, levando em conta o *crowding distance*. A forma como será calculado o líder social levando em conta o *crowding distance* é a mesma definida da proposta original do MOPSO-CDR apresentada no Capítulo 3.

Devido à adição de um termo na equação de velocidade, será necessário realizar um estudo paramétrico da nova constante de aceleração c_3. Por se tratar de mais um líder social, uma primeira abordagem consistiu em realizar uma soma dos valores de c_2 e c_3 e forçar essa soma a ser igual ao antigo valor da constante c_2. Entretanto foi necessário realizar um estudo mais aprofundado para validar e comprovar o benefício do termo adicional na equação de velocidade.

4.6 Pseudo-código

Como foi mencionado nas seções anteriores, algumas abordagens do MOPSO-CDR foram alteradas visando obter resultados mais satisfatórios. O pseudo-código do algoritmo MOPSO-DFR pode ser visto no Algoritmo 4.

Algoritmo 4: Pseudocódigo do MOPSO-DFR.

```
1  Inicialize o enxame;
2  Determine os líderes iniciais do arquivo externo;
3  Qualifique os líderes considerando DF e CD;
4  enquanto critério de parada não for alcançado faça
5      para cada partícula faça
6          Aplique o operador de turbulência utilizado no MOPSO;
7          Selecione líderes sociais usando DF e CD com seleção por roleta;
8          Atualize velocidade e posição;
9          Avalie a qualidade da partícula;
10         Atualize P⃗best(t) usando torneio binário;
11     Atualize líderes do arquivo externo;
12     Qualifique os líderes por DF e CD;
13 Retorne o Arquivo Externo.
```

Para testar e comprovar a eficácia do algoritmo proposto por este trabalho, foi testado para um conjunto bem definido de funções de teste. Esse conjunto de teste é introduzido no próximo capítulo assim como os resultados encontrados e um comparativo com outras técnicas de ennxame multiobjetivas presentes da literatura.

5 Resultados e Discussão

"O mundo e o universo são lugares extremamente belos,
e quanto mais os compreendemos mais belos eles parecem."
– Richard Dawkins

Este capítulo tem como objetivo apresentar o arranjo experimental definido para testar a eficácia do algoritmo proposto, exibir um comparativo dos resultados encontados pela técnica proposta com outros algoritmos de inteligência de enxame da literatura e discutir os resultados encontrados, assim como a contribuição gerada por este trabalho de conclusão de curso.

5.1 Arranjo Experimental

5.1.1 Funções de teste

Para comprovar a qualidade da proposta multiobjetiva deste trabalho de conclusão de curso foi necessário utilizar um conjunto de funções de teste propostas por Deb, Thiele, Laumanns e Zitzler [1]. Nesse estudo, Deb *et.al* identificaram várias características que podem dificultar o processo de convergência de um algoritmo multiobjetivo, dentre elas: não-uniformidade, multi-modalidade, influência de mínimos e máximos locais, convexidade ou não-convexidade, dentre outras. Para essas características foram criadas nove funções de teste, das quais sete estão sendo utilizadas para testar o algoritmo, uma vez que analisando os resultados para essas sete funções é possível identificar a melhor dentre as técnicas utilizadas no comparativo. São elas:

DTLZ-1

A primeira função de teste, intitulada DTLZ-1, é definida pelo conjunto de funções exibida na Equação (5.1), tendo M objetivos.

$$\begin{cases} \text{Minimizar } f_1(x) = \frac{1}{2}x_1x_2\cdots x_{M-1}(1+g(x_M)), \\ \text{Minimizar } f_2(x) = \frac{1}{2}x_1x_2\cdots(1-x_{M-1})(1+g(x_M)), \\ \cdots \\ \text{Minimizar } f_{M-1}(x) = \frac{1}{2}x_1(1-x_2)(1+g_(x_M), \\ \text{Minimizar } f_M(x) = \frac{1}{2}(1-x_1)(1+g_(x_M), \end{cases} \tag{5.1}$$

em que $0 <= x_i <= 1$, para $i = 1,2,\cdots,n$. E $g(x_M)$ requer $|x_M| = k$ variáveis e $g(x_M)$ pode ser qualquer função tal que $g >= 0$. A proposta original sugere $g(x_M)$ definida pela equação (5.2).

$$g(x_M) = 100|x_M| + \sum_{x_i \in x_M} (x_i - 0,5^2) - \cos[20\pi(x_i - 0,5))]. \tag{5.2}$$

Os valores da função objetiva repousam sobre o hiperplano linear: $\sum_{m=1}^{M} f_m = 0,5$ e é representada graficamente na Figura 5.1. A proposta original sugere um valor de $k = 5$. No problema acima o número total de variáveis $n = M + k - 1$. A Figura 5.1 exibe a Frente de Pareto encontrado pela execução algoritmo NSGA-II resolvendo o problema DTLZ-1 com 3 objetivos. As demais representações gráficas encontradas nessa seções foram obtidas através da execução do NSGA-II.

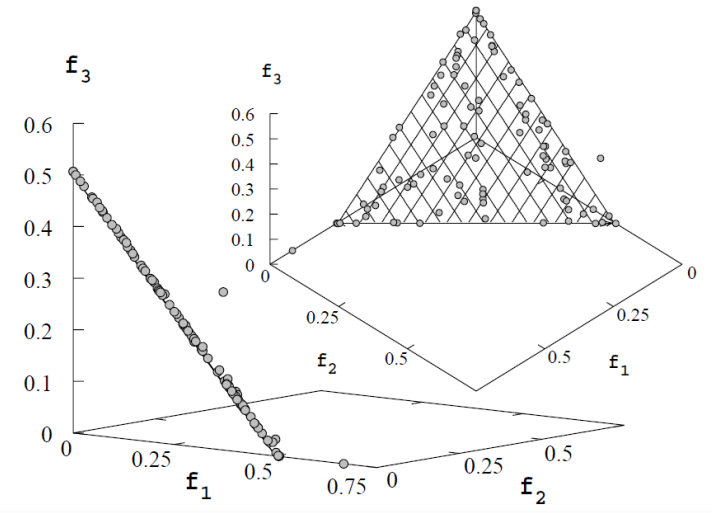


Figura 5.1: Representação gráfica da função de teste DTLZ-1. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

DTLZ-2

A segunda função teste apresentada por Deb *et al*, consiste em um problema esfera genérico definido pelo conjunto de Equações (5.3).

$$\begin{cases} \text{Minimizar } f_1(x) = (1+g(x_M))\cos\frac{x_1\pi}{2}\cdots\cos\frac{x_{M-1}\pi}{2}, \\ \text{Minimizar } f_2(x) = (1+g(x_M))\cos\frac{x_1\pi}{2}\cdots\sin\frac{x_{M-1}\pi}{2}, \\ \cdots \\ \text{Minimizar } f_M(x) = (1+g(x_M))\sin\frac{x_1\pi}{2}, \end{cases} \tag{5.3}$$

em que $0 <= x_i <= 1$, para $i = 1,2,\cdots,n$ e $g(x)$ é definida por:

$$g(x_M) = \sum_{x_i \in x_M} (x_i - 0,5^2) \tag{5.4}$$

Para este problema, é sugerido um número $k = |x_M| = 10$ e o número total de variáveis $n = M + k - 1$. A Figura 5.2 exibe graficamente o problema DTLZ-2 e um conjunto de soluções encontrado pelo NSGA-II.

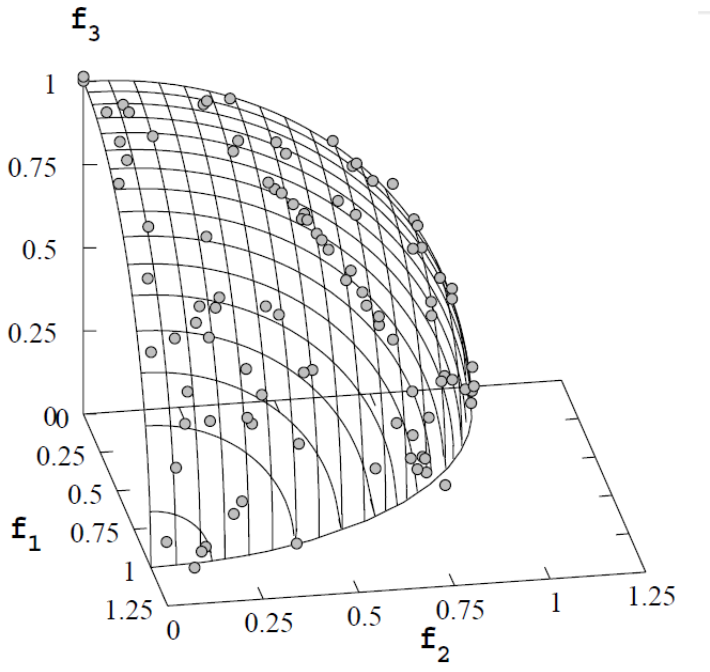


Figura 5.2: Representação gráfica da função de teste DTLZ-2. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

DTLZ-3

O próximo problema, o DTLZ-3, é definido pelas Equação apresentadas em (5.5).

$$\begin{cases} \text{Minimizar } f_1(x) = (1+g(x_M))\cos\frac{x_1\pi}{2}\cdots\cos\frac{x_{M-1}\pi}{2}, \\ \text{Minimizar } f_2(x) = (1+g(x_M))\cos\frac{x_1\pi}{2}\cdots\sin\frac{x_{M-1}\pi}{2}, \\ \cdots \\ \text{Minimizar } f_M(x) = (1+g(x_M))\sin\frac{x_1\pi}{2}, \end{cases} \tag{5.5}$$

em que $0 <= x_i <= 1$, para $i = 1,2,\cdots,n$ e $g(x)$ é definida pela Equação (5.6).

$$g(x_M) = 100|x_M| + \sum_{x_i \in x_M} (x_i - 0,5^2) - \cos(20\pi(x_i - 0,5)). \tag{5.6}$$

Para essa função de teste, é sugerido $k = |x_M| = 10$ e existe um total de $n = M + k - 1$ variáveis de decisão para esse problema. A Frente de Pareto verdadeira pode ser visto na Figura 5.3. O conjunto de soluções exibido na Figura 5.3 foi encontrado através da execução do NSGA-II.

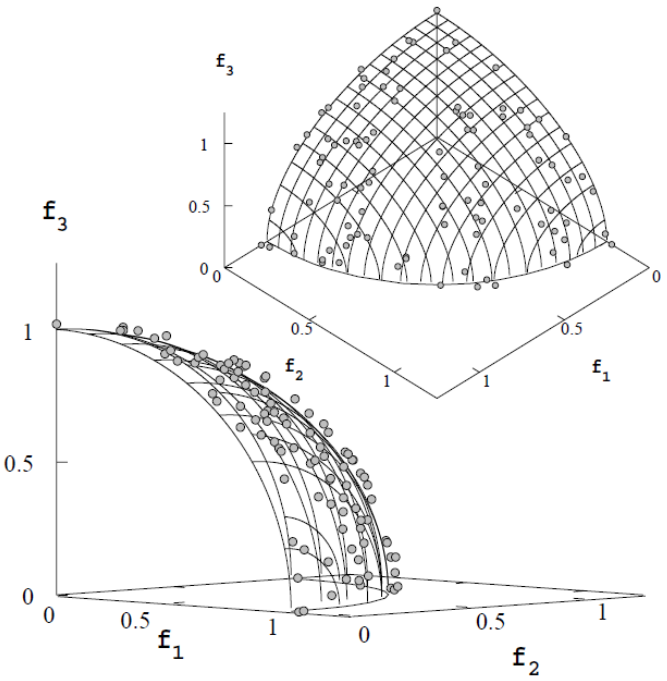


Figura 5.3: Representação gráfica da função de teste DTLZ-3. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

DTLZ-4

Com o intuito de investigar a habilidade do algoritmo multiobjetivo de manter uma boa distribuição de soluções, problema DTLZ-2 foi modificado com um mapeamento diferente da meta-variável. Esse problema é definido pelo conjunto de Equações (5.7).

$$\begin{cases} \text{Minimizar } f_1(x) = (1+g(x_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2)\cdots\cos(x_{M-2}^\alpha \pi/2)\cos(x_{M-1}^\alpha \pi/2), \\ \text{Minimizar } f_2(x) = (1+g(x_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2)\cdots\cos(x_{M-2}^\alpha \pi/2)\sin(x_{M-1}^\alpha \pi/2), \\ \text{Minimizar } f_3(x) = (1+g(x_M))\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2)\cdots\sin(x_{M-2}^\alpha \pi/2), \\ \cdots \\ \text{Minimizar } f_{M-1}(x) = (1+g(x_M))\cos(x_1^\alpha \pi/2)\sin(x_2^\alpha \pi/2), \\ \text{Minimizar } f_M(x) = (1+g(x_M))\sin(x_1^\alpha \pi/2), \end{cases} \tag{5.7}$$

em que $0 <= x_i <= 1$, para $i = 1,2,\cdots,n$ e $g(x)$ é definida pela Equação (5.8):

$$g(x_M) = \sum_{x_i \in x_M} (x_i - 0,5^2). \tag{5.8}$$

O parâmetro $\alpha = 100$ é sugerido para esse caso. Todas as variáveis de x_i a x_M variam em $[0,1]$ e para esse problema o valor de $k = 10$ é também sugerido. A representação gráfica do problema DTLZ-4 pode ser visto na Figura 5.4, onde é exibida também um conjunto de soluções encontrada pelo algoritmo NSGA-II.

DTLZ-5

O problema DTLZ-5 é uma variação do problema DTLZ-2 e é definida pelo conjunto de Equações (5.9).

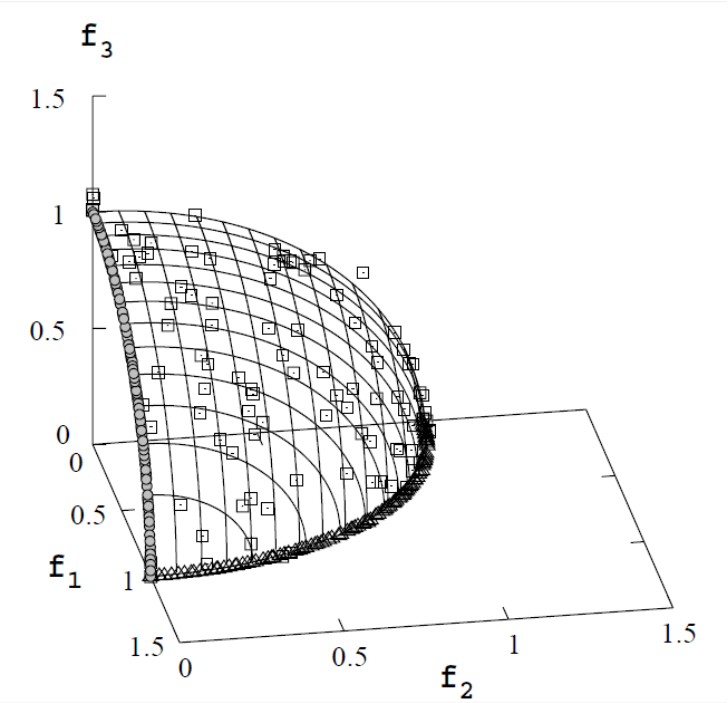


Figura 5.4: Representação gráfica da função de teste DTLZ-4. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

$$\begin{cases} \text{Minimizar } f_1(x) = (1+g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2), \\ \text{Minimizar } f_2(x) = (1+g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2), \\ \text{Minimizar } f_3(x) = (1+g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\sin(\theta_{M-2}\pi/2), \\ \cdots \\ \text{Minimizar } f_{M-1}(x) = (1+g(x_M))\cos(\theta_1\pi/2)\sin(\theta_2\pi/2), \\ \text{Minimizar } f_M(x) = (1+g(x_M))\sin(\theta_1\pi/2), \end{cases} \tag{5.9}$$

em que $\theta_i = \frac{\pi}{4(1+g(r))}(1+2g(r)x_i)$ para $i = 2,3,\cdots,(M-1)$. E g_{x_M} é definida pela Equação (5.10).

$$g(x_M) = \sum_{x_i \in x_M} (x_i - 0,5^2). \tag{5.10}$$

A função g com $k = |x_M| = 10$ é sugerida para este problema. E como nos outros casos, a quantidade total de variáveis de decisão para este problema é $n = M + k - 1$. Segundo Deb *et al* [1], esse problema irá testar a habilidade do algoritmo multiobjetivo de convergir para uma curva e também irá permitir uma forma mais fácil de demonstrar visualmente o desempenho do algoritmo multiobjetivo. A representação gráfica

da Frente de Pareto verdadeira assim como o conjunto de soluções encontrada pelo NSGA-II pode ser visto na Figura 5.5.

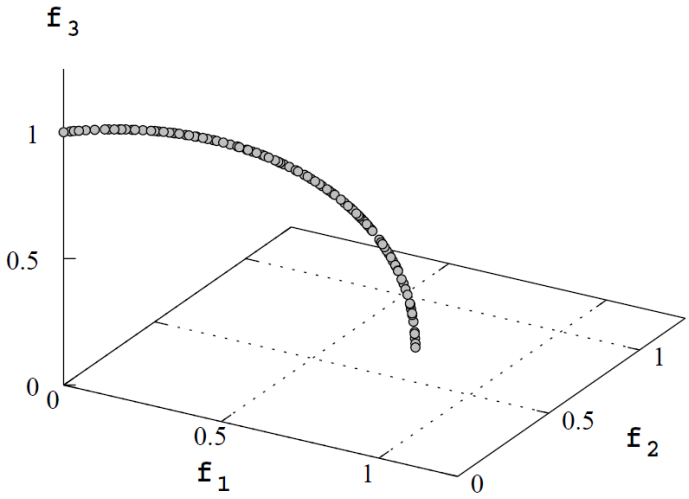


Figura 5.5: Representação gráfica da função de teste DTLZ-5. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

DTLZ-6

A penúltima função de teste utilizada por este trabalho, a DTLZ-6, é definida pelo conjunto de Equações (5.11).

$$\begin{cases} \text{Minimizar } f_1(x) = (1+g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2), \\ \text{Minimizar } f_2(x) = (1+g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2), \\ \text{Minimizar } f_3(x) = (1+g(x_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\cdots\sin(\theta_{M-2}\pi/2), \\ \cdots \\ \text{Minimizar } f_{M-1}(x) = (1+g(x_M))\cos(\theta_1\pi/2)\sin(\theta_2\pi/2), \\ \text{Minimizar } f_M(x) = (1+g(x_M))\sin(\theta_1\pi/2), \end{cases} \tag{5.11}$$

em que $\theta_i = \frac{\pi}{4(1+g(r))}(1+2g(r)x_i)$ para $i = 2,3,\cdots,(M-1)$.

A Equação $g(x_M)$ é definida pela Equação (5.12).

$$g(x_M) = \sum_{x_i \in x_M} (x_i^{0.1}). \tag{5.12}$$

O módulo do vetor x_M é igual a 10 para esse problema e o número total de variáveis

de decisão é o mesmo que no problema DTLZ-5. A modificação de g torna o processo de convergência para o pareto mais difícil do que na função DTLZ-5. Na Figura 5.6 é possível ver o pareto real assim como um conjunto de soluções encontrado pelo algoritmo NSGA-II.

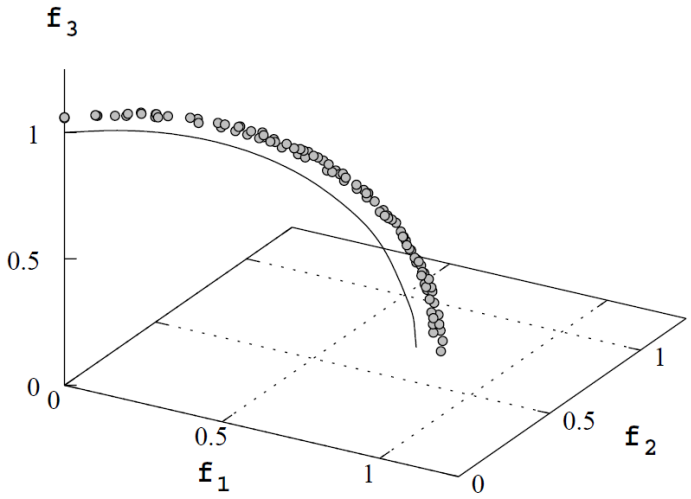


Figura 5.6: Representação gráfica da função de teste DTLZ-6. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

DTLZ-7

Por fim, a última função de teste utilizada neste trabalho de monografia, a DTLZ-7, é apresentada através do conjunto de Equações (5.13).

$$\begin{cases} \text{Minimizar } f_1(x) = x_1, \\ \text{Minimizar } f_2(x) = x_2, \\ \ldots \\ \text{Minimizar } f_{M-1}(x) = x_{M-1}, \\ \text{Minimizar } f_M(x) = (1+g(x_M))h(f_1, f_2, \cdots, f_{M-1}, g), \end{cases} \tag{5.13}$$

em que $g(x)$ é definido pela Equação (5.14):

$$g(x_M) = 1 + \frac{9}{|x_M|} \sum_{x_i \in x_M} x_i, \tag{5.14}$$

em que $0 <= x_i <= 1$ para $i = 1,2,\cdots,(M-1)$. E h é definida pela Equação (5.15), exibida abaixo:

$$h(f_1, f_2, \cdots, f_{M-1}, g) = M - \sum_{i=i}^{M-1} \frac{f_i}{1+g}[1+\sin(3\pi f_i)]. \tag{5.15}$$

Para essa última função de teste é sugerido $k = |x_M| = 20$ e existe um total de $n = M + k - 1$ variáveis de decisão para esse problema. Esse problema irá testar a habilidade do algoritmo manter subpopulações em diferentes regiões do Pareto-ótimo. Uma representação gráfica do pareto verdadeiro pode ser vista na Figura 5.7. O conjunto de soluções foi encontrado pelo algoritmo NSGA-II, na tentativa de solucionar o problema DTLZ-7.

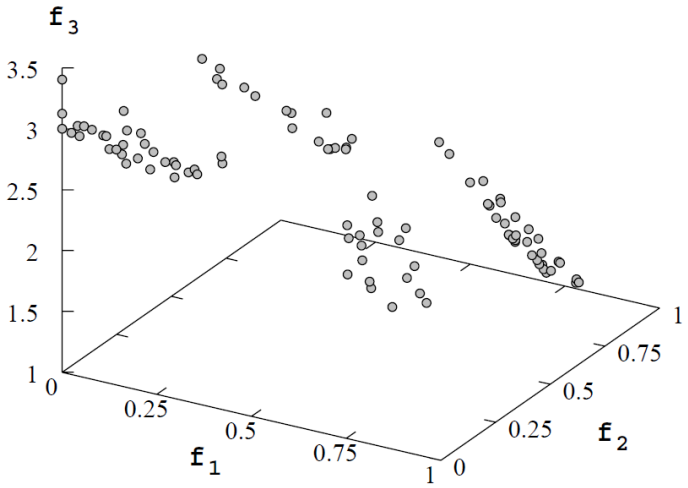


Figura 5.7: Representação gráfica da função de teste DTLZ-7. Frente de Pareto encontrada pela execução do NSGA-II. Adaptada de [1].

5.1.2 Métricas para avaliar o desempenho

As métricas utilizadas para avaliar o desempenho da técnica proposta por este trabalho, assim com nas demais técnicas utilizadas no comparativo são: *Coverage*, *Hypervolume*, *Spacing* e *Maximum Spread*. Essas métricas foram citadas na Seção 3.3 e através delas é possível analisar a qualidade de cada uma das técnicas multiobjetivas utilizada no comparativo. O cálculo das métricas é realizado através da análise do conjunto de soluções pertencentes ao AE gerado por cada técnica.

5.2 Parâmetros utilizados

O conjunto de parâmetros utilizados varia de acordo com a técnica utilizada. Todos os parâmetros utilizados são iguais aos sugeridos nas propostas originais. No caso do MOPSO-CDR, é utilizada uma taxa de mutação constante de $0,5$ e o fator de inércia diminui linearmente de $0,4$ a 0. Todas as varições do MOPSO-DFR utilizam o mesmo conjunto de parâmetros do MOPSO-CDR. Esses algoritmos possuem 100 partículas e as constantes de aceleração cognitiva e social têm valor $1,49445$. Já o SMPSO possui valor de coeficientes de aceleração aleatoriamente escolhidos entre $1,5$ e $2,5$ e este utiliza uma mutação polinomial a uma taxa de $0,166$. Para o NSGAII e SPEA2 foi utilizado como taxa de cruzamento $1,0$ e taxa de mutação $0,5$. O número máximo de soluções pertencentes ao arquivo externo foi definida como 100. O número total de iterações de todos os algoritmos foi definido como 300, e deste modo, um total de 300.000 avaliações da função *fitness* são realizadas em cada execução de cada técnica. Cada técnica foi executada 30 vezes.

O máquina utilizada para realizar as simulações de todos os experimentos possui como configuração processador Core i5-2300 2,80GHz com 4GB de RAM, executando o sistema operacional Windows 7 64 bits.

Para realizar o desenvolvimento do MOPSO-DFR e as simulações foi utilizado um *framework* em JAVA de algoritmos multiobjetivos intitulado *jMetal* [25][26]. O *jMetal* agrega diversas técnicas tanto baseada em enxames quanto evolucionárias, além de operadores e provê um ambiente de fácil desenvolvimento.

5.3 Experimentos

5.3.1 Análise paramétrica dos coeficientes de aceleração do MOPSO-DFR

Na técnica apresentada por este trabalho de conclusão de curso, um segundo termo social foi adicionado à equação de velocidade original. Essa alteração pode ser vista na Equação (5.16), já apresentada no Capítulo 4.

$$\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_{best} - \vec{x}_i) + c_2 \cdot r_2 \cdot (\vec{n}_{best_{DF}} - \vec{x}_i) + c_3 \cdot r_3 \cdot (\vec{n}_{best_{CD}} - \vec{x}_i). \quad (5.16)$$

Um dos elementos desse segundo termo social é o coeficiente de aceleração c_3. Como mais um termo está sendo adicionado à equação, é necessário encontrar os valores de c_1, c_2 e c_3 mais adequados. Para tanto, foi realizado uma análise paramétrica, onde foi testado um conjunto de valores para os três coeficientes de aceleração. Cada conjunto de valores foi atribuído a uma versão diferente da técnica MOPSO-DFR e esse arranjo pode ser visto na Tabela 5.1.

Tabela 5.1: Valores dos coeficientes de aceleração associados a cada versão da técnica MOPSO-DFR.

	c_1	c_2	c_3
MOPSO-DFR-A	1,49445	1,49445	1,49445
MOPSO-DFR-B	1,49445	1,49445	0,0
MOPSO-DFR-C	0,9963	0,9963	0,9963
MOPSO-DFR-D	1,49445	0,74722	0,74722
MOPSO-DFR-E	1,49445	0,96966	0,48483
MOPSO-DFR-F	1,49445	0,48483	0,96966
MOPSO-DFR-G	1,49445	1,93932	0,96966
MOPSO-DFR-H	1,49445	0,96966	1,93932

Os valores dos coeficientes foram atribuídos de acordo com os valores padrão $c_1 = c_2 = 1,49445$. Nota-se que o valor do coeficiente c_1 praticamente não variou do original. Essa escolha foi feita devido ao fato do termo adicional da equação de velocidade ser um segundo termo social e, portanto, penalizar o coeficiente de aceleração cognitivo não faria sentido. O único caso que esse coeficiente assume valor diferente do padrão é na versão MOPSO-DFR-C, onde foi realizada uma soma dos valores dos dois coeficientes originais, e à cada termo foi atribuído um terço desse valor. Nos demais casos, apenas os valores de c_1 e c_2 foram alterados, seguindo alguma combinação de frações do valor original.

Para realizar o estudo paramétrico da técnica MOPSO-DFR, foram utilizadas as funções de teste DTLZ-1, DTLZ-2 e DTLZ-3. Nas simulações realizadas foi primeiramente analisado os valores de *Coverage* obtidos pelos algoritmos. Foi constatado que a versão MOPSO-DFR-A encontrou as Frentes de Pareto que mais dominam o conjunto de soluções encontrado pelas demais versões e por isso apenas seus valores de *Coverage* em relação às demais técnicas serão exibidas nesse estudo. Por questões estéticas a métrica *Coverage* será abreviada para *Cov* e a técnica MOPSO-DFR-A, para DFR-A.

A Tabela 5.2 exibe os resultados da simulação das 8 variações para a função de

teste DTLZ-1. Em todas as células das tabelas deste capítulo são apresentadas a média seguida do desvio padrão em parênteses. O melhor valor médio encontrado é exibido em negrito. Pode-se observar que versões distintas do MOPSO-DFR, encontraram resultados diferentes. A versão MOPSO-DFR-F foi capaz de encontrar o melhor valor de *Spacing*, entretanto obteu o pior valor de *Max. Spread*. Já a versão MOPSO-DFR-A obteve o melhor valor de *Max. Spread*. MOPSO-DFR-G foi a versão responsável por obter o melhor valor de *Hypervolume* além de obter um valor satisfatório de *Max. Spread*. Contudo o MOPSO-DFR-A obteu valores de *Coverage* bem superiores em relação às versões MOPSO-DFR-B, MOPSO-DFR-C, MOPSO-DFR-D, MOPSO-DFR-E e MOPSO-DFR-F obtendo na maioria dos casos dominância com valores acima de 80%. Além disso, encontrou valores de *Coverage* equivalente às técnicas MOPSO-DFR-G e MOPSO-DFR-H.

Tabela 5.2: Comparativo das variações da técnica MOPSO-DFR para a função DTLZ-1 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR-A, *)	Cov(*, DFR-A)
MOPSO-DFR-A	7,3261 (2,8598)	**121,2050 (10,5595)**	0,4109 (0,1037)	-	-
MOPSO-DFR-B	4,1526 (4,1146)	63,0305 (45,0284)	0,4251 (0,1343)	**0,7903 (0,1037)**	0,0701 (0,1227)
MOPSO-DFR-C	6,3138 (5,4109)	79,2634 (46,4648)	0,5122 (0,1495)	**0,8114 (0,2656)**	0,0918 (0,1514)
MOPSO-DFR-D	4,3508 (4,4065)	64,6481 (45,3096)	0,5560 (0,1208)	**0,8980 (0,1897)**	0,0424 (0,0912)
MOPSO-DFR-E	4,8787 (3,6749)	75,7445 (45,9026)	0,5956 (0,1067)	**0,8455 (0,2496)**	0,0737 (0,1500)
MOPSO-DFR-F	**4,1471 (5,3289)**	59,5421 (42,0854)	0,5145 (0,1245)	**0,8969 (0,2127)**	0,0499 (0,1225)
MOPSO-DFR-G	8,8522 (4,3130)	117,671 (12,8739)	**0,3414 (0,1341)**	**0,4060 (0,2240)**	**0,3215 (0,2326)**
MOPSO-DFR-H	8,8285 (4,9589)	119,666 (14,0103)	0,3612 (0,1213)	**0,3799 (0,2657)**	**0,3700 (0,2475)**

A Tabela 5.3 mostra o resultados da simulação das técnicas MOPSO-DFR para a função DTLZ-2. Os valores de *Spacing* encontrados pelas variações MOPSO-DFR-D, MOPSO-DFR-E e MOPSO-DFR-F foram muitos próximos de zero. Isso significa dizer que as partículas encontradas estão equidistantes uma das outras e esse é o comportamento ideal de uma técnica multiobjetiva. Entretanto, pode-se notar que os valores de *Max. Spread* encontrados por essas técnicas foram baixos, o que pode ter acontecido devido a uma grande concentração de partículas devido a uma rápida convergência. As versões MOPSO-DFR-A, MOPSO-DFR-C e MOPSO-DFR-H obtiveram os melhores resultados de *Max. Spread* e resultados satisfatórios de *Spacing*.

Analisando os valores de *Hypervolume* encontrados, pode-se notar que as versões MOPSO-DFR-A, MOPSO-DFR-B, MOPSO-DFR-G e MOPSO-DFR-H encontraram valores satisfatórios e muito próximos um dos outros. Em relação ao *Coverage*, as Frentes de Pareto obtidas pela versão MOPSO-DFR-A, mostrou dominância em no mínimo 97% em relação à cinco das sete versões, além de obter resultados semalhantes às versões MOPSO-DFR-G e MOPSO-DFR-H.

Tabela 5.3: Comparativo das variação da técnica MOPSO-DFR para a função DTLZ-2 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR-A, *)	Cov(*, DFR-A)
MOPSO-DFR-A	0,0413 (0,0094)	1,0172 (0,0054)	**0,1869 (0,0057)**	-	-
MOPSO-DFR-B	0,0039 (0,0034)	0,0735 (0,0304)	0,4451 (0,0452)	**0,9813 (0,0467)**	0,0 (0,0)
MOPSO-DFR-C	0,0376 (0,0092)	1,0173 (0,0055)	0,1869 (0,0057)	**0,9907 (0,1037)**	0,0 (0,0)
MOPSO-DFR-D	**0,0006 (0,0008)**	0,0098 (0,0095)	0,4520 (0,0488)	**0,9997 (0,0017)**	0,0 (0,0)
MOPSO-DFR-E	**0,0006 (0,0014)**	0,0087 (0,0080)	0,4551 (0,0669)	**0,9730 (0,1453)**	0,0 (0,0)
MOPSO-DFR-F	0,0007 (0,0006)	0,0117 (0,0079)	0,4424 (0,0408)	**0,9983 (0,0073)**	0,0 (0,0)
MOPSO-DFR-G	0,0299 (0,0061)	1,0148 (0,0047)	0,1914 (0,0038)	**0,4331 (0,2098)**	**0,2703 (0,1720)**
MOPSO-DFR-H	0,0365 (0,0121)	**1,0189 (0,0054)**	0,1887 (0,0059)	**0,3329 (0,2140)**	**0,3899 (0,2243)**

O resultado das simulações das variações da técnica MOPSO-DFR para o problema DTLZ-3 pode ser visto na Tabela 5.4. Os valores de *Spacing* encontrados pelas versões das técnicas foram relativamente altos, quando comparados aos valores exibidos nas Tabelas 5.2 e 5.3, mas quem obteve o melhor resultado foi a versão MOPSO-DFR-D. Além disso, essa versão também obteve o melhor resultado de *Max. Spread* e valor satisfatório de Hypervolume. A versão que obteve o melhor resultado de *Hypervolume* foi MOPSO-DFR-A, seguido das versões MOPSO-DFR-G e MOPSO-DFR-H. A versão MOPSO-DFR-A obteve valores satisfatórios de *Coverage* em relação a parte das demais técnicas e resultados semalhantes às versões MOPSO-DFR-G e MOPSO-DFR-H

As versões das técnicas apresentadas possuem valores semelhantes de *Spacing*, *Maximum Spread* e *Hypervolume*, entretanto a técnica MOPSO-DFR-A mostrou melhores resultados na métrica *Coverage*. Dentre as quatro métricas analisadas por este trabalho, o *Coverage* é a mais importante. Devido a esse fato, a versão MOPSO-DFR-A será adotada como proposta deste trabalho e será utilizada no comparativo com as

Tabela 5.4: Comparativo das variação da técnica MOPSO-DFR para a função DTLZ-3 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR-A, *)	Cov(*, DFR-A)
MOPSO-DFR-A	42,2894 (12,1097)	187,9971 (10,4049)	**0,1273 (0,0437)**	-	-
MOPSO-DFR-B	**24,7200 (24,5390)**	158,1305 (43,2556)	0,1872 (0,0913)	**0,6591 (0,3810)**	0,1607 (0,2450)
MOPSO-DFR-C	31,8823 (23,8608)	183,1762 (65,1273)	0,2668 (0,1293)	**0,5081 (0,3630)**	0,2680 (0,2553)
MOPSO-DFR-D	30,9382 (18,5246)	**218,0787 (98,5954)**	0,3121 (0,1083)	**0,4235 (0,3649)**	0,3061 (0,2802)
MOPSO-DFR-E	37,1402 (26,6503)	205,9200 (75,9887)	0,2925 (0,1160)	**0,3634 (0,3437)**	0,3607 (0,2520)
MOPSO-DFR-F	32,2113 (22,1960)	197,5367 (83,0943)	0,2943 (0,1136)	**0,5129 (0,4037)**	0,2687 (0,2821)
MOPSO-DFR-G	37,0580 (15,0474)	192,3119 (8,52194)	0,1501 (0,0633)	**0,3638 (0,2553)**	**0,3292 (0,2372)**
MOPSO-DFR-H	37,9672 (20,5966)	189,6081 (7,6578)	0,1408 (0,0643)	**0,3747 (0,2174)**	**0,3761 (0,2611)**

demais técnicas.

5.3.2 Comparação do MOPSO-DFR com outras técnicas

Para realizar a comparação entre a técnica proposta por este trabalho de conclusão de curso, o MOPSO-DFR, e as demais técnicas foi necessário realizar simulações para cada técnica, utilizando as funções de testes DTLZ-1 a DTLZ-7 apresentadas na Seção 5.1.1. O resultado gerado pelas simulações são os resultados das aplicação das métricas nas soluções pertencentes ao Arquivo Externo de cada técnica.

As técnicas utilizadas para realizar o comparativo são: MOPSO-CDR, SMPSO, NSGA-II e SPEA2. As duas primeiras técnicas pertencem à classe de Inteligência de Enxames, enquanto as duas últimas pertencem à classe de Algoritmos Evolucionários. Todos as técnicas apresentadas no comparativo foram simuladas utilizando o *jMetal*.

A partir de então, a versão da técnica MOPSO-DFR-A será referenciada apenas como MOPSO-DFR e consiste na proposta final deste trabalho. Por questões estéticas as aparições nas tabelas da métrica *Coverage* será abreviada para *Cov* e como apenas está sendo analisado os valores da técnica MOPSO-DFR com as demais, será abreviado também o nome da técnica MOPSO-DFR para *DFR*.

A Tabela 5.5 exibe o valor das métricas de cada técnica para a função DTLZ-1. O resultado de *Spacing* encontrado por MOPSO-DFR foi equivalente ao valor encontrado

pela técnica MOPSO-CDR, entretanto obteve resultado inferior quando comparado às demais técnicas. Entretanto, a técnica MOPSO-DFR obteve o melhor resultado tanto no *Max. Spread* quanto no *Hypervolume*. Os valores de *Coverage* encontrados por MOPSO-DFR foi bem superior quando comparados às demais técnicas. A técnica proposta foi capaz encontrar valores de *Coverage* iguais a 100% quando comparadas a três das quatro técnicas apresentadas e obteve também um bom resultado quando comparado com o MOPSO-CDR (75%). Além disso, três das quatro técnicas obtiveram dominância em 0% quando comparadas ao MOPSO-DFR, além do valor encontrado pelo MOPSO-CDR insignificante.

Tabela 5.5: Resultado da simulação para a função DTLZ-1 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	7,3261 (2,8598)	**121,205 (10,5595)**	**0,4110 (0,1055)**	-	-
MOPSO-CDR	6,9623 (4,7167)	91,7448 (35,3433)	0,4582 (0,1394)	**0,7534** (0,2487)	0,0900 (0,1511)
SMPSO	**0,0016 (0,0005)**	2,0003 (0,0019)	0,4945 (9,1E-05)	**1,0 (0,0)**	0,0 (0,0)
NSGAII	0,0073 (0,0007)	2,0041 (0,0043)	0,4936 (0,0003)	**1,0 (0,0)**	0,0 (0,0)
SPEA2	0,1129 (0,4927)	3,5256 (6,8698)	0,5331 (0,1193)	**1,0 (0,0)**	0,0 (0,0)

A Tabela 5.6 mostra os resultados encontrados utilizando a função de teste DTLZ-2. Os valores de *Spacing* e *Max. Spread* encontrados pela técnica MOPSO-DFR foi semelhante aos valores encontrados pelas demais técnicas. A técnica MOPSO-DFR obteve o melhor resultado de *Hypervolume* quando comparado aos resultados das demais técnicas, porém com valores bem próximos. Os valores de *Coverage* encontrados por MOPSO-DFR apresentam dominância em relação às demais técnicas com valores superiores a 95%. Todos os valores de *Coverage* pelas demais técnicas em relação ao MOPSO-DFR foram iguais a 0%.

Na Tabela 5.7 pode ser visto os resultados encontrados na execução da função de teste DTLZ-3. A técnica MOPSO-DFR obteve resultado de *Spacing* inferior em relação às demais técnicas. Já os valores de *Max. Spread* obtidos foram superiores às técnicas NSGAII, SPEA2 e SMPSO, e inferior quando comparado com a técnica MOPSO-CDR. A técnica MOPSO-DFR obteve, mais uma vez, o melhor resultado de *Hypervolume* em relação às demais técnicas. Os valores de *Coverage* encontrados foram bastante satisfatórios em relação às técnicas, SMPSO, NSGAII e SPEA2, onde houve dominância de 100%. O valor de *Coverage* encontrado pela técnica MOPSO-CDR tem uma tendência a ser melhor que a técnica MOPSO-DFR, mas como os

Tabela 5.6: Resultado da simulação para a função DTLZ-2 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	0,0413 (0,0094)	**1,0171 (0,0054)**	**0,1869 (0,0057)**	-	-
MOPSO-CDR	**0,0011 (0,0001)**	1,0 (4,2E-08)	0,2108 (1,8E-05)	**0,9610 (0,0373)**	0,0 (0,0)
SMPSO	0,0016 (0,0002)	1,0 (4,4E-05)	0,2105 (6,8E-05)	**0,9603 (0,0359)**	0,0 (0,0)
NSGAII	0,0068 (0,0006)	1,0014 (0,0053)	0,2114 (0,0078)	**0,9527 (0,0482)**	0,0 (0,0)
SPEA2	0,0034 (0,0003)	1,0009 (0,0015)	0,2116 (0,0024)	**0,9533 (0,0487)**	0,0 (0,0)

desvios padrão encontrados foram relativamente altos, não é possível afirmar que resultado obtido tem significância estatística.

Tabela 5.7: Resultado da simulação para a função DTLZ-3 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	42,2894 (12,1096)	187,9971 (10,4049)	**0,1273 (0,0437)**	-	-
MOPSO-CDR	33,9158 (22,4035)	**208,6751 (19,6458)**	0,1738 (0,0672)	0,2760 (0,2939)	**0,4512 (0,2989)**
SMPSO	3,7927 (5,7577)	18,4984 (26,7290)	0,3351 (0,2789)	**1,0 (0,0)**	0,0 (0,0)
NSGAII	**0,0277 (0,0430)**	1,4890 (0,6076)	0,2632 (0,1122)	**1,0 (0,0)**	0,0 (0,0)
SPEA2	0,0689 (0,1585)	1,4728 (0,5897)	0,2982 (0,1680)	**1,0 (0,0)**	0,0 (0,0)

A Tabela 5.8 mostra os resultados encontrados pela simulação do problema DTLZ-4. É possível ver um valor de *Spacing* do MOPSO-DFR próximo aos demais algoritmos, mas ainda assim, um pouco inferior. Os valores de *Max. Spread* obtidos por MOPSO-DFR são um pouco superiores aos encontrados pelas demais técnicas. A técnica MOPSO-DFR obteve o melhor resultado de *Hypervolume* quando comparado com os valores obtidos pelas demais técnicas. Em relação ao *Coverage*, obteve em dominância resultados superiores a 80%. O MOPSO-DFR praticamente foi dominado em praticamente 0% em relação às demais técnicas.

Na Tabela 5.9 pode ser visto os resultados das métricas obtidos através da simulação do problema DTLZ-5. Dentre os resultados exibidos, é possível notar um valor de *Spacing* do MOPSO-DFR semelhante às demais técnicas. O valor de *Max. Spread* encontrado por MOPSO-DFR foi equivalente aos resultados obtidos pelas demais técnicas. O resultado de *Hypervolume* encontrado pela técnica MOPSO-DFR foi melhor em relação aos demais resultados obtidos. A técnica proposta obteve valor de *Cove-*

Tabela 5.8: Resultado da simulação para a função DTLZ-4 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	0,1157 (0,0308)	**1,0204 (0,0071)**	**0,1313 (0,0185)**	-	-
MOPSO-CDR	**0,0017 (0,0003)**	1,0 (2,2E-07)	0,2105 (0,0002)	**0,804 (0,1399)**	0,0 (0,0)
SMPSO	0,0018 (0,0004)	1,0003 (0,0003)	0,2109 (0,0005)	**0,8080 (0,1358)**	0,0 (0,0)
NSGAII	0,0064 (0,0018)	0,8011 (0,4006)	0,1692 (0,0846)	**0,8457 (0,1401)**	0,0018 (0,0099)
SPEA2	0,0032 (0,0009)	0,7334 (0,4422)	0,1699 (0,1044)	**0,8330 (0,1546)**	0,0 (0,0)

rage satisfatórios, onde tem dominância, no pior dos casos igual a 95%. As demais técnicas em relação ao MOPSO-DFR obtiveram 0% de dominância em todos os casos.

Tabela 5.9: Resultado da simulação para a função DTLZ-5 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	0,0338 (0,0078)	**1,0262 (0,0051)**	**0,1859 (0,0064)**	-	-
MOPSO-CDR	**0,0011 (0,0002)**	1,0079 (3,5E-08)	0,2108 (2,3E-05)	**0,9620 (0,0512)**	0,0 (0,0)
SMPSO	0,0016 (0,0003)	1,0080 (2,9E-05)	0,2105 (7,4E-05)	**0,9607 (0,0526)**	0,0 (0,0)
NSGAII	0,0070 (0,0006)	1,0089 (0,0026)	0,2107 (0,0040)	**0,9530 (0,0645)**	0,0 (0,0)
SPEA2	0,0033 (0,0003)	1,0095 (0,0049)	0,2126 (0,0072)	**0,9537 (0,0650)**	0,0 (0,0)

Os resultados para a função DTLZ-6, podem ser vistos na Tabela 5.10. Os resultados encontrados pelo MOPSO-DFR foram superiores quando comparados ao MOPSO-CDR, mas inferiores em relação às demais técnicas. O valor de *Max. Spread* obtido foi semelhante aos encontrados pelas outras técnicas. O valor de *Hypervolume* encontrado MOPSO-DFR foi bem melhor do que os obtidos pelas técnicas evolucionárias, mas similares aos encontrados por MOPSO-CDR e SMPSO. O MOPSO-DFR foi dominado em 99%, em relação ao MOPSO-CDR e obteve bom resultado apenas quando comparado ao SMPSO. Também obteve valor de *Coverage* similare ao NSGA-II e superior ao SPEA-2.

Por fim, a Tabela 5.11 exibe os resultados encontrados na simulação para a função de teste DTLZ-7. O valor de *Spacing* encontrado por MOPSO-DFR foi um pouco maior que os demais, mas ainda sim, consiste em um resultado satisfatório. O valor de *Max. Spread* encontrado por MOPSO-DFR foi similar aos valores encontrados pelas demais técnicas, entretanto um pouco inferior. No entanto, resultado de *Hypervolume* obtido

Tabela 5.10: Resultado da simulação para a função DTLZ-6 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	0,0292 (0,0085)	1,0123 (0,0087)	0,1910 (0,0046)	-	-
MOPSO-CDR	0,1965 (0,0969)	**1,6378 (0,2319)**	**0,1797 (0,0558)**	0,0 (0,0)	**0,9931 (0,0366)**
SMPSO	**0,0014 (0,0002)**	1,0079 (4,4E-16)	0,2107 (3,8E-05)	**0,8433 (0,0640)**	0,0018 (0,0048)
NSGAII	0,0111 (0,0029)	1,0397 (0,0349)	0,4936 (0,0003)	**0,3607 (0,3497)**	0,4058 (0,3885)
SPEA2	0,0135 (0,0193)	1,0865 (0,1190)	0,2783 (0,1035)	**0,5406 (0,3252)**	0,2054 (0,2605)

por MOPSO-DFR consistiu no melhor resultado obtido, apesar dos demais estarem bem próximos. Os resultados de *Coverage* obtidos foram satisfatórios em relação às demais técnicas onde na maioria dos casos houve dominância superior à 65%. Quando comparadas as demais técnicas ao MOPSO-DFR, houve dominância de 0% em todos os casos.

Tabela 5.11: Resultado da simulação para a função DTLZ-7 com 300.000 chamadas.

	Spacing	Max. Spread	Hypervolume	Cov(DFR, *)	Cov(*, DFR)
MOPSO-DFR	0,0176 (0,0121)	0,6367 (0,0455)	**0,2900 (0,0538)**	-	-
MOPSO-CDR	0,0013 (0,0002)	0,7451 (0,0005)	0,3346 (0,0008)	**0,4582 (0,1028)**	0,0 (0,0)
SMPSO	**0,0011 (0,0001)**	0,7452 (0,0004)	0,3345 (0,0004)	**0,6650 (0,1044)**	0,0 (0,0)
NSGAII	0,0055 (0,0005)	0,7453 (0,0013)	0,3344 (0,0022)	**0,6570 (0,1061)**	0,0 (0,0)
SPEA2	0,0037 (0,0004)	**0,7451 (0,0004)**	0,3345 (0,00078)	**0,6983 (0,0913)**	0,0 (0,0)

6 Conclusões e Trabalhos Futuros

"A educação tem raízes amargas
mas os seus frutos são doces. "
Aristóteles

Neste trabalho de conclusão de curso foi proposto um novo método de otimização multiobjetivo baseado em enxames, com o intuito de melhorar o desempenho, aumentar a diversidade do conjunto de soluções encontradas, como também contribuir com as pesquisas na área de algoritmos de inteligência de enxames aplicativo em problemas multiobjetivos.

Neste capítulo, serão feitas considerações finais sobre os resultados encontrados e ao comparativo feito com outras técnicas multiobjetivas da literatura. Por fim, os trabalhos futuros serão apresentados.

6.1 Contribuições e Conclusões

Devido ao avanço geral da computação e ao surgimento de diversos problemas complexos a serem resolvidos, é necessário realizar um estudo mais aprofundado nas técnicas de otimização de único e vários objetivos, com o intuito de melhorar as soluções produzidas. Este trabalho de conclusão de curso contribui com essa área de estudo que vem sido bastante estudada pela academia durante os últimos anos, que é a área de algoritmos de inteligência de enxames capazes de resolver problemas com objetivos conflitantes.

O algoritmo desenvolvido por este trabalho, consiste em uma variação do MOPSO-CDR, incorporando um novo mecanismo de seleção de líderes sociais e cognitivos, como também um critério de ordenação das soluções pertencentes ao Arquivo Externo (AE). A técnica desenvolvida encontrou resultados satisfatórios em diversas funções de testes quando comparadas à outras técnicas presentes na literatura, tanto da

área de Inteligência de Enxames quanto de Algoritmos Evolucionários e desta forma contribui de forma direta com essa área de pesquisa. Entretanto, ainda nota-se que é possível melhorar o desempenho da técnica proposta utilizando valores mais adequados de coeficientes de aceleração.

Tais contribuições foram comprovadas através dos resultados obtidos pelo comparativo realizada com as demais técnicas multiobjetivas. A técnica proposta foi capaz de encontrar resultados de *Coverage* superior às demais técnicas analisadas e bons resultados de *Spacing*, *Maximum Spread* e *Hypervolume*. As funções utilizadas nesses comparativos são bastante utilizadas pela comunidade acadêmica devido a sua capacidade de analisar de forma eficaz técnicas de otimização multiobjetivas.

6.2 Trabalhos Futuros

Como trabalhos futuros da técnica MOPSO-DFR pode ser realizado primeiramente um estudo paramétrico mais detalhado dos coeficientes de aceleração do algoritmo. Esses coeficientes, como vistos durante este trabalho, impactam diretamente no desempenho das técnicas baseadas em enxame.

Também pode ser proposto um mecanismo adaptativo de atualização desses coeficientes de aceleração. Esse mecanismo deve levar em conta informações correntes do enxame para inferir que decisões devem ser tomadas quanto à atualização dos coeficientes de aceleração. Ainda não existe na literatura nenhuma abordagem adaptativa de algoritmos de otimização multiobjetivos.

Além disso, o algoritmo proposto por este trabalho poderá ser aplicado à problemas com muitos objetivos (do inglês, *Many Objective Problems*), ou seja, problemas com mais de 3 objetivos e poderá analisar a escalabilidade e dinâmica do algoritmo na tentativa de resolver esses problemas mais complexos.

Pode ser realizado uma paralelização do MOPSO-DFR em placas de video NVIDIA CUDA, com o intuito de medir a capacidade de paralelização da técnica proposta. É possível que implementado de forma paralela, o MOPSO-DFR consiga diminuir de forma significativa seu tempo de execução.

E por fim, o MOPSO-DFR pode ser utilizado para resolver problemas do mundo real e, deste modo, analisar o desempenho e a capacidade do algoritmo encontrar um conjunto de soluções satisfatório para o problema a ser resolvido.

Referências Bibliográficas

[1] DEB, K. et al. Scalable test problems for evolutionary multiobjective optimization. In: ABRAHAM, A.; JAIN, L.; GOLDBERG, R. (Ed.). *Evolutionary Multiobjective Optimization*. : Springer London, 2005, (Advanced Information and Knowledge Processing). p. 105–145. ISBN 978-1-85233-787-2.

[2] COELLO, C.; PULIDO, G.; LECHUGA, M. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, v. 8, n. 3, p. 256–279, 2004.

[3] KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of IEEE Internacional Conference on Neural Networks*. Perth, Aust: IEEE, 1995. p. 1945–1948.

[4] HOLLAND, J. H. *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0-262-58111-6. Disponível em: <http://portal.acm.org/citation.cfm?id=129194>.

[5] DORIGO, M.; STüTZLE, T. *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004. ISBN 0262042193.

[6] KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, v. 39, n. 3, p. 459–471, nov. 2007. Disponível em: <http://dx.doi.org/10.1007/s10898-007-9149-x>.

[7] ZITZLER, E.; LAUMANNS, M.; THIELE, L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: GIANNAKOGLOU, K. C. et al. (Ed.). *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. Athens, Greece: International Center for Numerical Methods in Engineering, 2001. p. 95–100.

[8] DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 6, n. 2, p. 182–197, 2002.

[9] GARZA-FABRE, M.; TOSCANO-PULIDO, G.; COELLO, C. A. C. Two novel approaches for many-objective optimization. In: *IEEE Congress on Evolutionary Computation*. 2010. p. 1–8.

[10] COELLO, C.; LECHUGA, M. Mopso: A proposal for multiple objective particle swarm optimization. In: IEEE. *Proceedings of the Congress on Evolutionary Computation (CEC 2002)*. Honolulu, HI, EUA, 2002. v. 2, p. 1051–1056.

[11] NEBRO, A. J. et al. SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In: *2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009)*. : IEEE Press, 2009. p. 66–73.

[12] TSOU, C.; CHANG, S.; LAI, P. *Using Crowding Distance to Improve Multi-Objective PSO with Local Search - Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*. : IN-Tech Education and Publishing, 2007.

[13] SANTANA, R. A.; PONTES, M. R.; BASTOS-FILHO, C. J. A. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In: *ISDA '09: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*. Pisa, Itália: IEEE, 2009.

[14] ZHAN, Z.-H. et al. Adaptive particle swarm optimization. *Trans. Sys. Man Cyber. Part B*, IEEE Press, Piscataway, NJ, USA, v. 39, n. 6, p. 1362–1381, dez. 2009. ISSN 1083-4419. Disponível em: <http://dx.doi.org/10.1109/TSMCB.2009.2015956>.

[15] KENNEDY, J. Population structure and particle swarm performance. In: *Proceedings of the Congress on Evolutionary Computation (CEC 2002)*. : IEEE Press, 2002. p. 1671–1676.

[16] KENNEDY, J.; MENDES, R. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, v. 36, n. 4, p. 515–519, 2006. Disponível em: <http://dx.doi.org/10.1109/TSMCC.2006.875410>.

[17] EBERHART, R. C.; SHI, Y. Particle swarm optimization: Developments, applications and resources,"proc. *IEEE Int. Congress Evolutionary Computation*, n. 1, p. 81–86, 2001.

[18] EBERHART, R. C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: *in Proceedings 2000 Congress Evolutionary Computation*. p. 84–88.

[19] CLERC, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. 1999. v. 3. Disponível em: <http://dx.doi.org/10.1109/CEC.1999.785513>.

[20] CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 1, p. 58–73, 2002. Disponível em: <http://dx.doi.org/10.1109/4235.985692>.

[21] COELLO, C.; LAMONT, G.; VELDHUIZEN, D. V. *Evolutionary algorithms for solving multi-objective problems*. Nova York: Springer-Verlag New York Inc, 2007.

[22] SANTANA, R.; PONTES, M.; BASTOS-FILHO, C. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In: IEEE. *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*. Pisa, Itália, 2009. p. 237–242.

[23] RISCO-MARTíN, J. L. et al. Optimization of dynamic data types in embedded systems using devs/soa-based modeling and simulation. In: *InfoScale '08: Proceedings of the 3rd international conference on Scalable information systems*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. p. 1–11. ISBN 978-963-9799-28-8.

[24] ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, v. 3, n. 4, p. 257–271, 1999.

[25] DURILLO, J. J.; NEBRO, A. J. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, v. 42, p. 760–771, 2011. ISSN 0965-9978. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0965997811001219>.

[26] DURILLO, J.; NEBRO, A.; ALBA, E. The jmetal framework for multi-objective optimization: Design and architecture. In: *Congress on Evolutionary Computation 2010*. Barcelona, Spain: , 2010. p. 4138–4325.