ANDRÉ BRITTO DE CARVALHO

# NOVAS ESTRATÉGIAS PARA OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS APLICADAS A PROBLEMAS COM MUITOS OBJETIVOS

CURITIBA

2013

ANDRÉ BRITTO DE CARVALHO

# NOVAS ESTRATÉGIAS PARA OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS APLICADAS A PROBLEMAS COM MUITOS OBJETIVOS

Tese de doutorado apresentada como requisito à obtenção do grau de Doutor. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Profa. Dra. Aurora Pozo

CURITIBA

2013

# AGRADECIMENTOS

Inicialmente gostaria de agradecer à professora Aurora Pozo pela amizade, companheirismo e por todo conhecimento que ela me passou nesses 6 anos de trabalho. Agradeço pela orientação tanto no mestrado e no doutorado e pela oportunidade de efetuar um trabalho de alta qualidade.

À professora Sanaz Mostaghim, orientadora no período de doutorado sanduíche, que me recebeu de forma muito amigável e que introduziu diversas contribuições no meu trabalho mesmo num curto período de convívio.

À CAPES, pelo financiamento deste doutorado.

Aos amigos do grupo de pesquisa que proporcionaram diversos momentos de descontração e discussões valiosas para o desenvolvimento do meu trabalho.

À minha esposa Beatriz, por todo amor, confiança, companheirismo e suporte que ela me deu durante todos esses anos de estudo.

Aos meus pais, por demonstrarem muito amor, confiança e orgulho, e por todo apoio dado durante todo o meu doutorado.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE SÍMBOLOS

$AIS$ *Artificial Immune Systems*

$AR$ *Average Ranking*

$AG$ *Adaptive Grid*

$BR$ *Balanced Ranking*

$CD$ *Crowding Distance*

$CDAS$ *Controlling Dominance Area of Solutions*

$EA$ *Evolutionary Algorithms*

$EMO$ *Evolutionary Multi-Objective Optimization*

$GD$ *Generational Distance*

$IBEAs$ *Indicator Based Evolutionary Algorithms*

$IGD$ *Inverted Generational Distance*

$IBEA$ *Indicator Based Evolutionary Algorithm*

$LD$ *Largest Distance*

$MaOP$ *Many-Objective Optimization Problem*

$MGA$ *Multi-Level Grid Archiving*

$MOEAs$ *Multi-Objective Evolutionary Algorithms*

$MOP$ *Multi-Objective Optimization Problem*

$MOPSO$ *Multiobjective Particle Swarm Optimization*

$MR$ *Maximum Ranking*

$NSGA-II$ *Non-dominated Sorting Genetic Algorithm-II*

$PSO$ *Particle Swarm Optimization*

# RESUMO

Problemas de otimização multiobjetivo possuem mais de uma função objetivo que estão em conflito. Devido a essa característica, não existe somente uma melhor solução, mas sim um conjunto com as melhores soluções do problema, definidas pelos conceitos da teoria da Otimalidade de Pareto. Algoritmos Evolucionários Multiobjetivo são aplicados com sucesso em diversos Problemas de Otimização Multiobjetivo. Dentre esses algoritmos, os baseados na Otimização por Nuvem de Partículas Multiobjetivo (MOPSO) apresentam bons resultados para problemas multiobjetivo e se destacam por possuirem características específicas, como a cooperação entre as partículas da população. Porém, quando o número de funções objetivo cresce, os algoritmos evolucionários multiobjetivo baseados em dominância de Pareto encontram algumas dificuldades em definir quais são as melhores soluções e não efetuam uma busca que converge para as soluções ótimas do problema. A Otimização com muitos objetivos é uma área nova que visa propor novos métodos para reduzir a deterioração da busca desses algoritmos em problemas de otimização com muitos objetivos (problemas com mais de três funções objetivo). Assim, motivado por esse campo de pesquisa ainda em aberto e pelo fato da meta-heurística MOPSO ser pouco utilizada na Otimização com Muitos Objetivos, este trabalho de doutorado contribuí com a proposta de novas métodos e algoritmos que buscam explorar três diferentes aspectos da Otimização por Nuvem de Partículas Multiobjetivo: uso de novas relações de preferências, métodos de arquivamento e algoritmos MOPSO com múltiplos enxames. Neste estudo, é feita uma análise empírica que utiliza um conjunto de indicadores de qualidade e problemas de *benchmark* com o intuito de analisar aspectos como convergência e diversidade da busca dos algoritmos utilizados. Por fim, esta tese traça os principais caminhos que serão seguidos nos trabalhos futuros.

Palavras-chave: Otimização com Muitos Objetivos. Otimização por Nuvem de Partículas. Otimização Multiobjetivo.

# ABSTRACT

Multiobjective Optimization Problems have more than one objective function that are often in conflict. Therefore, there is no single best solution, but a set of the best solutions defined by the concepts of Pareto Optimality theory. Multiobjective Evolutionary Algorithms are applied successfully in several Multiobjective Optimization Problems. Among these algorithms, we highlight those based on Multiobjective Particle Swarm Optimization (MOPSO), since they have good results for multiobjective problems and exhibit unique characteristics such as cooperation among individuals of the population. However, Multi-Objective Evolutionary Algorithms scale poorly when the number of objectives increases. Many-Objective Optimization Problems are problems that have more than three objective functions. Many-Objective Optimization is a new area, which aims to propose new methods to reduce the deterioration of these algorithms. Thus, motivated by this research field still open and the fact that MOPSO algorithms are still underused in Many-Objective Optimization, this work aims to propose new methods for MOPSO metaheuristic applied to this context. The main contribution of this PhD work is the proposal of new methods and algorithms that seek to explore three different aspects of MOPSO metaheuristic: the use of new preference relations, exploring methods of archiving and exploring multi-swarm algorithms. Another important feature presented in this thesis are the empirical analyzes used to validate all new techniques. In this study, we use a set of quality indicators and benchmark problems in order to analyze aspects such as convergence and diversity of the search. Finally, this thesis outlines the main paths that will be followed in future work.

Keywords: Many-Objective Optimization. Particle Swarm Optimization. Multi-Objective Optimization.

# CAPÍTULO 1

# INTRODUÇÃO

Os Problemas de Otimização Multiobjetivo, (MOP, do inglês *Multi-Objective Optimization Problems*) possuem mais de uma função objetivo. Nestes problemas, as funções objetivo que são otimizadas são conflitantes, logo não há somente uma melhor solução, mas um conjunto com as melhores soluções. Para obter esse conjunto de soluções é utilizada a Teoria da Otimalidade de Pareto [20].

Esses problemas têm sido resolvidos através de diversos de Algoritmos Evolucionários Multiobjetivo (MOEAs, do inglês *Multi-Objective Evolutionary Algorithms*) [20]. MOEAs incorporam um mecanismo de seleção, geralmente baseado nos conceitos da Otimalidade de Pareto, com o objetivo de preservar as melhores soluções encontradas durante a busca e continuar um progresso em direção às soluções ótimas do problema. Além disso, esses algoritmos adotam mecanismos para a preservação da diversidade, para evitar a convergência para uma só solução. Nessa área, diferentes meta-heurísticas são utilizadas com sucesso, como as Abordagens Evolucionárias [41], Otimização por Nuvem de Partículas (PSO, do inglês *Particle Swarm Optimization*) [54], Sistemas Imunológicos Artificiais (AIS, do inglês Artificial Immune Systems) [26].

O PSO é uma meta-heurística populacional utilizada para resolver diversos problemas de otimização. Os algoritmos PSO são projetados para prover soluções robustas e escaláveis. Essa meta-heurística é inspirada no comportamento do voo de pássaros em busca de alimento. Um indivíduo no PSO, chamado de partícula, usa regras simples para controlar suas ações e através de interações com todo o grupo é gerado um comportamento emergente. Essas características permitem o uso do PSO para problemas multiobjetivo [79]. A Otimização por Nuvem de Partículas Multiobjetivo, MOPSO (do inglês *Multi-Objective Particle Swarm Optimization*) busca aplicar conceitos da Otimização Multiobjetivo em algoritmos PSO.

No entanto, a maioria dos MOEAs propostos, inclusive os baseados no PSO, é restrita a problemas com duas ou três funções objetivo. Recentemente, cresceu o interesse na pesquisa em trabalhos que lidam com um número grande de objetivos (maior do que três) devido à limitação dos MOEAs tradicionais quando utilizados neste contexto [46]. Quando o número de objetivos cresce há um aumento no número do conjunto das melhores soluções. Com um maior número de soluções, ocorre a deterioração da busca dos MOEAs, pois esses algoritmos enfrentam dificuldades em identificar e gerar as melhores soluções, o que prejudica o processo de seleção. Logo, se não houver um boa seleção das soluções não há um progresso contínuo em direção às soluções ótimas e não há convergência na busca. Além disso, o número de soluções necessárias para se aproximar do melhor resultado cresce exponencialmente [46] [82]. Assim, novos trabalhos têm sido propostos buscando atenuar a deterioração dos MOEAs para esses problemas. A área que visa estudar novas técnicas para problemas com muitos objetivos é chamada de Otimização com Muitos Objetivos ou *Many-Objective Optimization*. Problemas com mais de três funções objetivo são chamados de Problemas de Otimização com Muitos Objetivos, ou *Many-Objetive Optimization Problems* (MaOPs).

## 1.1 Motivações e metodologia

Este trabalho de doutorado tem como objetivo propor novas estratégias para reduzir as limitações de MOEAs em problemas com muitos objetivos. Além disso, o objetivo é propor novos métodos aplicados à Otimização por Nuvem de Partículas. Este estudo é motivado por:

- A Otimização com Muitos Objetivos é uma área de estudo nova e ainda possui diversos campos de pesquisa em aberto. Nessa área, poucas técnicas foram propostas [1], [27], [47], [81], [82] e ainda não há uma solução definitiva para a deterioração dos MOEAs.

- A Otimização por Nuvem de Partículas é uma meta-heurística que vem sendo aplicada com sucesso em Problemas de Otimização Multiobjetivo [15], [35]. O PSO

possui características importantes que podem ser exploradas para a proposta de novos métodos, como por exemplo, o mecanismo de troca de informação entre as partículas ou a possibilidade de se utilizar diversas populações em uma só execução.

- Poucos trabalhos da literatura utilizam o PSO para problemas com muitos objetivos [59], [85].

Assim, através dessa motivação, são definidos os objetivos deste trabalho:

- Estudar trabalhos da área da Otimização Evolucionária Multiobjetivo (EMO, do inglês *Evolutionary Multi-Objective Optimization*) especialmente os trabalhos que lidam com a Otimização com Muitos Objetivos e com a Otimização por Nuvem de Partículas.

- Aplicar técnicas da Otimização com Muitos Objetivos da literatura em algoritmos PSO e explorar estes novos algoritmos em Problemas de Otimização com Muitos Objetivos. Este passo busca identificar se as técnicas da literatura funcionam quando aplicadas a uma meta-heurística cooperativa.

- Propor novos métodos para Otimização com Muitos Objetivos, especialmente projetados para algoritmos MOPSO.

- Executar um conjunto de estudos empíricos utilizando problemas com muitos objetivos e medidas de desempenho, com ênfase na análise de aspectos como a convergência e a diversidade da busca de MOEAs. Esses estudos têm como objetivo a consolidação do conhecimento adquirido na aplicação do PSO em problemas com muitos objetivos e do funcionamento dos métodos da literatura. Além disso, eles são de fundamental importância para avaliar se as técnicas propostas são úteis para problemas com muitos objetivos.

## 1.2 Contribuições da tese

Definidos os objetivos, as principais contribuições desta tese são:

- Apresentar uma revisão bibliográfica e discussão dos principais trabalhos relacionados da Otimização com Muitos Objetivos e da Otimização por Nuvem de Partículas.
- Propor novos métodos para a Otimização com Muitos Objetivos.
- Propor novos algoritmos MOPSO apropriados para Otimização com Muitos Objetivos.
- Análise empírica para a validação dos métodos e algoritmos propostos.

A primeira contribuição refere-se aos dois próximos capítulos desta tese. Esses capítulos descrevem os principais conceitos da Otimização com Muitos Objetivos e da Otimização por Nuvem de Partículas. Além disso, os principais trabalhos relacionados são discutidos, bem como são definidos os desafios e caminhos tomados no desenvolvimento da tese.

A principal contribuição desta tese é a proposta de novos métodos e algoritmos especialmente direcionados para a Otimização com Muitos Objetivos. O desenvolvimento foi dividido em três partes: novas relações de preferência [16], métodos de arquivamento [9] e múltiplos enxames [36].

As novas relações de preferência compreenderam o uso de novas relações de preferência em algoritmos MOPSO. Essas novas relações têm o objetivo de induzir uma nova ordenação das soluções e com isso evitar a deterioração de MOEAs em Problemas de Otimização com Muitos Objetivos. Nesta etapa se destacam o uso da técnica Controle da Área de Dominância das Soluções [81] em um algoritmo MOPSO e proposta de novos métodos de ranking [23] chamados de *Balanced Ranking* e Combinação de Rankings. Através do uso dessas técnicas em algoritmos MOPSO são desenvolvidos os algoritmos Ranking-SMPSO [25] e CDAS-MOPSO [16].

O estudo dos métodos de arquivamento tem como principal objetivo explorar uma característica importante de algoritmos MOPSO. O arquivamento de soluções é uma tarefa executada por diversos MOEAs [64], porém possui importância na meta-heurística MOPSO. O objetivo é desenvolver novos métodos de arquivamento que permitam diminuir a deterioração da busca de algoritmos MOPSO em MaOPs. Nesse contexto são propostos

os métodos Arquivador Ideal [7], Arquivador Distribuído, Arquivador Distribuído pela Busca [9] e Arquivador Hiperplano.

Por fim, o último conjunto de algoritmos desenvolvidos nesta tese utiliza conceitos de múltiplos enxames. Múltiplos Enxames [36] são utilizados em algoritmos PSO com o objetivo de diminuir a complexidade da busca. Com esse objetivo, são desenvolvidos algoritmos que buscam combinar diferentes métodos de arquivamento em um ambiente com múltiplos enxames. Os algoritmos são chamados de *Multi-Swarm* Baseado em Arquivadores, *Multi-Swarm* com Arquivo Externo Compartilhado e *Iterated Multi-Swarm*, I-Multi e compõem as qualidades de diferentes métodos de arquivamento para melhorar a busca de algoritmos MOPSO.

Para a avaliação de todos os métodos e algoritmos desenvolvidos nesta tese, é apresentado um conjunto de experimentos que efetuam uma análise empírica dos resultados obtidos. Essa análise empírica aplica os algoritmos desenvolvidos em diferentes problemas artificiais com muitas funções objetivo e utiliza diferentes indicadores de qualidade. O principal objetivo é identificar se os métodos propostos realmente reduzem a deterioração da busca e se é possível obter bons resultados em termos de convergência e diversidade em cenários com muitos objetivos.

Este conjunto de algoritmos e as respectivas análises empíricas geraram um conjunto de publicações em revistas e conferências da área em [7], [8], [9], [13], [14], [16], [17], [25], [52] e [53].

## 1.3 Organização da tese

O restante desta tese está organizada seguinte maneira:

**Capítulo 2 - Otimização Multiobjetivo e Otimização com Muitos Objetivos:** Esse capítulo apresenta os principais conceitos da Otimização Multiobjetivo e Otimização com Muitos Objetivos. Nele são discutidos os conceitos básicos para o entendimento do trabalho desenvolvido nesta tese. Além disso, é feita uma revisão dos principais trabalhos relacionados da área com o objetivo de delinear os principais desafios enfrentados.

**Capítulo 3 - Otimização por Nuvem de Partículas Multiobjetivo:** Nesse

capítulo a meta-heurística utilizada no desenvolvimento da tese, a Otimização por Nuvem de Partículas Multiobjetivo, é discutida. A terminologia básica é apresentada, bem com são discutidos os principais conceitos da meta-heurística, como escolha do líder, equações de movimento e métodos de arquivamento. Da mesma forma que no capítulo anterior, é feita uma revisão dos principais trabalhos da literatura que são utilizados como guia no desenvolvimento dos métodos propostos.

**Capítulo 4 - Novas Relações Preferências Aplicadas à meta-heurística MOPSO na Otimização com Muitos Objetivos:** Nesse capítulo os novos métodos e os novos algoritmos baseados em novas relações de preferência são propostos. O uso de novas relações de preferências é um dos principais caminhos adotados para resolver Problemas com Muitos Objetivos na literatura. Esse capítulo apresenta a aplicação de métodos da literatura em algoritmo MOPSO, bem como a proposta de novos métodos. Nesse capítulo são apresentados os algoritmos Ranking-SMPSO, CDAS-MOPSO e os métodos *Balanced Ranking* e a Combinação de Rankings.

**Capítulo 5 - Explorando Métodos de Arquivamento:** Métodos de arquivamento são importantes na meta-heurística MOPSO e diferentes métodos já foram propostos na literatura. Nesse capítulo são propostos quatro novos métodos que buscam reduzir a deterioração de algoritmos MOPSO quando aplicados a MaOPs. São apresentados os métodos Arquivador Ideal, Arquivador Distribuído, Arquivador Distribuído pela Busca e Arquivador Hiperplano.

**Capítulo 6 - Explorando Múltiplos Enxames e Métodos de Arquivamento:** Esse capítulo apresenta os algoritmos *Multi-Swarm* Baseado em Arquivadores, *Multi-Swarm* com Arquivo Externo Compartilhado e *Iterated Multi-Swarm* que exploram a meta-heurística MOPSO com múltiplos enxames. Esses algoritmos têm como principal característica compor diferentes métodos de arquivamento através de múltiplos enxames.

**Capítulo 7 - Avaliação Empírica dos Métodos Propostos:** Nesse capítulo são apresentados os conjuntos de experimentos utilizados para validar todos os métodos propostos na tese. São efetuadas análises empíricas com o objetivo de avaliar cada técnica proposta, bem como confrontá-las entre si e com algoritmos da literatura.

**Capítulo 8 - Conclusões:** Por fim, esse capítulo apresenta as considerações finais desta tese e planeja possíveis trabalhos futuros.

## 1.4 Notas sobre terminologia

Esta tese utiliza diversas palavras e expressões em inglês tais como *fitness*, *benchmark*, *swarms*, entre outros, bem como alguns acrônimos derivados do inglês como PSO, MOPSO, EMO, MaOPs. Foi definido deixar esses termos em inglês devido ao seu uso comum na literatura especializada para que não haja uma interpretação equivocada dos termos referenciados.

# CAPÍTULO 2

# OTIMIZAÇÃO MULTIOBJETIVO E OTIMIZAÇÃO COM MUITOS OBJETIVOS

Este capítulo apresenta as definições da Otimização Multiobjetivo e da Otimização com Muitos Objetivos. Além disso, é feita uma análise dos principais trabalhos da área e apresentados os desafios que serão enfrentados nesta tese. O capítulo está organizado da seguinte forma: a Seção 2.1 discute a Otimização Multiobjetivo, bem como apresenta análise de alguns trabalhos relacionados. O tema principal de estudo nesta tese, a Otimização com Muitos Objetivos, é discutido na Seção 2.2. Por fim, as considerações finais desse capítulo são apresentadas na Seção 2.3.

## 2.1 Otimização multiobjetivo

Um Problema de Otimização Multi-Objetivo (MOP, do inglês *Multi-Objective Optimization Problem*) envolve a satisfação simultânea de duas ou mais funções objetivo. Nestes problemas, as funções objetivo que são otimizadas estão em conflito[1], ou seja, se um valor de uma função objetivo melhora, em geral, o valor de outra função objetivo piora. Isso representa que não há somente uma melhor solução, mas um conjunto com as melhores soluções. Para obter esse conjunto de soluções é utilizada a Teoria da Otimalidade de Pareto [20]. O problema multi-objetivo geral, de minimização e sem restrições, é definido a seguir.

Um vetor de variáveis de decisão $\overrightarrow{x}$ é o conjunto de valores numéricos que serão escolhidos pelo problema de otimização (uma solução do problema). O vetor $\overrightarrow{x} \in \Omega$, onde $\Omega$ é o conjunto das soluções possíveis do problema, é denotado por $\overrightarrow{x} = (x_1, x_2, ..., x_n)$,

[1]O termo otimizar corresponde a determinar um conjunto de soluções as quais apresentem os melhores valores possíveis para as funções objetivos ou os valores das funções que sejam aceitáveis para o tomador de decisão. A etapa da tomada de decisão é um dos estágios da otimização multi-objetivo e o tomador de decisão é o responsável por escolher, dentre todas as soluções geradas, uma única solução [20]

onde $n$ é a dimensão do vetor de variáveis de decisão e $x_i$ é o $i$-ésimo valor pertencente a este vetor. Num problema de otimização, a qualidade de uma possível solução é definida através de alguns critérios. Esses critérios são expressos através de funções computáveis das variáveis de decisão, que são chamadas de funções objetivo, denotadas por $f_j(\overrightarrow{x})$, $F : R^n \rightarrow R$.

Em problemas multiobjetivo há várias funções objetivo, assim é utilizado um vetor de funções objetivo $\overrightarrow{f(\overrightarrow{x})} \in \Lambda$, onde $\Lambda$ é o espaço do vetor das funções objetivo para o problema. A otimização multiobjetivo é definida por:

$$Minimize \overrightarrow{f(\overrightarrow{x})} = (f_1(\overrightarrow{x}), f_2(\overrightarrow{x})..., f_m(\overrightarrow{x})) \tag{2.1}$$

onde $m$ é o número de funções objetivo do problema (dimensão de $\Lambda$ ) e $f_j(\overrightarrow{x})$ é a $j$-ésima função objetivo. Para cada solução $\overrightarrow{x}$ no espaço das variáveis de decisão, existe um mapeamento para o vetor $\overrightarrow{f(\overrightarrow{x})}$ no espaço das funções objetivo, $F : R^n \rightarrow \Lambda$.

A `Otimização Multiobjetivo` [20] tem como propósito otimizar as $m$ funções objetivo simultaneamente com o objetivo de encontrar um conjunto de soluções que representam um melhor compromisso entre os objetivos. Assim, nesse conjunto de soluções, para cada solução $\overrightarrow{x}$ não é possível diminuir o valor de uma função objetivo sem que haja o aumento de pelo menos outra função. Em problemas multiobjetivo o ótimo é definido através de termos como Ótimo de Pareto, Dominância de Pareto, Conjunto Ótimo de Pareto e Fronteira de Pareto [20].

**Definição 1, Dominância de Pareto**: Dados $\overrightarrow{f(\overrightarrow{x})} = (f_1(\overrightarrow{x}), f_2(\overrightarrow{x})..., f_m(\overrightarrow{x}))$ e $\overrightarrow{f(\overrightarrow{y})} = (f_1(\overrightarrow{y}), f_2(\overrightarrow{y})..., f_m(\overrightarrow{y}))$, $\overrightarrow{f(\overrightarrow{x})}$ ***domina*** $\overrightarrow{f(\overrightarrow{y})}$ , denotado por $\overrightarrow{f(\overrightarrow{x})} \preceq \overrightarrow{f(\overrightarrow{y})}$, se e somente se (minimização):

$$\forall i \in \{1, 2, ..., m\} : f_i(\overrightarrow{x}) \leq f_i(\overrightarrow{y}), \text{ e } \exists i \in \{1, 2, ..., m\} : f_i(\overrightarrow{x}) < f_i(\overrightarrow{y})$$

$\overrightarrow{f(\overrightarrow{x})}$ é dito não dominado se não existe nenhum $\overrightarrow{f(\overrightarrow{y})}$ que domine $\overrightarrow{f(\overrightarrow{x})}$.

**Definição 2, Ótimo de Pareto**: Dada a solução $\overrightarrow{x} \in \Omega$, com vetor de funções objetivo $\overrightarrow{f(\overrightarrow{x})} = (f_1(\overrightarrow{x}), f_2(\overrightarrow{x})..., f_m(\overrightarrow{x}))$, $\overrightarrow{x}$ é dito ***Pareto Ótimo*** se e somente se não

existe $\overrightarrow{y} \in \Omega$, com vetor de funções objetivo $\overrightarrow{f(\overrightarrow{y})} = (f_1(\overrightarrow{y}), f_2(\overrightarrow{y})..., f_m(\overrightarrow{y}))$, tal que $\overrightarrow{f(\overrightarrow{y})} \preceq \overrightarrow{f(\overrightarrow{x})}$. Ou seja, $\overrightarrow{x}$ é dito Pareto Ótimo se o seu vetor de funções objetivo é não dominado. Uma solução que é Ótimo de Pareto possui um vetor de funções objetivos que não pode ser simultaneamente melhorado.

**Definição 3, Conjunto Pareto Ótimo**: Para um MOP, o Conjunto Pareto Ótimo, $P^*$, é o conjunto das melhores soluções em $\Omega$, ou seja, é o conjunto das soluções Pareto Ótimo do problema. $P^*$ é definido como:

$$P^* := \{\overrightarrow{x} \in \Omega \mid \nexists \overrightarrow{y} \in \Omega \ , \ \overrightarrow{f(\overrightarrow{y})} \preceq \overrightarrow{f(\overrightarrow{x})}\} \tag{2.2}$$

**Definição 4, Fronteira de Pareto**: Cada solução em $P^*$ possui uma imagem de um ponto não dominado em $\Lambda$. O conjunto de todos os pontos não dominados no espaço das funções objetivo é chamado de Fronteira de Pareto (PF, do inglês *Pareto Front*), definido por:

$$PF^* := \{\overrightarrow{f(\overrightarrow{x})} \mid \overrightarrow{x} \in P^*\} \tag{2.3}$$

**Definição 5, Vetor Ideal**: é o vetor que contém os melhores valores possíveis para cada função objetivo. O vetor ideal é denotado por:

$$\overrightarrow{f^0(\overrightarrow{x})} = [f_1^0(\overrightarrow{x}), f_2^0(\overrightarrow{x}), ..., f_m^0(\overrightarrow{x})] \tag{2.4}$$

onde $f_i^0(\overrightarrow{x})$ é o valor ótimo da função objetivo $f_i(\overrightarrow{x})$. Em geral $\overrightarrow{f^0(\overrightarrow{x})}$ é um vetor utópico, geralmente utilizado para definir a melhor posição possível no espaço de objetivos. Porém, em geral, essa posição não pode ser alcançada.

**Definição 6, Vetores Extremos**: é o conjunto de vetores que contêm o valor ótimo para $m-1$ funções objetivo e o valor máximo para a $j$-ésima função ($f_j^{max}(\overrightarrow{x})$). Num problema com $m$ funções objetivo existem $m$ vetores extremos. O vetor extremo $\overrightarrow{E_i}$ é denotado por:

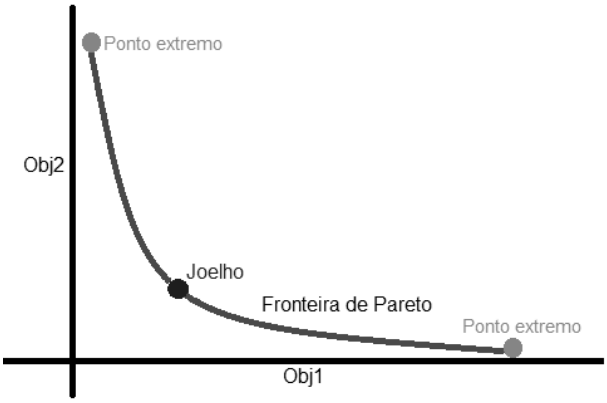



Figura 2.1: Exemplo do joelho e vetores extremos num espaço com duas funções objetivo.

$$\overrightarrow{E_j} = [f_1^0(\overrightarrow{x}), f_2^0(\overrightarrow{x}), ..., f_j^{max}(\overrightarrow{x}), ..., f_m^0(\overrightarrow{x})] \tag{2.5}$$

onde $f_i^0(\overrightarrow{x})$ é o valor ótimo da função objetivo $f_i(\overrightarrow{x})$ e $f_j^{max}(\overrightarrow{x})$ é o valor máximo para a função $f_j(\overrightarrow{x})$. O conjunto com os $m$ vetores $\overrightarrow{E_j}$ normalmente é usado para representar um ponto em cada eixo de objetivos.

**Definição 7, Joelho da fronteira de Pareto**: De forma simples, o joelho da fronteira de Pareto pode ser definido como um ponto de maior convexividade (uma protuberância) da curva [24]. Uma discussão detalhada, apresentando os conceitos teóricos na definição do joelho pode ser encontrada em [24]. Nesta tese, o conceito do joelho será explorado somente em curvas que definem fronteiras de Pareto simples. A Figura 2.1 mostra o exemplo do joelho para uma curva de uma fronteira de Pareto de minimização. O joelho é o ponto de maior arqueamento da curva. É comum que o joelho esteja localizado no meio da curva, o que representa o melhor compromisso entre os valores das funções objetivo. Nos trabalhos [24] e [47] é discutido que tomadores de decisão em geral preferem pontos próximos ao joelho da fronteira.

Problemas de Otimização Multiobjetivo são solucionados por diferentes áreas de pesquisa. Em geral, MOP são problemas da classe NP-Completo [20]. Neste contexto, a grande complexidade de MOPs impossibilita o uso de algoritmos de busca tradicionais. Assim, para esse conjunto de problemas, algoritmos bioinspirados tais como os Algorit-

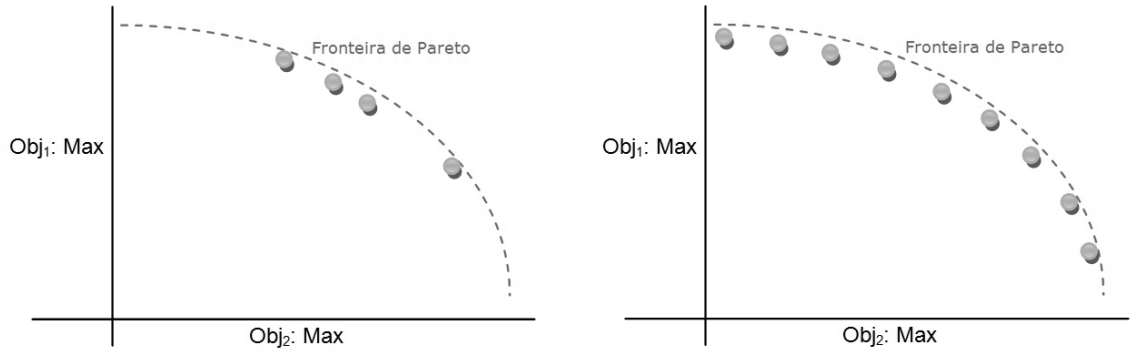
mos Evolucionários, entre eles os Algoritmos Genéticos [41], Programação Evolucionária, bem como outras meta-heurísticas, como a Otimização por Nuvem de Partículas [79], Sistemas Imunológicos Artificiais [26], entre outros, têm sido aplicados com sucesso. Em geral, este conjunto de algoritmos utiliza uma população de soluções na busca e assim permite a geração de diversos elementos da Fronteira de Pareto em uma só execução. A área da Otimização Evolucionária Multiobjetivo visa a aplicação de Algoritmos Evolucionários Multiobjetivo (MOEA, do inglês *Multi-Objective Evolutionary Optimization Algorithms*)[2] para a solução de MOPs.

Os Algoritmos Evolucionários Multiobjetivo modificam os algoritmos evolucionários de duas maneiras: incorporam um mecanismo de seleção, geralmente baseado nos conceitos da Otimalidade de Pareto, e adotam mecanismos para a preservação da diversidade, para evitar a convergência para uma só solução. Como a maioria dos MOPs são problemas complexos, os MOEAs focam em determinar um conjunto de soluções mais próximo possível do Conjunto Ótimo de Pareto, denominado de conjunto de aproximação. Por clareza, a imagem deste conjunto de aproximação no espaços das funções objetivo é chamada de $PF_{aprox}$ e a fronteira de Pareto do problema é chamada de $PF_{real}$.

Através destes conceitos pode-se definir que o principal objetivos de um MOEA é gerar um bom conjunto de soluções para o tomador de decisão, para a seleção do resultado no espaço de variáveis. Com este intuito, um MOEA utiliza os seguintes meios: (1) preservar os pontos não dominados e a sua associação no espaço de busca (soluções Pareto ótimo); (2) manter o progresso do algoritmo na direção da fronteira de Pareto; (3) manter a diversidade dos pontos no espaço de objetivos e de variáveis (obter uma boa cobertura da fronteira de Pareto).

Em resumo, um MOEA deve convergir para as melhores soluções do problema, porém deve cobrir de melhor forma as diferentes regiões para possibilitar a geração de melhor conjunto para o tomador de decisão.

[2]Na literatura, em geral, há uma diferenciação entre Algoritmos Evolucionários (baseados na teoria da evolução das espécies) e as demais meta-heurística bio-inspiradas (qualquer meta-heurística que é inspirada num comportamento biológico, tais como o comportamento formigas, pássaros, abelhas e mesmo o comportamento evolutivo de espécies. Porém, na literatura multiobjetivo, todos os algoritmos bio-inspirados que são utilizados para otimizar mais de um objetivo são classificados como MOEAs e essa será a notação utilizada nesta tese).




(a) Boa convergência (b) Boa diversidade

Figura 2.2: Representações de conjuntos de aproximação com boa convergência e boa diversidade. Os pontos indicam as fronteiras aproximadas.

### 2.1.1 Convergência versus diversidade

Em [20], a convergência de $PF_{aprox}$ para $PF_{real}$ é definida, de forma simples, como um movimento crescente de aproximação do conjunto de soluções corrente até $PF_{real}$, durante as iterações do MOEAs (considerando o espaço das funções objetivo). Convergência da busca de um algoritmo em relação à fronteira de Pareto é observada se a fronteira aproximada gerada está localizada perto da Fronteira de Pareto. Em contrapartida, aproximação de $PF_{aprox}$ não é a única qualidade desejada na busca de um MOEAs. Além disso, é interessante que o conjunto de soluções gerado esteja espalhado por diferentes regiões do espaço de busca. A diversidade em relação à fronteira de Pareto é observada se as soluções estão distribuídas por toda a região que a fronteira de Pareto ocupa[3].

A Figura 2.2(a) mostra um exemplo de um conjunto de aproximação com boa convergência, pois as soluções estão próximas à fronteira de Pareto. Porém, nesse exemplo não há uma boa diversidade, pois as soluções estão concentradas em alguns pontos e não estão bem espaçadas. A Figura 2.2(b) mostra um conjunto de aproximação com a boa diversidade, espalhado pela região da fronteira de Pareto e bem espaçado, mas sem convergência, longe da fronteira.

Esses conceitos serão explorados em diversos pontos desta tese. O objetivo deste

[3]A discussão sobre convergência e diversidade em geral envolve a prova de teoremas que discutem de maneira formal esses aspectos de algoritmos multiobjetivo. Porém, o objetivo desta seção não é discutir esses teoremas, mas sim apresentar de maneira simples e prática os aspectos de convergência e diversidade que são desejáveis em Algoritmos Evolucionários Multiobjetivo. A discussão desses teoremas pode ser encontrada no Capítulo 6 em [20].

trabalho é propor novos métodos para Otimização Multiobjetivo que apresentem bons resultados em termos de convergência e diversidade em relação à fronteira de Pareto. Os métodos propostos têm como objetivo melhorar o resultado de um MOEA (nesta tese são explorados os algoritmos baseados na Otimização por Nuvem de Partículas) com o objetivo que a $PF_{aprox}$ se aproxime mais da fronteira de Pareto real e que esse conjunto também seja bem diversificado.

### 2.1.2 Relação de preferência para algoritmos evolucionários multiobjetivo

MOEAs buscam solucionar problemas de otimização multiobjetivo através de diferentes formas. Nesses algoritmos, uma função de *fitness* é utilizada para avaliar as funções objetivo e determinar quais são as melhores soluções da população em cada iteração do algoritmo. Essa função de *fitness* define uma relação de preferência entre as soluções, utilizada para identificar as regiões do espaço com melhor potencial [47]. Uma relação de preferência é um mecanismo para decidir se uma solução $\overrightarrow{x}$ é preferível a uma solução $\overrightarrow{y}$.

Algoritmos Multiobjetivo utilizam diferentes funções de *fitness* como relações e preferência. Em geral, três abordagens de função de *fitness* se destacam: `lexicográfica`, `agregação linear de funções` e a `abordagens baseadas em dominância de Pareto` [39].

A abordagem mais simples é a lexicográfica. Nesse método são definidas prioridades para os objetivos e cada objetivo é avaliado de acordo com esta prioridade. Assim, na comparação entre duas soluções, primeiro é feita a comparação com o objetivo com maior prioridade, caso não haja uma diferença significante para este valor é feita a análise com o próximo objetivo, e assim por diante. Há também a possibilidade de se escolher a função objetivo de maior prioridade de forma aleatória em cada iteração do algoritmo.

A vantagem desse método é que os objetivos são analisados separadamente, não há uma mistura destes valores durante a comparação das soluções e é um método de baixa complexidade computacional. O problema é que ela introduz novos parâmetros complexos para o problema: a definição da ordem das soluções e a definição de um valor limite que

defina se uma solução é melhor que a outra.

Uma outra forma de se trabalhar com problemas multiobjetivo é através da agregação linear de funções [66]. Nessa abordagem o problema multiobjetivo é convertido em um problema escalar através de uma soma ponderada, onde é atribuído uma ponderação para cada função objetivo:

$$min \sum_{j}^{m} w_j f_j(\overrightarrow{x}) \tag{2.6}$$

onde $w_i \geq 0$ e $j = 0, ..., m$ são os pesos de cada função objetivo $j$. Em geral, esses pesos são normalizados e a soma de todos os pesos é igual a 1. Outra abordagem de ponderação é de Tchebycheff [66]. O método da soma ponderada considera uma combinação dos diferentes objetivos através da atribuição de pesos para cada objetivo (vetor $\overrightarrow{\lambda}$) e a soma destes valores. A abordagem de Tchebycheff utiliza a seguinte equação para agregar os objetivos:

$$max_{1 \leq j \leq m}\{\lambda_i | \overrightarrow{f_j(\overrightarrow{x})} - \overrightarrow{z_j}|\} \tag{2.7}$$

onde $m$ é o número de objetivos, $\overrightarrow{\lambda}$ é o vetor de pesos, $\overrightarrow{f_j(\overrightarrow{x})}$ é o valor do $j$-ésimo objetivo e $\overrightarrow{z_j}$ é um ponto de referência que em geral é o vetor ideal.

Agregação de funções tem como vantagem ser simples e também de baixa complexidade. Além disso, como mostrado em [20], para qualquer conjunto positivo de pesos para uma determinada função objetivo, pode ser encontrada uma solução Pareto Ótimo. Porém, esta técnica possui alguns problemas. O primeiro é a necessidade de se encontrar os melhores valores possíveis para o peso de forma empírica. A segunda é o fato de o modelo criado ficar especializado somente para os valores definidos pelos pesos. Por fim, se a fronteira de Pareto for não convexa, o ótimo nessa porção da fronteira não pode ser achado através deste método, ou seja, algumas soluções da fronteira de Pareto não podem ser encontradas através deste método.

Apesar do uso de abordagens que definem a função de *fitness* através da definição de

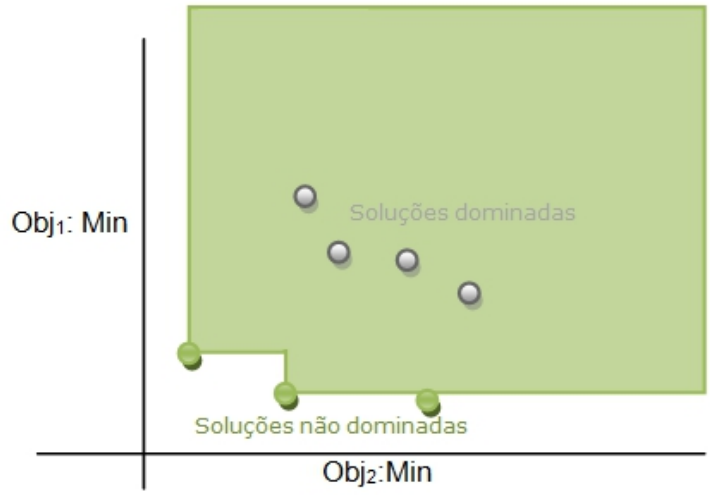



Figura 2.3: Relação da dominância de Pareto para um espaço bi-objetivo

prioridades ou da agregação de funções, o objetivo da otimização multiobjetivo é encontrar os elementos do conjunto Pareto Ótimo. Assim, a maneira mais utilizada por MOEAs para se definir uma função de *fitness* é incorporar diretamente o conceito da dominância de Pareto como mecanismo de seleção. Assim, uma solução é escolhida como melhor, se ela domina as demais soluções da população.

A Figura 2.3 mostra um exemplo da seleção das melhores soluções através da dominância de Pareto. Nessa figura é apresentado um espaço com duas funções objetivos, para um problema de minimização. Cada ponto representa um vetor objetivo de uma solução. Pela definição de dominância de Pareto, as soluções marcadas em verde dominam as soluções marcadas em cinza.

Dentre os diversos Algoritmos Evolucionários Multiobjetivo descritos na literatura, os algoritmos evolucionários se destacam [41]. Essas abordagens evolucionárias são baseadas em conceitos da evolução das espécies e utilizam mecanismos como elitismo, seleção natural, combinação de indivíduos e mutação. A ideia é que em uma população em que os indivíduos competem entre si, as melhores soluções são selecionadas para a procriação e com esse processo há a convergência para as melhores soluções do problema.

Um dos principais algoritmos do estado da arte é o *Non-Dominated Sorting Genetic Algorithm* II, NSGA-II [28]. Esse algoritmo tem a característica de inserir o elitismo na

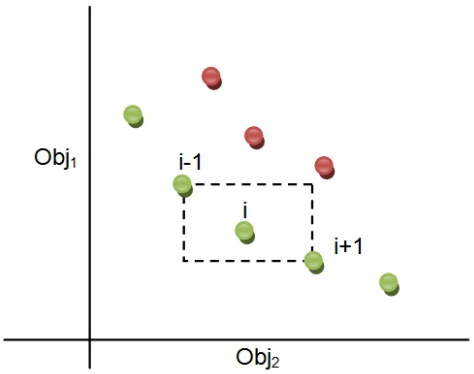



Figura 2.4: Exemplo do cálculo da distância de *Crowding*

busca através de um procedimento que ordena as soluções não dominadas e mantém a diversidade através da proposta de um estimador de densidade, a distância de *crowding*. A distância de *Crowding* (CD, do inglês *Crowding Distance*) é um medida utilizada para estimar a densidade de soluções em volta de uma solução em particular. Essa densidade é calculada através da distância média em relação aos pontos mais próximos em cada eixo de objetivo. Através dessas distâncias médias é definido um cuboide que define o perímetro entre as soluções mais próximas. A Figura 2.4 mostra um exemplo da representação desse cuboide. Para o cálculo da distância de *Crowding*, primeiro as soluções são ordenadas, por cada objetivo. Para cada objetivo, para as soluções nos extremos da ordenação é atribuído um valor infinito. Para as demais soluções é calculada a distância entre elas por cada objetivo. O valor da medida CD é obtido somando os valores das distâncias obtidas. Assim, quanto maior o valor do CD de uma solução, menos densa será a área em que esta solução está localizada.

O método proposto pelo NSGA-II ordena as soluções de acordo com a dominância de Pareto. Uma descrição do algoritmo NSGA-II é apresentada a seguir. Inicialmente, são obtidas as soluções não dominadas e elas são assinaladas ao primeiro *front*. Em seguida, essas soluções são retiradas da população e o processo é repetido para as soluções restantes. As novas soluções não dominadas são assinaladas ao *front* seguinte e o processo é repetido até que não sobrem mais soluções na população. A distância de *crowding* é calculada separadamente para as soluções em cada *front*. A população de filhos é obtida através de processos de seleção (torneio binário), recombinação e mutação. Após

a geração dos filhos, as duas populações são agrupadas e as melhores soluções passam para a próxima iteração. Para a seleção através de torneio binário, duas soluções são escolhidas aleatoriamente. Em seguida, é observado qual é a melhor solução através do seguinte critério: se as soluções estiverem em fronts *diferentes*, a melhor é a que estiver no melhor *front*; se elas estão no mesmo *front*, a melhor é a que possuir o melhor valor de CD. No artigo original, o NSGA-II é comparado ao PAES [57], um algoritmo que também tem como base o elitismo e que também tem o objetivo explícito de preservar a diversidade do *front* de soluções não dominadas. Os algoritmos são avaliados através de 5 problemas de teste, analisam a uniformidade da distribuição das soluções, através da medida *Spread*. Neste conjunto de testes, o NSGA-II superou o PAES. O NSGA-II tem baixa complexidade computacional, quando comparado a outros MOEAs da literatura e foi utilizado em diversos outros trabalhos obtendo bons resultados. Esse algoritmo é comumente utilizado como base de comparação para a proposição de novos algoritmos evolucionários multiobjetivo.

Outras metaheurísticas são utilizadas no desenvolvimento de algoritmos multiobjetivo. A Otimização por Nuvem de Partículas é o tema de estudo nesta tese e será explorada no Capítulo 3. Outra alternativa são os sistema artificiais imunológicos(AIS, do inglês *Artificial Immune Systems*) [26]. Esta metaheurística é baseada no modelo do sistema imunológico dos seres humanos. O sistema imunológico tem a capacidade de identificar os organismos estranhos, chamados de antígenos, e atacá-los através dos anticorpos, que possuem o principal papel de defesa do corpo. Quando um antígeno é identificado os anticorpos que melhor reconhecem esse antígeno se proliferam através de um processo de clonagem. Este princípio é chamado da seleção clonal [26]. Na literatura existem alguns trabalhos que utilizam a metaheurística dos sistemas artificiais imunológicos para resolver problemas de otimização.

Em [19] é apresentado um dos primeiros trabalhos que aplicam essa metaheurística a problemas multiobjetivo, *Multiobjective Immunologic System Algorithm*, MISA. O MISA é baseado no princípio da seleção clonal, onde somente os anticorpos com maior afinidade serão clonados. Os anticorpos representam uma solução para o problema. Não

são utilizados antígenos, a afinidade dos anticorpos é definida através de conceitos da dominância de Pareto e uma medida da diversidade das soluções, definida através de um *grid* adaptável [58]. No algoritmo, as melhores soluções devem ser clonadas, estas soluções são as soluções não dominadas ou as soluções dominadas por menos soluções. O número de clones é definido através da diversidade das soluções. As soluções localizadas em regiões mais densas no grid tem um número menor de clones, enquanto as soluções em regiões menos densas tem um número maior de clones. Após o processo de clonagem, os clones gerados sofrem um processo de mutação. Após a mutação, os melhores indivíduos são selecionados entre todas as soluções e o processo é repetido até uma condição de parada. O algoritmo MISA é avaliado em diferentes problemas de *benchmarking* e seus resultados são comparados aos algoritmos microGA, NSGA-II e PAES. Para a comparação dos algoritmos são utilizados alguns indicadores de qualidade como a *Error Ratio* e o *Spacing* [84]. Além dessas medidas, esse artigo propõe a medida *Inverted Generational Distance*, que é a inversão da *Generational Distance* [84]. Com essa medida é possível medir convergência e diversidade. O artigo concluiu que o MISA apresenta resultados competitivos em relação a algoritmos do estado da arte da literatura. Além disso, o algoritmo é uma primeira aplicação de AIS em problemas multiobjetivo e que a técnica da seleção clonal pode ser efetiva para resolver esses problemas de forma simples.

A maioria dos trabalhos que lidam com multiobjetivo somente utilizam a dominância de Pareto como relação de preferência entre as soluções. Porém, como discutido anteriormente, existem diferentes abordagens para trabalhar com multiobjetivo. Entre elas, as que agregam o conjunto das funções ganham destaque. Através deste método, é possível obter uma solução Pareto ótimo em que o seu objetivo é uma agregação de $m$ funções objetivo. Assim, a aproximação da fronteira de Pareto para um problema multiobjetivo é obtida através da decomposição do problema em um número de problemas escalares. Nesses métodos, cada solução é obtida através da variação dos pesos das funções de agregação. Dentre estes métodos o algoritmo MOEA/D [86] se destaca, pois apresenta resultados competitivos com algoritmos multiobjetivo do estado da arte, como o NSGA-II [28], inclusive superando esses algoritmos para problemas com muitos objetivos.

O MOEA/D possui a característica de introduzir a decomposição em problemas na computação evolucionária multiobjetivo. Como é um algoritmo que visa resolver $N$ problemas escalares apresenta mais facilidade para lidar com problemas como a manutenção da diversidade e possui um custo computacional menor em relação aos algoritmos multiobjetivo do estado da arte. No MOEA/D é dado um conjunto de vetores de peso $\overrightarrow{\lambda^1}$, ...,$\overrightarrow{\lambda^N}$, onde $\overrightarrow{\lambda^i} = (\lambda_1^i, ..., \lambda_m^i)$, $m$ é o número de objetivos e $N$ o número de soluções. A otimização de um problema escalar, utilizando a agregação de funções com um conjunto de pesos $\overrightarrow{\lambda^i}$ irá gerar uma solução do conjunto Pareto ótimo. Assim, uma solução $\overrightarrow{x_i}$ é gerada através de um método de decomposição (soma ponderada, abordagem de Tchebycheff) e um vetor $\overrightarrow{\lambda^i}$. O MOEA/D baseia-se no fato de cada vetor de pesos $\overrightarrow{\lambda^i}$ possuir um vizinhança com $T$ vetores de pesos. Essa vizinhança é definida por um conjunto de vetores semelhantes, que são vetores próximos no espaço de pesos. O artigo argumenta que vetores de pesos semelhantes irão gerar soluções ótimas semelhantes. Então, o MOEA/D utiliza um algoritmo evolucionário, ou seja, uso de operadores genéticos (recombinação de soluções, mutação, etc.), para otimizar o conjunto de pesos vizinhos, com o intuito de gerar um conjunto de soluções $\overrightarrow{x_i}$ que cubra toda a fronteira de Pareto. O MOEA/D é comparado com o MOGLS [51] (um algoritmo de decomposição que apresenta bons resultados quando comparados a outros MOEAs da literatura), e ao NSGA-II. Foi utilizado o problema da mochila multiobjetivo e utilizadas as métricas de cobertura (métrica C) e a distância a um representante na fronteira de Pareto (métrica D). A análise mostrou que o MOEA/D apresenta menor custo computacional e superou os demais algoritmos na comparação das métricas.

## 2.2 Otimização com muitos objetivos

Apesar da aplicação com sucesso de diversos MOEAs, a maioria dos estudos em problemas multiobjetivo foca em problemas com um número pequeno de objetivos. No entanto, problemas práticos envolvem um número grande de critérios, em especial problemas da engenharia, tais como problemas da engenharia automotiva e aeroespacial [38]. Problemas de Otimização Multiobjetivo com mais de 3 funções objetivos são referidos como

Problemas com Muitos Objetivos e a área que tem como objetivo estudar novas soluções para esses problemas é chamada de Otimização com Muitos Objetivos [46].

Diversos trabalhos na literatura mostram como MOEAs tradicionais escalam de maneira pobre quando aplicados a problemas com muitos objetivos [46] [47] [77]. Logo, os esforços de pesquisa têm sido orientados a investigar a escalabilidade desses algoritmos com respeito ao número de funções objetivos [46]. Neste contexto as principais dificuldades enfrentadas pelos MOEAs em problemas com muitos objetivos são:

- **Deterioração da habilidade de busca:** quando o número de objetivos cresce o número de soluções não dominadas também cresce [46]. Como consequência, a habilidade de busca de algoritmos que usam a dominância de Pareto como função de seleção é deteriorada. Neste caso, não é possível impor uma preferência na seleção das soluções e não há pressão em direção à fronteira de Pareto. Além disso, quanto maior é o número de funções objetivo, maior o número de soluções locais e mais difícil torna-se o problema, pois há uma maior chance da busca ficar presa em regiões de ótimos locais [82]. Em geral, a busca é guiada praticamente de forma aleatória ou guiada por aspectos de diversidade.

- **Dimensão da fronteira de Pareto:** outro problema é número de soluções para aproximar a fronteira de Pareto. Como discutido em [82], dado um problema com duas funções objetivo, se um conjunto de aproximação com 100 soluções for suficiente para representar o conjunto Pareto ótimo este número cresce de maneira excessiva quando o número de funções também cresce. Se o número de objetivos for definido como 15 o número de soluções necessárias será de $10^{28}$. Nessa situação há o desafio em definir uma estrutura de dados para trabalhar com esse número de soluções e definir mecanismos de diversidade para permitir uma distribuição equivalente dessas soluções.

- **Visualização da fronteira de Pareto:** A visualização dos pontos na fronteira de Pareto é importante para a tomada de decisão [46]. Porém, com mais de três dimensões não é possível plotar os pontos da fronteira de Pareto com alguma forma

convencional.

Em [82], Schutze *et al.* destacam que os desafios que devem ser dominados pelos MOEAs em problemas com muitos objetivos são: a habilidade de se manter um bom conjunto de soluções, com o objetivo de direcionar a população em direção ao conjunto de interesse; aumentar a probabilidade de se melhorar um individuo da população. Neste sentido, os MOEAs devem ter um mecanismos de identificar boas soluções, melhorar a população e convergir para fronteira de Pareto, porém este conjunto de soluções deve ser bom o suficiente para se obter uma boa cobertura da fronteira de Pareto.

A Figura 2.5 mostra um exemplo da deterioração da busca de um algoritmo evolucionário multiobjetivo. O algoritmo utilizado é o SMPSO [70] para o problema DTLZ2 [31], com 2, 5, 10 e 20 número de objetivos. Uma descrição do algoritmo é apresentada na Seção 3.3.1 desta tese. O algoritmo foi executado um total de 1000 gerações e para cada iteração é calculada a menor distância Euclidiana de um elemento da $PF_{atual}$ (pontos não dominados encontrados até então) para algum elemento da $PF_{real}$. Quanto menor a distância, mais próximo $PF_{atual}$ está de $PF_{real}$. O gráfico apresenta o valor da menor distância em relação ao crescimento do número de iterações, assim é possível observar se há convergência na busca com a execução do algoritmo.

Para um número pequeno de funções objetivo, pode ser observado que o algoritmo possui um poder de convergência em direção à $PF_{real}$, já que a distância diminui (tende a zero) quando o número de iterações aumenta. No entanto, quando o número de objetivos aumenta, o algoritmo SMPSO não consegue definir uma ordem entre as soluções geradas e não converge para a $PF_{real}$. Para 10 e 20 funções objetivo, o algoritmo varia a sua busca para pontos que estão próximos e que estão longe da fronteira.

Para a atenuação desses problemas, recentemente várias novas propostas têm sido apresentadas na literatura [45] e elas podem ser categorizadas em:

- **Adaptação das relações de preferência:** essa abordagem visa aumentar a pressão em direção da fronteira de Pareto através de novas relações de preferência que induzem uma nova ordenação do espaço de objetivos [49]. Neste sentido, vários

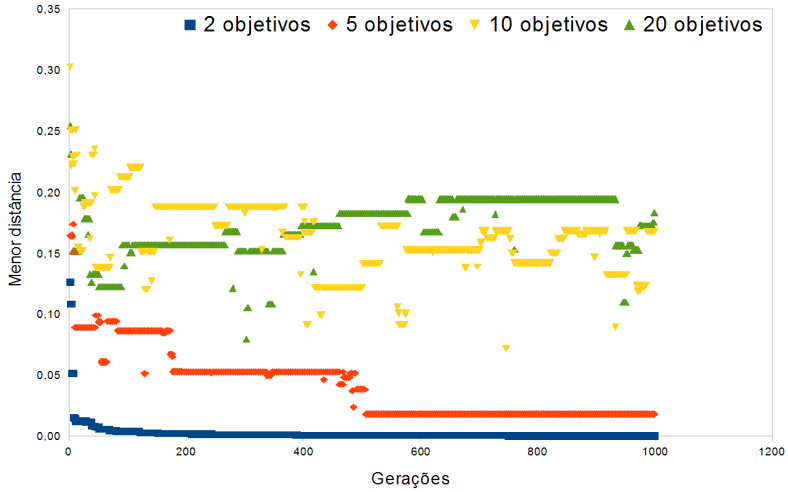



Figura 2.5: Deterioração da habilidade de busca do algoritmo SMPSO para muitos objetivos, problema DTLZ2.

trabalhos se destacam, desde métodos de controle da área de dominância [81] ao uso de esquemas de ranking [23]. Esses métodos serão discutidos mais adiante neste capítulo;

- **Redução da dimensionalidade:** é uma abordagem alternativa para trabalhar com problemas com muitos objetivos [63], [10], [48], [11]. A ideia geral é identificar os objetivos menos conflitantes (os quais possam ser removidos sem que haja uma alteração no conjunto Ótimo de Pareto) e descartá-los. Por exemplo, são identificadas as funções objetivos altamente correlacionadas e somente uma é deixada no vetor de objetivos.

- **Estratégias de decomposição:** essa abordagem usa métodos de decomposição, os quais têm sido estudados na comunidade da programação matemática [86] aplicada no contexto do EMO. Esse conjunto de métodos decompõe o problema multiobjetivo em vários problemas escalares, e então, um algoritmo evolucionário é aplicado para otimizar esses subproblemas [86], [2]. Um algoritmo desta abordagem é o MOEA/D, discutido anteriormente.

- **Incorporação de preferências:** esse conjunto de métodos usa informação de preferências, normalmente definidas pelo tomador de decisão, para executar um algoritmo para muitos objetivos que se concentra em determinadas regiões da fronteira

de Pareto [77] [30].

- **Diferentes esquemas de avaliação de *fitness*:** Essa abordagem não utiliza a dominância de Pareto como função de *fitness*, mas sim um conjunto diferente de medidas de avaliação. Esses algoritmos utilizam medidas desempenho do conjunto de soluções, como por exemplo um conjunto de indicadores, tais como o hipervolume, epsilon e R2 [44], [89].

A seguir são discutidos diversos trabalhos das diferentes categorias descritas acima que buscam resolver problemas com muitos objetivos. Em [46] é apresentado um *survey* que discute os principais problemas que os MOEAs enfrentam quando trabalham com problemas com muitos objetivos, anteriormente discutidos neste capítulo. Além disso, são apresentadas algumas das principais abordagens propostas na literatura. Dentre estas técnicas destacam-se o Controle da Área de Dominância das Soluções [81], o uso de métodos de ranking [23], os algoritmos evolucionários baseados em indicadores (IBEAs) [89] e a redução do número de objetivos [65]. Além da descrição dos principais métodos, o trabalho indica um conjunto de funções de *benchmark* que são utilizadas para avaliação dos novos algoritmos propostos.

Dentre os trabalhos que lidam com muitos objetivos destaca-se o controle da área de dominância das soluções [81] (CDAS). Esta técnica apresenta bons resultados em alguns trabalhos da literatura, sendo aplicada em algoritmos genéticos em [81], [47] e no PSO em [25], [14]. O CDAS foi inicialmente utilizado em algoritmos PSO através de trabalhos do autor. O uso do CDAS no PSO será explorado nos próximos capítulos dessa tese. Este método induz um rankeamento apropriado para as soluções em problemas com muitos objetivos. O método controla o nível de contração e expansão da área de dominância e soluções que originalmente são não dominadas tornam-se dominadas e vice-versa, através de um parâmetro $S$ definido pelo usuário. Sato *et al.* aplicaram o método no NSGA-II [28] e conseguiram obter bons resultados utilizando problemas com muitos objetivos. Uma explicação mais detalhada do método é apresentada no Capítulo 4.

Apesar de apresentar bons resultados em trabalhos da literatura, a técnica CDAS apresenta dois problemas: a influência do parâmetro $S_i$ nos resultados da busca e o fato

que o método privilegia soluções no centro da fronteira de Pareto. Para contornar esses problemas Sato *et al.* (2007) propuseram uma extensão do CDAS, chamada de S-CDAS, *Self-Controlling Dominance Area of Solutions*. Este método controla automaticamente a área de dominância, sem a necessidade da definição do parâmetro $S$. Além disso, O S-CDAS gera um rankeamento das soluções mais granular, levando em consideração que as soluções no extremos de cada função objetivo não são dominadas por nenhuma outro, após o cálculo da nova área de dominância. Deste modo o método possibilita a obtenção de uma busca que melhor balanceie a convergência e a diversidade em direção da fronteira de Pareto. Através de um conjunto de experimentos, utilizando o problema da mochila multiobjetivo, variando entre 4 e 10 funções objetivo, o trabalho mostrou que o S-CDAS obtém melhores resultados em termos de convergência e diversidade, do que algoritmos da litetura, como o NSGA-II [28] e o IBEA [89]. Além disso, o S-CDAS obteve melhores resultados que o CDAS, utilizando valores ótimos de S.

Outra abordagem possível é a geração de rankings para as soluções. Da mesma forma que o CDAS, essa estratégia é estudada em nosso trabalho no Capítulo 4. Em [23] vários métodos de ranking são comparados: *Average Ranking* e *Summed Weighted Ratio* [4], a relação favour [34], $k$-optimality [33] e rankings aleatórios. Todos os rankings foram aplicados a um algoritmo evolucionário simples, onde a seleção é obtida através dos métodos de ranking e é feita somente uma mutação nas soluções. Foi comparada a métrica da cobertura (métrica C) e os métodos foram avaliados nos problemas do caixeiro viajante multiobjetivo e o problema do escalonamento multiobjetivo. Neste trabalho é concluído que o AR, que é um método simples e de baixa complexidade, obtém os melhores resultados e superou métodos recentes, como o *k-optimality*.

Outro método que visa construir um ranking através do conjunto de objetivos de uma solução é a relação *favour* [34]. A relação *favour* compara duas soluções, $s_i$ e $s_j$, e é baseada no número de objetivos em que $s_i$ é melhor que $s_j$. Ou seja, a solução $s_i$ é melhor que $s_j$ pela relação *favour* se, para um problema de maximização:

$$|f_k(s_i) > f_k(s_j)| > |f_k(s_j) > f_k(s_i)|, 1 < k < m$$

onde $m$ é o número de objetivos e $f_k$ é o valor para a função objetivo $k$. No artigo em que a relação é proposta, ela é utilizada em um algoritmo evolucionário, que utiliza operadores genéticos de recombinação e mutação. A relação *favour* é comparada com a relação da dominância de Pareto e com um algoritmo com a agregação de funções (soma ponderada). Os métodos são comparados através de problemas de *benchmarking* e a relação *favour* apresentou os melhores resultados.

Em [47] é feito um estudo comparando diversas relações de preferência, incluindo o controle da área de dominância, alguns métodos de ranking incluindo o AR e o MR, a relação *favour* [34] e a relação da ordem de preferência [75]. Este trabalho utiliza um algoritmo genético simplificado na execução dos experimentos. As diferentes relações de preferências são comparadas através de indicadores de qualidade como o *Generational Distance* [84] e a *Inverted Generational Distance* [19]. Além desses dois indicadores é proposta uma nova abordagem, que mede a distância de cada solução em relação ao "joelho"da busca, isto é, o ponto que fica no centro da curva dos pontos da fronteira de Pareto. É discutido que em muitos casos os tomadores de decisão preferem soluções próximas ao joelho, logo, se um algoritmo gera suas soluções perto do joelho, melhores são os resultados do algoritmo. São utilizados os problemas de *benchmarking* DTLZ2 e DTLZ7 [31]. A conclusão apresentada é que o método que controla a área de dominância apresenta os melhores resultados, porém este método concentra suas soluções em uma pequena região da fronteira de Pareto, em geral, próxima ao joelho. Os métodos de ranking também apresentam bons resultados para problemas com muitos objetivos, porém normalmente o uso destes rankings privilegia a geração de soluções nos extremos da fronteira de Pareto, ou seja, possui soluções com alguns valores de objetivos muito maiores que os demais.

Os métodos de ranking discutidos anteriormente têm a ideia de construir uma função de *fitness* através da associação dos valores dos objetivos. Esse novo *fitness* é utilizado para a definição da ordem das soluções. Em [40] são propostas três relações de preferência que, como nos métodos de ranking, definem um *fitness* para as soluções com muitos objetivos. O primeiro método proposto, chamado de *Global Detriment*, visa medir o quão significante é a diferença entre os valores de cada objetivo. O segundo método, chamado

de *Profit*, mede o ganho de uma solução $s_i$ em relação ao ganho de uma solução $s_j$. O ganho é definido através da maior diferença entre os objetivos de cada solução $s_i$ e $s_j$. Por fim, o terceiro método, chamado de *Distance To The Best Known Solution* (GB), mede a distância Euclidiana entre o vetor objetivo da solução e a solução ideal. As três abordagens são comparadas com métodos da literatura como, a agregação de funções com a soma ponderada dos objetivos, os métodos AR e MR [4], a relação *favour* [34] e a relação de ordem de preferência [75]. Todos os métodos foram aplicados em um algoritmo genético simples, igual ao utilizado em [47]. Os algoritmos são avaliados nos problemas DTLZ1, DTLZ3 e DTLZ6 [31] e é utilizada uma medida de convergência, proposta em [29], que mede a distância mínima entre o conjunto de aproximação e a fronteira de Pareto. Os algoritmos foram aplicados em problemas com objetivos variando entre 5 e 50. Concluí-se que os métodos propostos superam os métodos da literatura e não é possível definir qual método obteve os melhores resultados, sendo as abordagens propostas equivalentes.

Os trabalhos discutidos anteriormente são baseados em propor novas relações de preferência baseadas na modificação da dominância de Pareto ou na geração de rankings a partir do vetor objetivo das soluções. Outra abordagem é a redução da dimensionalidade. Em [65] são apresentadas algumas novas técnicas para se trabalhar com problemas com muitos objetivos. São apresentadas duas contribuições, a primeira é um método para a redução da dimensionalidade que pode ser usado durante a busca de um MOEA. São propostos dois algoritmos para a redução da dimensionalidade: MOSSA e KOSSA. O MOSSA visa obter o número mínimo de objetivos que gere o menor erro. O KOSSA possui uma ideia semelhante, porém encontra um subconjunto de tamanho $k$ que gere o menor erro. Ambos os métodos são baseados na técnica de seleção de atributos não supervisionada. Os dois algoritmos compartilham um mesmo conjunto de passos: agrupar os objetivos homogêneos, quanto maior o conflito entre eles menos homogêneos eles são; selecionar o grupo mais compacto, o qual possui menores distâncias entre os objetivos; reter o centróide do grupo e descartar os demais objetivos. Os métodos propostos são comparados com algoritmos da literatura e apresentam algumas vantagens, como menor complexidade de tempo e valores competitivos nas métricas comparadas. A segunda con-

tribuição é a análise de diferentes relações de preferências desenvolvidas para trabalhar com muitos objetivos. A comparação das relações de preferência é discutida em [47].

Outro trabalho que utiliza a abordagem da redução da dimensionalidade é apresentado em [63]. Nele, é apontado o fato que reduzir o número de objetivos pode induzir um erro na busca que leve o Pareto ótimo gerado através da redução de objetivos ser diferente do Pareto ótimo original. O trabalho tem o objetivo de reduzir o número de objetivos, porém gerar um Pareto ótimo muito próximo ao original. O modelo proposto é testado em uma aplicação industrial mostrando que é possível reduzir o número de objetivos sem uma diferença muito grande no Pareto ótimo gerado. Porém, é enfatizado que não há a definição de um número de objetivos que se deve ser reduzido para que se possa obter uma boa aproximação do Pareto ótimo, sendo necessário para cada problema reduzir diferentes números de objetivos e observar a qualidade do conjunto gerado.

Dentre essas as abordagens que utilizam a decomposição das funções objetivo destaca-se o MOEA/D, discutido anteriormente. Este algoritmo, que é baseado na decomposição da função objetivo em funções escalares, também apresenta resultados promissores quando utilizado em problemas com muitos objetivos.

A quarta categoria listada anteriormente é a incorporação de preferências na busca. As abordagens de incorporação de preferência são úteis pois geram um conjunto de soluções de interesse do tomador de decisão. Em [77], é utilizada uma abordagem de incorporação de preferências diferente. É proposta a ideia de uma família de areas de interesse que são exploradas progressivamente durante a busca. Neste trabalho, as preferências são definidas como vetores alvo no espaço de objetivos. Este conjunto de vetores alvo coevolui simultaneamente com a população de soluções, deste modo as preferências mantêm sua utilidade com a evolução da busca. O algoritmo proposto tem como base um algoritmo evolucionário básico, onde uma população evolui durante um determinado número de gerações. Em cada iteração é feita uma seleção aleatória e a mutação das soluções. A população de preferência (vetores alvo) é utilizada nos métodos de seleção para a definição do *fitness*. Essa população evolui da mesma maneira que a população de soluções. Cada elemento da população de preferência ganha *fitness* se alguma solução é satisfeita por ela,

porém esse *fitness* é reduzido quanto mais soluções satisfazem essa preferência. O trabalho compara o esquema coevolucionário com diferentes esquemas de seleção: esquema de seleção do NSGA-II, seleção através do *Average Ranking* e seleção aleatória. Em todos os casos é utilizado o algoritmo evolucionário básico. Os algoritmos são comparados em diversos cenários com muitos objetivos, comparando-se o hipervolume e feita a análise do ranking de dominância. A principal conclusão do trabalho é que a abordagem coevolucionária apresenta um bom desempenho e pode ser mais explorada em trabalhos da otimização com muitos objetivos, bem como ser utilizada como parâmetro de comparação de novos métodos.

Por fim, a última categoria listada é o uso de diferentes esquemas de *fitness*. Neste contexto, os Algoritmos Evolucionários Baseados em Indicadores, IBEAS (do inglês, *Indicador Bases Evolutionary Algorithms*) [89] obtêm destaque. Esses algoritmos definem medidas binárias de desempenho do conjunto de soluções, chamados de indicadores, e utilizam esses indicadores como objetivo da busca, ao invés do uso da relação dominância de Pareto. É requerido que o indicador binário utilizado seja compatível com a dominância de Pareto, ou seja, se uma solução $s_i$ domina $s_j$ então o valor do indicador é positivo, caso contrário, é negativo. Para a integração dos indicadores a um MOEA, uma função de *fitness* de uma solução é definida. Essa função mede o valor do indicador de cada solução em relação às demais soluções da população, ou seja, mede a perda de qualidade da população se a solução for retirada. A partir da definição desta função de *fitness* é proposto um algoritmo geral, IBEA, que pode ser utilizado com diferentes indicadores da literatura. O IBEA tem a ideia de retirar da população os piores indivíduos da população, deixando somente os indivíduos que maximizem o indicador e em seguida aplicar operadores evolucionários para evoluir a população. O algoritmo proposto é comparado com o NSGA-II e o SPEA2. Os algoritmos são avaliados em diversos problemas de *benchmarking* da literatura, com 2, 3 e 4 objetivos. Eles foram comparados através dos indicadores de qualidade Epsilon e um indicador baseado no Hipervolume [91]. O IBEA apresentou um resultado significativamente melhor na maioria dos problemas testados.

Apesar de uma grande parte dos trabalhos da Otimização com Muitos Objetivos po-

derem ser categorizados através da classificação apresentada anteriormente, há também um conjunto de trabalhos que propõem diferentes abordagens, bem como trabalhos que exploram importantes características dos problemas com muitos objetivos. Em [27] é apresentada uma extensão do algoritmo NSGA-II [28] desenvolvida para problemas com muitos objetivos. O algoritmo, chamado de MO-NSGA-II, utiliza um conjunto de pontos de referência com o intuito de direcionar a busca em direção à fronteira de Pareto sem que haja perda de diversidade na busca. Como no NSGA-II, a população é definida através de *fronts* que são definidos de acordo com a dominância de Pareto. Porém, ao invés da distância de *Crowding* é utilizado um novo estimador de densidade visando identificar regiões do espaço de objetivos e identificar as soluções da população que estão mais próximas dessas regiões. Para isto, é utilizado um conjunto de pontos de referência que são definidos através de um hiperplano cujo vértices são as soluções extremas de cada função objetivo. Além do novo estimador de densidade, o algoritmo proposto utiliza um novo torneio binário. O torneio escolhe duas soluções de forma aleatória e caso haja uma solução que pertence a um *front* não dominado melhor, vence a solução com melhor *front*. Se ambas as soluções pertencem a um mesmo *front*, vence a solução que está localizada perto de um ponto de referência em uma região menos povoada. Por fim, há a possibilidade que os pontos de referência estejam distantes um dos outros e em consequência os elementos da população também podem estar longe entre si. Assim, o algoritmo utiliza um método de recombinação que gere as soluções filhas perto dos pais, para evitar que haja perda de convergência na busca. O algoritmo MO-NSGA-II é comparado com o MOEA/D para os problemas DTLZ1, DTLZ2, DTLZ3 e DTLZ4, em termos de convergência e diversidade, utilizando entre 3 e 10 funções objetivo. O algoritmo proposto produziu um resultado que apresenta boa convergência e boa diversidade, tendo, em geral, um melhor resultado do que o MOEA/D.

Outra abordagem é discutida em [1]. Nesse trabalho, é discutido que em problemas com muitos objetivos, duas características importantes na busca de MOEAs, em geral, entram em conflito: aproximação à fronteira de Pareto e diversidade das soluções. Além disso, é ressaltado, que devido à complexidade da busca em problemas com muitos ob-

jetivos, em geral só uma pequena parte da fronteira de Pareto é coberta, não havendo diversidade. Assim, o trabalho propõe novas estratégias para promover diversidade na Otimização com Muitos Objetivos. São propostos dois novos mecanismos de gerenciamento da diversidade: DM1, que ativa ou desativa a promoção da diversidade de acordo com o espalhamento das soluções da população e DM2, que adapta o intervalo da mutação de acordo com as condições locais da solução em termos de espalhamento e povoamento da região. Para a validação dos métodos, eles são aplicados ao algoritmo NSGA-II e comparados separadamente e juntos, aplicados ao problema DTLZ2, com 6 a 20 funções objetivo. São utilizadas as medidas *maximum spread* e *Generational Distance*. Os resultados mostraram que o mecanismo DM2 falhou em gerar uma boa distribuição, porém o DM1 melhorou os resultados do NSGA-II para muitos objetivos. Porém, o trabalho apresenta limitações, pois, diferentes trabalhos na literatura apontam a baixa performance do NSGA-II em problemas com muitos objetivos [81] [47]. Neste contexto, apesar de o mecanismo DM1 apresentar uma melhora da diversidade, não é possível identificar se essas melhorias são competitivas com os demais algoritmos para muitos objetivos presentes na literatura, tais como CDAS e IBEAs.

Em [82] é apresentado um amplo estudo que analisa como o crescimento do número de objetivos torna mais complexos os problemas multiobjetivo. Esse trabalho busca encontrar quais são as principais razões para a dificuldade de se otimizar um conjunto com muitas funções objetivos. Com esse objetivo, o trabalho busca analisar o comportamento da evolução de um algoritmo multiobjetivo com respeito ao número $m$ de objetivos. Para isto, observam-se somente as propriedades de convergência em relação a fronteira de Pareto (observado através de uma medida de distância). O objetivo é observar como a população evolui em direção à fronteira de Pareto, quando o número de objetivos cresce. O trabalho apresenta a ideia que o aumento da multimodalidade do problema influencia mais do que o aumento do número de objetivos. A justificativa dada pelos autores é que cada ótimo local de cada função objetivo também é um ótimo local para problema multiobjetivo. Assim, o aumento no número de objetivos aumenta a chance de novas soluções em ótimos locais. Assim, o trabalho propõe problemas unimodais com intuito de observar

o quanto o algoritmo NSGA-II [28] se aproxima da fronteira de Pareto quando o número de objetivos cresce nestes problemas. Como resultado, é observado que essa deterioração não é tão significante e conclui que para problema unimodais o aumento da complexidade dos problemas quando o número de objetivos cresce não é significante. Por fim, o trabalho argúi que adição de mais funções objetivos nem sempre torna o problema mais difícil. Esse trabalho apresenta um novo ponto de vista para se trabalhar com problemas com muitos objetivos e introduz um novo caminho para melhorar o desempenho de MOEAs tradicionais em cenários com muitos objetivos.

Outro trabalho que busca analisar a influência das funções objetivo em MOEAs é apresentado em [45]. Neste trabalho, é discutido o comportamento de um algoritmo multiobjetivo quando as funções são correlacionadas. Em geral, esta rotina também é explorada pelas abordagens que buscam a redução da dimensionalidade. Porém, este trabalho explora algoritmos conhecidos, tais como o NSGA-II, SPEA2 e o MOEA/D em um contexto onde os objetivos são altamente correlacionados, sem executar a redução do vetor objetivo. Como resultado, foi mostrado que a convergência do NSGA-II e do SPEA2 não é degradada demais quando o número de objetivos cresce com um conjunto de funções correlacionadas. Além disso, em alguns casos a adição de funções correlacionadas aos problemas utilizados melhorou o desempenho desses algoritmos. O MOEA/D não apresentou bons resultados quando o número de objetivos aumentou, mesmo quando eles eram altamente correlacionados. Por fim, o trabalho conclui que as características específicas dos algoritmos e dos problemas influenciam nos resultados das buscas para problemas com muitos objetivos. Uma crítica ao trabalho é que não são apresentadas quais características do NSGA-II e do SPEA2 possibilitam bons resultados com funções correlacionadas. Da mesma forma não há uma análise buscando-se identificar as dificuldades do MOEA/D neste contexto.

Outro importante campo de pesquisa na Otimização com Muitos Objetivos é a geração de problemas de bechmark. O conjunto de problemas que sejam escaláveis em relação ao número de objetivos é pequeno, o que limita a avaliação de novas técnicas. Em [31], Deb *et. al.* apresentam estratégias para a construção de problemas de testes com muitos objetivos.

As estratégias propostas enfatizam a possibilidade de construção de diferentes problemas, os quais podem possuir características como multimodalidade, geram mais soluções em determinadas regiões da fronteira de Pareto ou produzirem fronteiras desconexas. A partir das estratégias propostas, os autores desenvolveram uma família de problemas de teste, chamada de DTLZ, que é amplamente utilizada em diversos trabalhos da literatura. Essa família envolve diferentes tipos de problemas que podem ser utilizados para avaliar diferentes características de MOEAs em cenários com muitos objetivos. Alguns problemas DTLZ são discutidos no Capítulo 7.

## 2.3 Considerações finais

O estudo de problemas que possuem mais de três funções objetivos é recente e apresenta um conjunto de dificuldades que ainda não foram superadas. Através da análise de diversos trabalhos da literatura é possível concluir que a maior dificuldade enfrentada por Algoritmos Evolucionários Multiobjetivo é referente ao aumento do número de soluções não dominadas que dificulta a escolha das melhores soluções. Além disso, o aumento do número de soluções na fronteira de Pareto também é um obstáculo para a obtenção de um bom resultado da busca. Outra dificuldade relatada é o aumento de múltiplos ótimos locais com a adição de novos objetivos.

Para atenuar os efeitos negativos derivados dessas dificuldades na busca, diferentes técnicas têm sido propostas e aplicadas em diferentes meta-heurísticas e em diferentes problemas. Técnicas como modificar a relação de dominância, utilizar novas funções de *fitness* para os algoritmos evolucionários, reduzir o número de objetivos, entre outras, foram propostas. Essas técnicas conseguem reduzir a deterioração da qualidade da busca do algoritmos, proporcionando melhoras na convergência ou na diversidade. Porém, em geral, as técnicas propostas privilegiam uma característica em relação à outra: obtém-se uma busca que se aproxima da fronteira de Pareto, porém reduzida a uma pequena região, ou obtém-se um conjunto bem distribuído em relação à fronteira, porém com soluções subótimas. Em geral, a maioria dos métodos propostos privilegia a convergência e encontrar um boa distribuição das soluções em torno da fronteira de Pareto é um dos

principais desafios da Otimização com Muitos Objetivos.

Este trabalho tem como principal objetivo explorar a Otimização por Nuvem de Partículas Multiobjetivo na Otimização com Muitos Objetivos. As técnicas discutidas nesse capítulo serão aplicadas a algoritmos MOPSO e utilizadas como base para o desenvolvimento de novos algoritmos e novos métodos para Problemas de Otimização com Muitos Objetivos. O próximo capítulo apresenta a definição da Otimização por Nuvem de Partícula, meta-heurística escolhida nesta tese para enfrentar os problemas da Otimização com Muitos Objetivos.

# CAPÍTULO 3

# OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS MULTIOBJETIVO

Este capítulo apresenta a meta-heurística utilizada no desenvolvimento desta tese, a Otimização por Nuvem de Partículas Multiobjetivo. Essa meta-heurística apresenta bons resultados em diversos problemas de otimização, especialmente problemas multiobjetivo. Além disso, ela é pouco explorada com Problemas com Muitos Objetivos. O capítulo está organizado da seguinte forma: inicialmente os conceitos básicos da Otimização por Nuvem de Partículas são discutidos na Seção 3.1. Em seguida, mostra-se como essa meta-heurística é aplicada a problemas multiobjetivo, Seção 3.2. Alguns algoritmos que exploram a meta-heurística MOPSO são apresentados na Seção 3.3. A Seção 3.4 discute os principais trabalhos relacionados da área MOPSO, com ênfase nos algoritmos MOPSO da literatura, MOPSO com múltiplos enxames e alguns estudos que exploram o PSO em Problemas com Muitos Objetivos. Por fim, a Seção 3.5 apresenta as considerações finais deste capítulo.

## 3.1 Otimização por nuvem de partículas

A Otimização por Nuvem de Partícula, PSO (do inglês *Particle Swarm Optimization*), desenvolvida por Kennedy e Eberhart [54], é uma meta-heurística baseada em população inspirada no comportamento social de pássaros à procura de alimento. Na Otimização por Nuvem de Partículas, o conjunto das possíveis soluções é um conjunto de partículas, chamada de enxame (*swarm*) ou população, que se move no espaço de busca. Cada partícula representa uma possível solução e possui uma posição que é uma coordenada no espaço de variáveis de decisão. A ideia do PSO é executar um conjunto de operadores e movimentar cada partícula, através de uma busca cooperativa, para regiões promissoras no espaço de busca. A cada iteração um novo conjunto de soluções é obtido. Esses

movimentos são executados por um operador que ajusta a velocidade da partícula. O cálculo da velocidade é baseado na melhor posição encontrada por uma vizinhança da partícula (componente social) e pela melhor posição encontrada pela partícula até então (componente local). A seguir são descritos os principais termos utilizados na Otimização por Nuvem de Partículas:

- ***Swarm* ou Enxame**: População do algoritmo.
- **Partícula ($p$)**: Indivíduo da população. Cada partícula representa uma possível solução para o problema e possui os componentes posição e velocidade.
- **Posição ($\overrightarrow{x}(t)$)**: Componente da partícula que representa a posição da partícula no espaço de variáveis de decisão.
- **Velocidade ($\overrightarrow{v}(t)$)**: Operador que movimenta a partícula pelo espaço de estados em direção às melhores soluções do problema. É guiada pelo componente local, a melhor posição da partícula encontrada até então, e pelo componente social, a melhor posição já encontrada por alguma partícula de sua vizinhança.
- **Líderes**: Partículas da população que possuem os melhores valores da função objetivo para o problema.
- $\overrightarrow{p_{best}}$: Melhor posição já alcançada pela partícula, chamado de líder local, representa o componente local.
- $\overrightarrow{p_{lider}}$: Melhor posição já alcançada por uma partícula da vizinhança. Representa o componente social e é chamado de líder global.
- **Vizinhança**: Determina o conjunto de partículas que será usado como vizinhança de uma determinada partícula.
- **Peso de inércia ($w$)**: Usado para controlar a influência dos valores anteriores da velocidade no cálculo da velocidade atual.

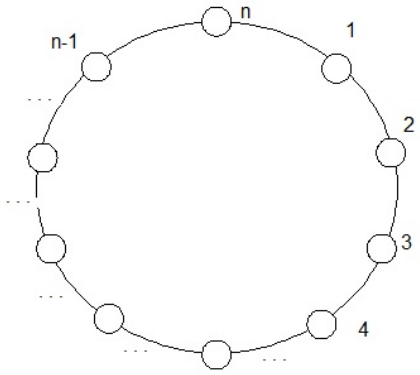



Figura 3.1: Topologia de vizinhança em forma de anel. Cada círculo representa uma partícula.

- **Fator de aprendizado**: Representa a atração que uma partícula terá em direção ao seu próprio sucesso ($p_{best}$) ou ao sucesso do enxame ($p_{lider}$). Representado por $C_1$, peso de atração em direção ao sucesso da partícula (componente cognitivo) e $C_2$, peso de atração em direção ao sucesso dos vizinhos (componente social).

A próxima seção irá discutir algumas topologias de vizinhança existentes. A Seção 3.1.2 apresenta as equações de movimento das partículas pelo espaço de busca.

### 3.1.1 Topologias de vizinhança

Na Otimização por Nuvem de Partículas um conjunto de possíveis soluções, representadas através de partículas, movimenta-se pelo espaço de busca. Esse movimento é influenciado pela própria história da partícula e por um componente social, que é definido através de um conjunto de partículas, chamado vizinhança. A forma como essa vizinhança está definida pode alterar como a busca é efetuada, pois ela define a forma como os líderes serão selecionados. A seguir são apresentadas as principais topologias de vizinhança utilizadas [79].

- **Grafo vazio**: Nesta vizinhança a partícula está conectada somente à ela mesma. Assim, não há a influência das demais partículas no seu movimento.

- **Estrela**: Nesta topologia todas as partículas estão conectadas somente a uma partícula, chamada de partícula focal (Figura 3.2). A partícula focal compara o

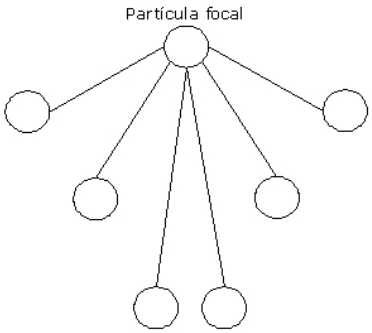



Figura 3.2: Topologia de vizinhança em forma de estrela. Cada círculo representa uma partícula.

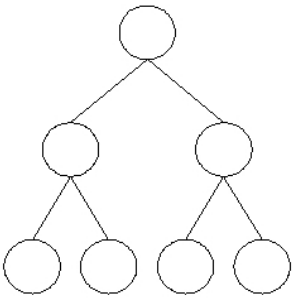


Figura 3.3: Topologia de vizinhança em forma de árvore. Cada círculo representa uma partícula.

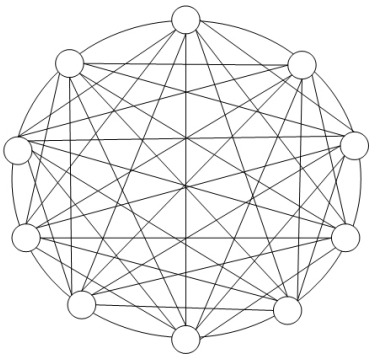


Figura 3.4: Topologia de vizinhança em forma de grafo completo. Cada círculo representa uma partícula.

desempenho de todas as partículas da população e efetua o seu movimento de acordo com a melhor partícula. As demais partículas se movimentam de acordo com a partícula focal, que propaga a escolha da melhor solução para as demais partículas. Essa topologia consegue produzir um movimento com a influência da melhor partícula da população sem executar muitas operações, porém a propagação da melhor posição depende da posição da partícula focal.

- **Árvore**: As partículas estão organizadas em árvore (Figura 3.3). Cada partícula sofre influência de sua partícula pai na árvore, o seu líder é o seu pai na árvore. Se uma partícula em um nó filho encontrar uma melhor posição no espaço de estados que seu pai, é feita uma troca de posição entre estas duas partículas. Esta topologia propicia um acesso rápido às melhores soluções do problema, porém a estrutura dinâmica da árvore torna a busca complexa.

- **Grafo completo**: Nesta vizinhança a partícula está conectada a todas as partículas do enxame (Figura 3.4). $p_{lider}$ é definido como sendo a melhor partícula entre todas as partículas da população. Possibilita que todas as partículas se influenciem pela melhor partícula da população, mas executa o maior número de operações entre todas as topologias.

- **Melhor local**: Nesta vizinhança a partícula está conectada a $k$ partículas. O líder é definido como sendo a melhor partícula da vizinhança $k$. Equivalente à topologia de anel se $k = 2$ (Figura 3.1) e equivalente ao grafo completo caso $k$ seja igual ao número total de partículas menos 1. A topologia de anel efetua poucas operações para o cálculo do movimento, porém este movimento é influenciado por um número pequeno de partículas.

### 3.1.2 Equações de movimento das partículas

A Otimização por Nuvem de Partículas é definida por um conjunto de equações que movimenta as partículas no espaço de estado em direção às melhores soluções do problema. Cada partícula $p_i$, em um determinado tempo $t$ (o tempo normalmente é representado

pela iteração da execução), possui uma posição $\overrightarrow{x(t)} \in R^n$. A posição da partícula, num tempo $t+1$, é obtida adicionando-se a velocidade, $\overrightarrow{v(t)} \in R^n$, a $\overrightarrow{x(t)}$, tal que:

$$\overrightarrow{x}(t+1) = \overrightarrow{x}(t) + \overrightarrow{v}(t) \tag{3.1}$$

A velocidade da partícula $p_i$ , no tempo $t+1$, é definida:

$$\overrightarrow{v}(t+1) = \varpi \cdot \overrightarrow{v}(t) + C_1 \cdot \phi_1 \cdot (\overrightarrow{p}_{best}(t) - \overrightarrow{x}(t)) + C_2 * \phi_2 \cdot (\overrightarrow{p}_{lider}(t) - \overrightarrow{x}(t)) \tag{3.2}$$

onde $\phi_1$ e $\phi_2$, na Equação 3.2, são números reais aleatórios e são sorteados a cada iteração. $C_1$ e $C_2$ são constantes utilizadas para definir a influência do componente local e do componente local no cálculo da velocidade da partícula. No algoritmo do PSO, após a atualização da velocidade e da posição de todas as partículas o processo é repetido na próxima geração (iteração) até o final da execução do algoritmo.

As melhores partículas da população são encontradas através da função de *fitness*, que é a função objetivo do problema. O *fitness* de uma partícula $p_i$, representado como $\alpha(p)$, é uma função em relação à posição da partícula, $\alpha : S \subseteq R^n \rightarrow R$. Então, para problemas de minimização, em um tempo $t$, $p_i$ é melhor que $p_j$, se (O inverso é verdade para problemas de maximização):

$$\alpha(\overrightarrow{x_i}(t)) < \alpha(\overrightarrow{x_j}(t))$$

O Algoritmo 1 mostra o pseudo-código da Otimização por Nuvem de Partículas. O primeiro passo é a inicialização da população. Essa inicialização pode ser um procedimento aleatório, estratégias gulosas ou a hibridização com outras meta-heurísticas [55]. Após a inicialização da população há a escolha do líder, que é a partícula com maior valor de *fitness*. Após essa etapa de configuração inicial o laço evolutivo é executado. Neste laço são executadas as operações de atualização da posição e dos líderes descritas anteriormente. Ao final do laço, a partícula como melhor *fitness* é considerada a solução do problema.

**Algoritmo 1** Pseudo-código da otimização por nuvem de partículas

```
1: Iniciar população
2: Definir líderes
3: Enquanto condição de término não ocorrer
4:    Para cada partícula p_i da população faça
5:       Atualizar a posição e a velocidade de p_i
6:       Avaliar o fitness
7:       Atualizar p_best
8:    Atualizar o líder
9: Retornar melhor solução da população
```

### 3.1.3 Equação da velocidade com fator de constrição

Em [18] é proposta uma extensão da equação original do PSO para evitar um crescimento ilimitado da velocidade das partículas. Para isto, foi adicionado um fator de constrição $\chi$ que multiplica toda a equação da velocidade original, Equação 3.2. Este fator de constrição, $\chi$, é definido através da Equação 3.4, que utiliza o valor $\varphi$ como base, que, por sua vez, é definido através de $C_1$ e $C_2$ pela Equação 3.5. A Equação 3.3 apresenta a equação da velocidade modificada, tal que:

$$\overrightarrow{v}(t+1) = \chi \cdot (\varpi \cdot \overrightarrow{v}(t) + (C_1 \cdot \phi_1) \cdot (\overrightarrow{p}_{best}(t) - \overrightarrow{x}(t)) + C_2 \cdot \phi_2 \cdot (\overrightarrow{p}_{lider}(t) - \overrightarrow{x}(t))) \quad (3.3)$$

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi}|} \quad (3.4)$$

$$\varphi = C_1 + C_2 \quad (3.5)$$

## 3.2 Otimização por nuvem de partículas multiobjetivo

Na Otimização por Nuvem de Partícula Multiobjetivo, MOPSO (do inglês *Multi-Objective Particle Swarm Optimization*), existem várias funções objetivos e é possível encontrar soluções que explorem os conceitos da dominância de Pareto. Baseado nesses conceitos, discutidos no Capítulo 2, não há somente uma melhor solução, mas sim um conjunto das

melhores soluções. Assim, cada partícula do enxame pode ter diferentes líderes, mas para aplicação do operador de velocidade somente um deve ser escolhido.

Este conjunto de líderes é guardado num arquivo externo (ou repositório) que contém as soluções não dominadas encontradas até então na busca. Normalmente, esse arquivo é limitado e possui um tamanho máximo, pois, sem a definição de um limite o número de soluções no repositório cresce e a busca se aproxima de uma execução exaustiva, onde todos os líderes possíveis são considerados. Duas importantes características de um algoritmo MOPSO são: o método para arquivar as soluções no repositório (arquivamento dos líderes) e como as partículas vão escolher os seus líderes (escolha do líder). Os tópicos de arquivamento dos líderes e escolha dos líderes são discutidos nas Seções 3.2.1 e 3.2.2, respectivamente.

Os componentes do MOPSO são semelhantes ao algoritmo PSO monobjetivo. A velocidade da partícula $p_i$ no PSO multiobjetivo é baseada na melhor posição alcançada pela partícula até então, $\overrightarrow{p}_{best}(t)$, e a posição de uma partícula do repositório, que é escolhida como líder global de $p_i$, $\overrightarrow{R}_h$. A função de atualização da velocidade no tempo $t+1$ para um algoritmo MOPSO é definida através da Equação 3.6. O fator de constrição $\chi$ pode ser adicionado à Equação 3.6 da mesma forma apresentada na Seção 3.1.3, tal que:

$$\begin{aligned}\overrightarrow{v}(t+1) = \varpi * \overrightarrow{v}(t) + (C_1 \cdot \phi_1) \cdot (\overrightarrow{p}_{best}(t) - \overrightarrow{x}(t)) \\ +(C_2 \cdot \phi_2) \cdot (\overrightarrow{R_h}(t) - \overrightarrow{x}(t))\end{aligned} \tag{3.6}$$

As próximas seções discutem os métodos para arquivamento e seleção dos líderes. Esses métodos, podem ser aplicados tanto para a obtenção dos líderes locais e globais. Porém, a influência do líder global é mais importante para um algoritmo MOPSO, pois é através destes líderes que há cooperação entre as partículas. Nas seções a seguir a discussão será restrita à influência de cada método na escolha do líder global, porém as mesmas estratégias podem ser utilizadas para definição do $\overrightarrow{p}_{best}(t)$.

### 3.2.1 Arquivamento dos líderes

Os líderes, soluções não dominadas encontradas até então, são guardados em um arquivo externo através de métodos de arquivamento. Esse processo de arquivamento não é exclusivo para o MOPSO, mas sim utilizado em diversas outras meta-heurísticas como em alguns algoritmos genéticos, algoritmos AIS, entre outros [64]. Porém, no MOPSO, o conjunto de líderes é utilizado para guiar diretamente as partículas para as melhores posições do espaço de busca e não funciona somente como um repositório que guarda as melhores soluções. Assim, os diferentes métodos para se inserir ou retirar as soluções do arquivo externo influenciam diretamente a busca de um algoritmo MOPSO. O arquivo contém a posição $\overrightarrow{x}$ da partícula $p_i$.

Basicamente, dada uma nova solução $\overrightarrow{x}$ gerada por um MOEA (a partir de $\overrightarrow{x}$ é possível calcular $\overrightarrow{f(\overrightarrow{x})}$ ), o objetivo do `arquivador` é manter as melhores soluções (somente pontos não dominados) encontradas até então na busca [20]. Esse arquivo é limitado em um tamanho $N$ e o arquivador deve utilizar diferentes estratégias para introduzir aspectos como convergência e diversidade na busca [64].

Uma formalização de um arquivo básico com tamanho $N$ é apresentada em [22] e é definida uma classe simples de arquivadores, chamados de Arquivadores Precisos (*precise archivers*). Dado uma sequência de pontos (soluções geradas pelo algoritmo) o objetivo do arquivador é manter um subconjunto destes pontos em um arquivo $A$, onde $|A| \leq N$. Além disso, $A^t$ denota o arquivo após a apresentação do $t$-ésimo vetor objetivo [64].

Dado um arquivo $A^{t-1}$, ele contém somente soluções com vetores objetivo não dominados até então. Seja $\overrightarrow{x}$ uma nova solução tentando entrar no arquivo $A^{t-1}$, de tamanho máximo $N$, o algoritmo básico de um arquivador preciso é apresentado no Algoritmo 2. Um arquivador preciso possui algumas características (representadas pelos números em negrito no Algoritmo 2): (1) dada o ponto $\overrightarrow{x}$, se $\overrightarrow{x}$ domina alguma solução em $A^{t-1}$, $\overrightarrow{x}$ é adicionado em $A^{t-1}$ e os pontos dominados por $\overrightarrow{x}$ são removidos, obtendo-se $A^t$; (2) se $\overrightarrow{x}$ é dominado por algum ponto de $A^{t-1}$, $A^{t-1}$ permanece sem modificações, então $A^{t-1} = A^t$; (3) se $|A^{t-1}| < N$, $\overrightarrow{x}$ não domina nenhuma ponto em $A^{t-1}$ e é não dominado por nenhum ponto em $A^{t-1}$, $\overrightarrow{x}$ é adicionado $A^t$; (4) se o arquivo está cheio, $|A^{t-1}| = N$,

**Algoritmo 2** Pseudo-código do arquivador preciso

**Entrada:** $A^{t-1}$, $\overrightarrow{x}$, $N$ (Arquivo atual, nova solução, Tamanho máximo do arquivo )
**Saída:** $A^t$

Se $\exists \overrightarrow{y} \in A^{t-1}$, tal que $\overrightarrow{f(\overrightarrow{x})} \preceq \overrightarrow{f(\overrightarrow{y})}$ então
    $A^t \leftarrow A^{t-1} \cup \overrightarrow{x} / \{\forall \overrightarrow{y} \in A^{t-1}, \overrightarrow{f(\overrightarrow{x})} \preceq \overrightarrow{f(\overrightarrow{y})}\}$ **(1)**
senão Se $\exists \overrightarrow{y} \in A^{t-1}$, tal que $\overrightarrow{f(\overrightarrow{y})} \preceq \overrightarrow{f(\overrightarrow{x})}$ então
    $A^t \leftarrow A^{t-1}$ **(2)**
senão Se $\nexists \overrightarrow{y} \in A^{t-1}$ tal que, $\overrightarrow{f(\overrightarrow{x})} \preceq \overrightarrow{f(\overrightarrow{y})}$ ou $\overrightarrow{f(\overrightarrow{y})} \preceq \overrightarrow{f(\overrightarrow{x})}$ então
    Se $(|A| < N)$ então
        $A^t \leftarrow A^{t-1} \cup \overrightarrow{x}$ **(3)**
    senão $A^t \leftarrow filtro(A^{t-1}, \overrightarrow{x})$ **(4)**

$\overrightarrow{x}$ não domina e não é dominado por nenhum ponto em $A^{t-1}$,a função filtro é aplicada para definir quais $N$ soluções vão permanecer no conjunto $A^{t-1} \cup \overrightarrow{x}$.

Na literatura existem diferentes métodos que são utilizados como arquivadores. Em [64], Ibanez *et al.* apresentam um importante trabalho que analisa o comportamento de diversos métodos de arquivamento em Problemas Multiobjetivo. Nesse trabalho, oito arquivadores são estudados e importantes características de cada um são observadas, através de uma análise teórica e de uma análise experimental. A seguir são descritos 6 métodos estudados por Ibanez *et al.* [64] que seguem o padrão do arquivador preciso.

**Arquivador *Adaptive Grid*:** o Gride Adaptativo (*Adaptive Grid*) foi proposto em [58] e força uma distribuição uniforme de pontos não dominados no espaço de objetivos, colocando estes pontos em diferentes células de uma gride. Nessa gride, o espaço de objetivos é dividido com o intuito de medir a densidade de diferentes regiões. O arquivador *Adaptive Grid* usa essa gride como um mecanismo para introduzir diversidade nos pontos selecionados. Nesse arquivador, os pontos guardados no arquivo são organizados na gride. A função $filtro(A^{t-1}, \overrightarrow{x})$ adiciona o novo ponto $\overrightarrow{x}$ ao arquivo, reconstrói a gride e então remove um ponto da célula mais povoada, de forma aleatória.

**Arquivador *Average Rank*:** Esse arquivador utiliza o valor do método de ranking *Average Rank* (AR) (discutido no Capítulo 4) para definir quais soluções irão permanecer no arquivo. Métodos de ranking buscam associar os valores do vetor de funções objetivo para produzir uma nova ordenação das soluções. Em geral, esses métodos são utilizados

como relações de preferência e são aplicados no contexto da Otimização com Muitos Objetivos para substituir a relação da dominância de Pareto. A função $filtro(A^{t-1}, \overrightarrow{x})$ desse arquivador calcula o valor do AR para todas as soluções do conjunto $A^{t-1} \cup \overrightarrow{x}$ e remove a solução com maior valor do ranking. Em geral, esse método privilegia soluções no extremos da fronteira de Pareto [47].

**Arquivador *Crowding Distance* ou Arquivador do NSGA-II:** Esse arquivador usa a distância de *Crowding* (CD), apresentada no Capítulo 2 para decidir quais pontos irão continuar no arquivo externo. Esse arquivador busca introduzir mais diversidade na busca. A função $filtro(A^{t-1}, \overrightarrow{x})$ adiciona $\overrightarrow{x}$ ao arquivo, calcula o novo valor de CD entre os pontos do arquivo e remove o ponto com o pior valor de CD, isto é, o ponto localizado na região mais povoada do espaço de objetivos.

**Arquivador Dominante:** Esse é um arquivador simples, somente adiciona ao arquivo os pontos que tenham dominado algum outro ponto do arquivo (propriedade (1) do Arquivador Preciso). Basicamente, a função $filtro(A^{t-1}, \overrightarrow{x})$ retorna $A^{t-1}$.

***Multi-level Grid Archiving* (MGA):** Esse arquivador, proposto por [61], combina o esquema do *Grid* Adaptativo com o método de arquivamento $\epsilon$-Pareto (apresentado mais à frente neste capítulo). Nesse método, se o arquivo se tornar cheio, a função $filtro(A^{t-1}, \overrightarrow{x})$ adiciona $\overrightarrow{x}$ à $A^{t-1}$ e divide o espaço de objetivo em blocos. Esses blocos são obtidos através de intervalos, definidos por um incremento $b$, que dividem de forma uniforme cada eixo de uma função objetivo, até que todos os pontos façam parte de um bloco. Cada bloco é definido por um índice que define o encontro de diferentes divisões de cada eixo. Cada ponto no arquivo é associado à um índice de um bloco. Após, é verificada a relação de dominância entre esses índices dos blocos. Se existir alguma solução $\overrightarrow{y}$ que pertence a um bloco dominado, um dos pontos que pertença à um bloco dominado é removido de forma aleatória. Se não há nenhuma relação de dominância entre os blocos, o espaço de objetivos é re-dividido em blocos menores, com novo intervalo $b$, menor que o anterior, até que se encontre uma dominância entre os blocos.

A Figura 3.5 mostra um exemplo do funcionamento do método MGA, para um espaço com dois objetivos. O eixo de coordenadas da esquerda, mostra os vetores objetivos de

um conjunto de soluções pertencentes a um arquivo posicionados numa gride dividido em um intervalo $b_1$. Nessa primeira situação, os índices de cada gride estão definidos através de marcações em vermelho. Nessa primeira situação não há bloco que contenha alguma ponto que domine outro bloco. Assim, a gride é reconstruída utilizando um novo valor $b_2$, sendo que $b_2 < b_1$. Nessa nova gride, o bloco marcado com o rótulo 1 domina os blocos 2 e 3. A função filtro do arquivador MGA remove de forma aleatória um dos 3 pontos que pertencem aos blocos 2 e 3.

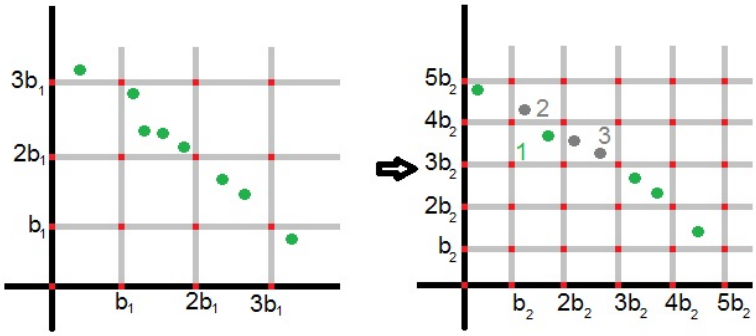



Figura 3.5: Representação do método MGA para um espaço com dois objetivos.

**Arquivador aleatório:** A função $filtro(A^{t-1}, \overrightarrow{x})$ remove algum ponto aleatório do conjunto $A^{t-1} \cup \overrightarrow{x}$.

**Arquivador Não Limitado (*Unbound archiver*):** Nesse método, não há um limite $N$ para o arquivo externo. A função $filtro(A^{t-1}, \overrightarrow{x})$ retorna $A^{t-1} \cup \overrightarrow{x}$.

Além dos Arquivadores Precisos também é possível utilizar diferentes estratégias para preencher o arquivo externo. Em geral, essas estratégias também só permitem que soluções com vetor objetivo não dominado entrem no arquivo, porém ao invés de verificar se o arquivo está cheio a cada novo ponto gerado pelo algoritmo, é feito um procedimento ao final da iteração para se eliminar um conjunto de pontos não desejados pelo arquivo. Uma estratégia utilizada em alguns MOEAs é utilizar conceitos da dominância $\epsilon$ [20] para controlar a entrada de soluções no arquivo. Dois arquivadores que utilizam essa estratégia são:

**Arquivamento *Adaptive* $\epsilon$*-Approx* ($\epsilon$app):** Basicamente, ao invés de usar os conceitos da dominância de Pareto para decidir quais soluções entram no arquivo, esse arquivador utiliza a dominância $\epsilon$. O método somente aceita um novo ponto se ele domina

pela dominância $\epsilon$ algum outro elemento do arquivo [60]. Essa abordagem não limita o arquivo em um tamanho $N$, porém há a possibilidade de se estender esse conceito para limitar o arquivo, como é mostrado em [60]. O valor $\epsilon$ é definido como parâmetro.

**Arquivamento *Adaptive* $\epsilon$-*Pareto* Archiving ($\epsilon$aps):** Esse arquivador discretiza o espaço de objetivos em diferentes blocos definidos pelo valor $\epsilon$ [60]. Inicialmente todo ponto não dominado entra no arquivo. Após o final de uma iteração do algoritmo, dado um conjunto de pontos do arquivo é efetuado um procedimento que elimina alguns pontos desses arquivo. Esse procedimento efetua as seguintes tarefas: para cada função objetivo é feita a discretização desse eixo em intervalos com valor $\epsilon$. Assim, de forma análoga ao arquivador MGA, são definidos blocos, que são representados por um índice. A ideia é que todo ponto do arquivo deve pertencer a um bloco específico. Se dois ou mais pontos pertencerem a um mesmo bloco, somente é selecionado o ponto mais próximo do índice do bloco. Esse arquivador também não possui um limite máximo de $N$ pontos.

### 3.2.2 Escolha dos líderes

A escolha do líder global, $\overrightarrow{R_h}$, é um importante fator de influência no movimento das partículas num algoritmo MOPSO. A escolha de $\overrightarrow{R_h}$ é um dos responsáveis em ditar características como convergência e diversidade que irão afetar todo o enxame. Encontrar a melhor estratégia para escolher $\overrightarrow{R_h}$ é uma tarefa importante para algoritmos que exploram conceitos multiobjetivo na meta-heurística PSO. Se a escolha de $\overrightarrow{R_h}$ é em uma região mais povoada, então será privilegiada a convergência, porém a busca pode convergir para um ótimo local. Se a escolha for em regiões menos povoadas, há um aumento da diversidade, porém pode haver perda do poder de convergência. Visando explorar essas características, alguns métodos foram propostos na literatura e são discutidos a seguir:

**Método Sigma:** Nesse método, dada uma partícula $p_i$ e seu vetor de funções objetivo ($\overrightarrow{f(\overrightarrow{x_i})}$), é calculado um vetor $\sigma_i$ que denota a direção de uma reta que passa entre a origem e o ponto do vetor objetivo. O líder de $p_i$ ($\overrightarrow{R_h}$), é o ponto do arquivo externo que possui a menor distância Euclidiana entre o seu vetor sigma e o vetor sigma $p_i$, ou seja, o líder é um ponto que possui uma direção semelhante à partícula $p_i$. A ideia é que

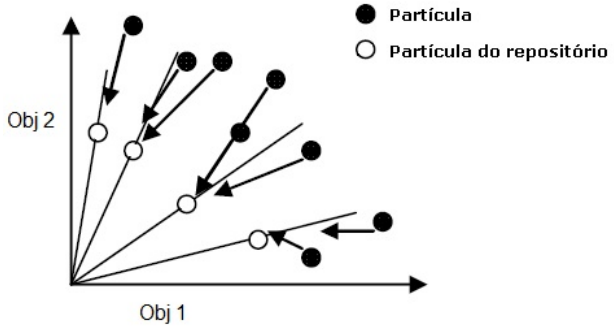



Figura 3.6: Representação do método Sigma para um espaço com dois objetivos.

vetores que tenham um vetor sigma semelhante estão localizados em regiões semelhantes (ver Figura 3.6), haverá uma maior distribuição dos líderes presentes no arquivo externo entre as partículas da população.

Para um problema com duas funções objetivos, $f_1(\overrightarrow{x})$ e $f_2(\overrightarrow{x})$, o vetor sigma de uma partícula é definido com um só valor calculado através da Equação(3.7).

$$\sigma = \frac{f_1(\overrightarrow{x})^2 - f_2(\overrightarrow{x})^2}{f_1(\overrightarrow{x})^2 + f_2(\overrightarrow{x})^2} \tag{3.7}$$

Para problemas com mais de dois objetivos, o vetor sigma é formado com $\binom{m}{2}$ elementos, representando a combinação da Equação(3.7) para todos objetivos. A Figura 3.6 mostra um exemplo do método sigma para um espaço com dois objetivos. Os círculos escuros representam as partículas da população e os círculos claros as partículas do repositório. Os segmentos de retas indicam o vetor sigma dos líderes e cada partícula da população escolhe a partícula do repositório que possui menor distância Euclidiana, em relação ao seu vetor sigma.

**Torneio Binário com distância de *Crowding*:** a escolha do líder através do método do torneio binário com distância de *Crowding* é baseada na seleção executada pelo algoritmo NSGA-II [28]. Inicialmente, é calculada a distância de *Crowding* entre todos os líderes. Em seguida, dois líderes são escolhidos de forma aleatória com distribuição uniforme. O líder $\overrightarrow{R_h}$ é o ponto que tiver o maior valor de CD, ou seja, a solução localizada numa área menos povoada.

**NWSum:** Esse método foi proposto em [72] e consiste em guiar as partículas através

de líderes que estejam localizados perto do eixo de coordenadas o qual a partícula está mais próxima. Assim, seja $dim_{prox}$ a dimensão mais próxima da posição da partícula $p_i$. O líder será a partícula do repositório mais próxima de $dim_{prox}$. Isso é feito através do cálculo de pesos para cada valor objetivo de $p_i$, com este conjunto de pesos é feito uma soma ponderada do vetor objetivo de cada líder $l_i$. O líder escolhido é o que maximizar essa soma ponderada. O valor do NWSum é calculado através da equação 3.8, em que $\overrightarrow{x_i}$ é posição da partícula $p_i$ e $\overrightarrow{f(\overrightarrow{x_i})}$ seu vetor objetivo.

$$F = \sum_j \frac{f_j(\overrightarrow{x_i})}{\sum_k f_k(\overrightarrow{x_i})} f_j(\overrightarrow{l_i}) \tag{3.8}$$

**Escolha aleatória:** Essa abordagem é a mais simples e menos complexa computacionalmente e foi inicialmente apresentada em [21]. Nesse método, cada partícula escolhe um líder do repositório de forma aleatória. Esse método beneficia a diversidade se os líderes estiverem bem distribuídos no espaço de objetivos. Porém, se houver um conjunto de líderes agrupado em uma região, isso resulta numa perda de diversidade na busca.

### 3.2.3 Múltiplos enxames

A Otimização por Nuvem de Partículas é definida por um conjunto de indivíduos que se movimentam de forma cooperativa no espaço de busca. Essa ideia é estendida através dos algoritmos PSO com múltiplos enxames [36]. Múltiplos enxames utilizam várias populações (*multi-swarms*)[1] executando de forma independente trocando informações entre si durante a busca. Cada população independente, chamadas subenxame (*sub-swarm*), é uma execução individual de um algoritmo PSO, contendo características específicas como próprio conjunto de líderes, método de arquivamento e de escolha de líder, equações do movimento, ou seja, para cada *sub-swarm* pode-se executar um algoritmo PSO diferente. O principal ganho de se utilizar diferentes populações otimizando um mesmo problema é reduzir a complexidade do problema, como por exemplo, focando a busca em diferentes regiões do espaço de busca ou reduzindo o número de objetivo otimizados por cada

[1]Os termos múltiplos enxames e *multi-swarm* serão utilizados da mesma forma nesta tese para indicar algoritmos MOPSO com várias populações.

**Algoritmo 3** Pseudo-código de um algoritmo PSO com múltiplos enxames

```
1: Para cada sub-swarm s_i faça
2:  Execute o movimento das partículas de cada sub-swarm
3:  Se condição de troca for satisfeita
4:    Execute a troca de informações entre os sub-swarms
5: Retornar melhores soluções encontradas pelos sub-swarms
```

*sub-swarm*.

O Algoritmo 3 apresenta o pseudo-código do PSO com múltiplos enxames. Dada uma condição de troca de informações, os *sub-swarms* trocam informações entre si através de diferentes abordagens. Essa condição define quando será feita a troca de informações, como por exemplo: trocar a cada iteração; somente quando haja alteração no líderes; quando houver uma movimentação significante das partículas no espaço de variáveis de decisão, etc. As abordagens de troca definem quais informações serão repassadas entre cada *sub-swarm* tais como: cada sub-swarm troca os líderes; no caso multiobjetivo pod-se trocar todo o conjunto ou somente um subconjunto de líderes entre os *sub-swarms*; pode-se trocar partículas entre as populações. Outro fator importante nessas estratégias é a topologia em que os *sub-swarms* efetuam essa troca [12]. Em geral, são utilizadas as mesmas topologias da vizinhanças das partículas, discutidas na Seção 3.1.1.

## 3.3 Algoritmos MOPSO

Existem diferentes algoritmos baseados na meta-heurística da Otimização por Nuvem de Partículas propostos na literatura. Basicamente, esses algoritmos buscam definir diferentes métodos para escolha de líder e o arquivamento dos líderes. Porém, há também a exploração de outros aspectos da meta-heurística, tais como a introdução de mutação nas soluções [20], mecanismos para controlar de forma explícita a velocidade das partículas, a aplicação de diferentes relações de preferência ao invés da dominância de Pareto, entre outros.

As próximas seções apresentam detalhes de dois algoritmos MOPSO, SMPSO [70] e SigmaMOPSO [67]. Esses algoritmos são escolhidos como base nesta tese, pois de acordo com os resultado apresentados em [35], [70] e [79] apresentam bons resultados em

Problemas Multiobjetivo. Além disso, os algoritmos utilizam diferentes abordagens tanto na escolha e arquivamento dos líderes, equações de velocidade e mutação.

### 3.3.1 Algoritmo SMPSO

O algoritmo SMPSO foi apresentado em [70] e segue os passos básicos de um algoritmo MOPSO. Ele tem como principal característica limitar a velocidade das partículas, através de valores definidos pelo usuário. Esse limite impede que as partículas percorram regiões do espaço de busca longe do resultado desejado e então o algoritmo converge mais rápido.

Na proposta, a velocidade da partícula é limitada através de valores de limites superiores (*upper_limit*) e de limites inferiores (*lower_limit*). Cada posição do vetor velocidade é limitada individualmente e caso o valor supere os limites definidos pelo usuário, o valor da velocidade é definido pela Equação 3.9. As Equações 3.9 e 3.10 representam o limite da velocidade de $p_i$ para cada valor da velocidade $j$, onde:

$$v_{i,j} = \begin{cases} delta_j & \text{Se } v_{i,j}(t) > delta_j, \\ -delta_j & \text{Se } v_{i,j}(t) \leq -delta_j, \\ v_{i,j} & \text{Caso contrário.} \end{cases} \tag{3.9}$$

$$delta_j = \frac{upper\_limit_j - lower\_limit_j}{2} \tag{3.10}$$

Outra característica é o uso da equação da velocidade com o fator de constrição modificada. As equações utilizadas no SMPSO apresentadas em [70] são:

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi}} \tag{3.11}$$

$$\varphi = \begin{cases} C_1 + C_2 & \text{se } C_1 + C_2 > 4, \\ 1 & \text{se } C_1 + C_2 \leq 4. \end{cases} \tag{3.12}$$

Na Equação 3.11, quando $\varphi = 1$, o termo $\sqrt{\varphi^2 - 4 \cdot \varphi}$ possui uma raiz negativa.

Na implementação desse algoritmo nesta tese, quando $\varphi = 1$ o valor de $\sqrt{\varphi^2 - 4 \cdot \varphi}$ foi definido como 0.

O método de arquivamento utilizado pelo do SMPSO é efetuado através do Arquivador *Crowding Distance* (Seção 3.2.1), visando introduzir maior diversidade na busca. Para a escolha do líder (definição do componente social, $\overrightarrow{R_h}(t)$), é utilizado o Torneio Binário com distância de *Crowding* (Seção 3.2.2), também buscando um aumentado da diversidade da busca. O líder local, $\overrightarrow{p_{best}}$, é a melhor posição atingida pelo algoritmo. Se uma nova posição alcançada pela partícula tiver uma relação de não dominância com o $\overrightarrow{p_{best}}$ atual, o novo valor para $\overrightarrow{p_{best}}$ é definido de forma aleatória com distribuição uniforme entre as duas posições.

Além disso, o algoritmo utiliza um operador de mutação em 15% das partículas. Um número pequeno de partículas é escolhido para mutação, para evitar que haja muita perturbação na busca e o algoritmo perca convergência, já que tanto o arquivamento, quanto a escolha do líder privilegiam a diversidade. As partículas são escolhidas aleatoriamente e é aplicada a mutação polinomial [28], com probabilidade de $\frac{1}{n}$, onde $n$ é o número de variáveis do problema.

Para a validação do SMPSO, no estudo apresentado em [70], ele é comparado a cinco algoritmos do estado da arte, o NSGA-II [28], o SPEA2 [90], o AbYSS [69], baseado em busca por dispersão, o MOCell [71], um algoritmo genético celular e o OMOPSO [83]. Os algoritmos foram avaliados nos problemas DTLZ e ZTD utilizando os indicadores Hipervolume, o Epsilon [91] e o Spread [84]. Conclui-se que o SMPSO supera os demais algoritmos, obtém os melhores conjuntos de aproximação e possui uma convergência mais rápida em direção à fronteira de Pareto.

### 3.3.2 Algoritmo SigmaMOPSO

O algoritmo SigmaMOPSO foi proposto em [67]. A principal contribuição desse algoritmo é a proposta do método de escolha do líder, método Sigma, discutido na Seção 3.2.2. Esse método, de acordo com os resultados apresentados em [79], é um dos mais adequados para a técnica MOPSO. O método de arquivamento utilizado é baseado numa técnica

agrupamento dos líderes apresentada em [87]. Nesse método, se o número de soluções for maior que o tamanho do arquivo é executado um algoritmo para agrupar os líderes em pelo menos $|N|$ grupos (número máximo de pontos no arquivo). O $\overrightarrow{p_{best}}$ é escolhido de forma aleatória caso haja uma relação de não dominância entre $\overrightarrow{p_{best}}$ e a nova posição alcançada pela partícula. Além disso, é aplicado um operador de mutação simples em todas as partículas com probabilidade de 1% para cada posição. Nessa mutação, cada posição da partícula pode ser alterada por um valor $R_t$ que varia aleatoriamente no intervalo $[0, 1]$.

O método sigma foi comparado com outro método de escolha de líder, chamado de D-tree [37] e conseguiu encontrar soluções com melhor convergência e diversidade em funções de teste com 2 e 3 funções objetivo. O algoritmo foi comparado com o SPEA2 [90] e também obteve um melhor resultado em termos de convergência e diversidade. Além disso, pesquisas recentes [35], [72], [79], entre outros, mostraram bons resultados do método Sigma.

## 3.4 Trabalhos relacionados

Existem na literatura diversos trabalhos relacionados que exploram tópicos da Otimização com Nuvem de Partículas Multiobjetivo. Alguns desses trabalhos foram apresentados nesse capítulo na introdução dos conceitos da meta-heurística. Esta seção discute alguns trabalhos relacionados que são importantes no desenvolvimento desta tese.

Dentre os trabalhos que exploram conceitos da meta-heurística MOPSO, destaca-se [72]. Esse trabalho analisa a influência de métodos de escolha de líderes na busca de um algoritmo MOPSO. Diferentes métodos da literatura são revisados, em que a convergência, diversidade e tempo computacional são analisados. Além de utilizar métodos da literatura, esse estudo propõe um novo método, chamado de NWSum, previamente discutido nesta tese. Todos os métodos são comparados utilizando-se diferente problemas multiobjetivo (usando até 3 funções objetivo). Foi apontado que o método NWSum e o método Sigma se destacam, obtendo os melhores resultados em termos de convergência e diversidade. Além disso, esse trabalho conclui que a escolha do líder realmente modifica o desempenho de algoritmo MOPSO e é importante considerar diferentes métodos de escolha líder quando

novos algoritmos MOPSO forem projetados.

Em [35] é apresentado um estudo que compara os melhores algoritmos MOPSO da literatura, escolhidos através do trabalho [79]. Esse trabalho tem com objetivo mostrar as capacidades de cada algoritmo. Os algoritmos escolhidos são: NSPSO [62], Sigma-MOPSO [67], OMPSO [83], AMOPSO [76], MOPSOpd [3] e uma abordagem não descrita em [79], MOCLPSO [42]. Esses algoritmos utilizam diferentes abordagens de preenchimento do arquivo externo, escolha do líder e operador de mutação. Para a avaliação dos algoritmos, eles são aplicados em diferentes problemas de *benchmark* amplamente utilizadas na literatura multiobjetivo, como a família DTLZ [31], a família ZTD [88] e a família WFG [43]. Todos os problemas foram utilizados com dois objetivos. Para a medição da qualidade dos conjuntos gerados por os algoritmos MOPSO, é utilizado um conjunto de indicadores de qualidade como o Hipervolume, indicador Epsilon [91] e *Spread* [84]. Nos experimentos, o algoritmo OMPSO se destacou, obtendo os melhores resultados na maioria dos problemas. O algoritmo SigmaMOPSO também apresentou bons resultados. Porém, os algoritmos não conseguiram convergir em alguns problemas multimodais. Para resolver esse problema, foi proposto um novo método que limita a velocidade dos algoritmos PSO. Com este novo método foi proposto um novo algoritmo, chamado de SMPSO. Esse algoritmo é mais bem detalhado em [70]. O novo algoritmo proposto foi comparado com o OMOPSO e com o NSGA-II e apresentou resultados competitivos. Os resultados apresentados neste artigo serviram de base para a escolha dos algoritmos PSO utilizados em nosso trabalho. Os algoritmos SMPSO e SigmaMOPSO são explorados nesta tese e são detalhados nas Seções 3.3.1 e 3.3.2.

As próxima seções compilam a discussão trabalhos relacionados sobre os tópicos MOPSO com múltiplos exames, Seção 3.4.1 e trabalhos que utilizam a meta-heurística MOPSO no contexto da Otimização com Muitos Objetivos, Seção 3.4.2.

### 3.4.1 Algoritmos MOPSO com múltiplos enxames da literatura

El-Abd e Kamel [36] apresentam um survey que descreve diversos algoritmos MOPSO com múltiplos enxames da literatura. Os algoritmos desenvolvidos nesse estudo cobrem

uma gama de diferentes problemas, dentre eles problemas monoobjetivo, multiobjetivo e problemas de otimização com restrição. A partir da análise e descrição de diversos trabalhos da literatura, o estudo apresentado por El-Abd e Kamel propõe uma taxonomia para a classificação dos algoritmos PSO com múltiplos enxames. Essa taxonomia é baseada nos métodos de decomposição do problema adotados por cada algoritmo. De forma geral, os algoritmos são divididos entre os algoritmos que usam múltiplos enxames para trocar soluções entre si, chamado de método de decomposição do problema e os algoritmos que utilizam múltiplos enxames para cobrir diferentes áreas do espaço de busca, chamado de método de decomposição do espaço. Os trabalhos discutidos a seguir são restritos à otimização multiobjetivo, principal tema desta tese.

Dentre os trabalhos de PSO com múltiplos enxames, destacam-se os algoritmos que utilizam *multi-swarms* na otimização multiobjetivo. Em [73] é discutida a aplicação de um algoritmo PSO *multi-swarm* em problemas com muitos objetivos. É proposta a paralelização do algoritmo VEPSO [74] e a aplicação deste algoritmo em muitos objetivos. O VEPSO é uma variante cooperativa do PSO baseado no algoritmo *Vector Evaluated Genetic Algorithm* (VEGA) [74]. O VEGA é um algoritmo evolucionário que seleciona subpopulações da geração de acordo com cada objetivo separadamente. Após a seleção, as subpopulações são misturadas e uma próxima geração é obtida através de etapas de recombinação e mutação. No VEPSO, cada objetivo é otimizado por um *swarm* e durante a busca os *swarms* trocam informações sobre as melhores soluções encontradas até então. Nesse trabalho, o VEPSO é aplicado a diferentes problemas com muitos objetivos, variando entre 2 e 10 objetivos. Porém, o VEPSO é apenas comparado ao VEGA, para cada objetivo, e não é feita nenhuma análise sobre o desempenho dos VEPSO em relação ao aumento do número objetivos. Não se sabe como o VEPSO se comporta quando o número de objetivos cresce, só é conhecido o seu desempenho em relação ao VEGA. É mostrado que o VEPSO consegue melhores resultados que o VEGA para todos os objetivos.

Em [56] é apresentado um algoritmo *multi-swarm* que é aplicado ao problema de aprendizado da resposta de agentes. Nesse estudo, a função de premiação é dividida em várias funções e o problema é modelado como um problema multiobjetivo. Para a

solução do problema é proposto um novo algoritmo PSO com múltiplos enxames, chamado de MS-PSO. Como no VEPSO, este algoritmo também define um *sub-swarm* para cada objetivo a ser otimizado, porém possui uma forma diferente de troca de informações entre os *swarms*. Nesta abordagem, cada *swarm* vizinho troca informação através de um novo componente na equação da velocidade das partículas, que representa a distância da partícula para a melhor partícula do *swarm* vizinho. O MS-PSO é comparado com o algoritmo PSO tradicional e o algoritmo MO-PSO [21]. O MO-PSO não é discutido em nosso trabalho, pois não é representativo e não apresenta bons resultados quando comparado aos algoritmos PSO do estado-da-arte. Os algoritmos foram utilizados em dois ambientes diferentes do agente e o MS-PSO conseguiu obter melhores resultados que o algoritmo básico do PSO e que o algoritmo MO-PSO.

Outro algoritmo que utiliza *multi-swarm* para problemas multiobjetivo é apresentado em [76]. Porém, diferentemente dos trabalhos anteriores, todos os *sub-swarms* otimizam todos os objetivos. Nesse trabalho, o *multi-swarm* é utilizado com o intuito de cobrir diferentes partes do espaço de busca em uma só execução, evitando o deslocamento desnecessário de partículas. Nessa abordagem, cada *swarm* representa um próprio nicho, com um mesmo número de partículas e com seus próprios líderes. Cada nicho é otimizado separadamente e ao final da atualização de todos os *swarms*, todos os líderes, de todos os nichos, são agrupados. Em seguida, os líderes semelhantes são divididos entre os *swarms* existentes. Os resultados do algoritmo proposto, chamado de AMOPSO, são comparados com alguns algoritmos multiobjetivo, como o NSGA-II [28] e o MO-PSO [21]. Os algoritmos são aplicados em funções de teste com dois objetivos e são consideradas medidas como Generational Distance e Spacing [84]. Conclui-se que o algoritmo AMOPSO é competitivo com os melhores algoritmos multiobjetivo da literatura.

Por fim, outro trabalho relevante é apresentado em [68]. Aqui, é apresentado um algoritmo multi-swarm que divide a busca em diferentes regiões de forma explícita. Nesse algoritmo, cada *sub-swarm* limita a região de busca de sua população. Essa região é definida através de uma partícula semente, a qual será o centro da região de busca. Em cada iteração do algoritmo, as sementes são escolhidas e é gerada a população dentro da região

definida pela semente. O algoritmo executa um processo iterativo de escolha das sementes e definição das novas regiões de busca. Ao final de cada iteração as melhores soluções de cada *sub-swarm* são reunidas e as novas sementes são definidas. Esse processo é repetido por um número de iterações definido pelo usuário. O trabalho propõe diferentes métodos de escolha das sementes. Um dos métodos é baseado em agrupamento: inicialmente as soluções são agrupadas utilizado o vetor objetivo; o número de grupos é definido como o número de *sub-swarms* do algoritmo; a semente é o centróide de cada grupo. Esse método é utilizado como base para o desenvolvimento de um algoritmo *multi-swarm* aplicado à problemas com muitos objetivos proposto nesta tese, discutido no Capítulo 6.

### 3.4.2 Algoritmos MOPSO aplicados à Otimização com Muitos Objetivos

Como discutido nas seções anteriores, diversos algoritmos e métodos foram propostos para utilizar meta-heurística da Otimização por Nuvem de Partículas em Problemas Multiobjetivo. Porém, da mesma forma que as demais meta-heurísticas multiobjetivo, o PSO enfrenta os problemas discutidos no Capítulo 2 quando aplicados a Problemas com Muitos Objetivos.

Visando a atenuação desses problemas, recentemente algumas pesquisas focam na meta-heurística MOPSO aplicada à Otimização com Muitos Objetivos. Na literatura, a maioria dos estudos que focam em problemas com muitos objetivos utiliza abordagens evolucionárias. Porém, são poucos os trabalhos que visam explorar as qualidades do PSO em problemas com muitos objetivos.

Em [85] é apresentada uma abordagem que usa uma métrica de distância baseada nas preferências do usuário para encontrar as melhores soluções. Nesse estudo, o usuário define boas regiões no espaço de objetivos que devem ser exploradas pelo algoritmo. O PSO é utilizado como base e as partículas atualizam sua posição e velocidade de acordo com sua proximidade às regiões preferenciais. Nesse método, o algoritmo PSO não conta com a relação da dominância de Pareto para encontrar as melhores soluções. O algoritmo proposto é comparado com um algoritmo PSO também baseado na preferência do usuário

que utiliza a dominância de Pareto. Os resultados mostraram que o algoritmo proposto obtém melhores resultados, em especial quando o número de objetivos é grande. Isso mostra que é possível explorar um algoritmo MOPSO definido por um região de preferência. Esse comportamento é explorado nesta tese através da proposta do Arquivador Hiperplano. Outra abordagem é proposta em [59]. Nesse trabalho, é proposto um algoritmo PSO que trabalha com muitos objetivos usando uma dominância de Pareto gradual para superar o problema.

Em [52] é utilizada a técnica *Self-Controlling Dominance Area* (SCDAS) para gerar uma nova classificação das soluções não dominadas no arquivo externo. Esse método é uma extensão da técnica Controle da Área de Dominância das Soluções [81], que é estudada nesta tese no Capítulo 4. No algoritmo proposto por *Castro Junior et al.*, chamado de SCDAS-SMPSO, a técnica SCDAS é utilizada para refinar o arquivo externo após cada iteração do algoritmo. No algoritmo proposto, o algoritmo SMPSO é utilizado como algoritmo PSO base, discutido na Seção 3.3.1. A técnica SCDAS é semelhante à CDAS e busca modificar a relação da dominância de Pareto, com o intuito que algumas soluções não dominadas passem a ser dominadas. Nesse método, o grau de expansão da relação de dominância é controlado automaticamente pelo algoritmo e além disso a técnica possibilita que as soluções nos extremos de cada dimensão não sejam removidas. No método proposto por *Castro Junior et al.*, ao final da iteração do PSO, a técnica SCDAS é aplicada aos pontos do arquivo externo, logo, a relação de dominância entre essas soluções é alterada. Em seguida, com a nova relação de dominância é feita uma poda, removendo-se as soluções que passaram a ser dominadas. O algoritmo SCDAS-SMPSO é comparado ao algoritmo CDAS-MOPSO (desenvolvido nesta tese e apresentado na Seção 4.1.2) e com o algoritmo SMPSO. A comparação buscou observar aspectos como convergência e diversidade da busca em cenários com muitos objetivos. O SCDAS-SMPSO conseguiu obter uma convergência equivalente ao CDAS-MOPSO, porém apresentou melhor diversidade. Além disso, o SCDAS-SMPSO é mais simples que o CDAS-MOPSO, pois não precisa de configuração do parâmetro que define o grau de extensão da dominância de Pareto.

Outro trabalho que explora o PSO na Otimização com Muitos Objetivos é apresentado

em [17]. Esse trabalho tem como objetivo estudar como diferentes métodos de escolha de líder funcionam em Problemas com Muitos Objetivos. Com esse intuito, é utilizado como base o algoritmo CDAS-MOPSO e observa-se o comportamento do algoritmo, em termos de convergência e diversidade, quando são utilizados diferentes métodos para escolha do líder. Nesse trabalho são utilizados os métodos do torneio binário com distância de *crowding* [70], WSUM [5], NWSum [72], método sigma [67] e escolha aleatória [21]. Os métodos foram aplicados aos problemas DTLZ2 e DTLZ4, variando entre 2 e 30 funções objetivo. Os melhores resultados foram obtidos pelos métodos NWSum e Sigma, especialmente pois eles introduziram maior convergência na busca. É importante ressaltar que o método original do algoritmo CDAS-MOPSO é o torneio binário com distância de *crowding* e o trabalho [17] estendeu os resultados iniciais do algoritmo CDAS-MOPSO.

## 3.5 Considerações finais

A Otimização com Nuvem de Partículas é uma meta-heurística quem tem sido aplicada com sucesso na Otimização Multiobjetivo [79]. Diferentemente das abordagens evolucionárias (que são mais utilizas em trabalhos multiobjetivo), o PSO tem como base a cooperação entre os indivíduos da população. Através dessa cooperação entre os indivíduos, é possível controlar aspectos como convergência e diversidade da busca e desenvolver diferentes estratégias para a solução de problemas de otimização.

Na otimização multiobjetivo, pode-se destacar dois aspectos de um algoritmo MOPSO: arquivamento dos líderes e escolha dos líderes. Essas características são os pontos mais importantes de um algoritmo MOPSO. Outro característica importante que pode ser explorada em algoritmos MOPSO é a possibilidade de se utilizar múltiplos enxames, com o objetivo de se reduzir a complexidade do problema.

Porém, apesar dos bons resultados dos algoritmos MOPSO, essa meta-heurística ainda é pouco utilizada na Otimização com Muitos Objetivos. O objetivo desta tese de doutorado é explorar diferentes aspectos do MOPSO na Otimização com Muitos Objetivos, tais como a introdução de novas relações de preferência, métodos de arquivamento, uso de múltiplos exames, entre outros. Todos esses temas serão discutidos nos próximos

capítulos.

# CAPÍTULO 4

# NOVAS RELAÇÕES PREFERÊNCIAS APLICADAS AO MOPSO NA OTIMIZAÇÃO COM MUITOS OBJETIVOS

Este capítulo apresenta a primeira estratégia explorada nesta tese para a extensão da Otimização por Nuvem de Partículas Multiobjetivo aplicada a Problemas com Muitos Objetivos: o uso de novas relações de preferência. Como discutido nos capítulos anteriores, a relação da dominância de Pareto se mostra ineficiente quando o número de funções objetivo cresce. Visando atenuar essa deterioração, alguns trabalhos da literatura apresentam novas estratégias baseadas na definição de novas relações de preferência [47] [81] [23]. A ideia utilizada nesses métodos é construir uma função de fitness para um algoritmo que possibilite a definição das melhores soluções mesmo com um número grande de funções objetivo.

Dentre esses trabalhos, dois possuem destaque: o Controle da Área de Dominância das Soluções [81] e Métodos de Ranking de Soluções [23]. Essas técnicas apresentam bons resultados em problemas com muitos objetivos em trabalhos da literatura, são técnicas simples, podem ser aplicadas em diferentes metaheurísticas e são técnicas que utilizam princípios diferentes (modificação da dominância e rankeamento das soluções), o que possibilita o estudo de diferentes abordagens em nossa tese. Outro fator importante é que elas ainda foram pouco exploradas em algoritmos MOPSO.

A primeira estratégia explorada é a aplicação da técnica do Controle da Área de Dominância das Soluções em algoritmos PSO, discutida na Seção 4.1. A segunda contribuição, apresentada na Seção 4.2, visa explorar diferentes métodos de ranking em algoritmos MOPSO. São propostos novos algoritmos que utilizam métodos de ranking da literatura, bem como, novos métodos de ranking.

## 4.1 Controlando a área de dominância das soluções em algoritmos MOPSO

A técnica do Controle da Área de Dominância das Soluções (CDAS, do inglês *Controlling the Dominance Area of Solutions*) foi proposta em [81]. Esse método controla a área de dominância das soluções para induzir uma ordenação das soluções apropriada para o problema, especialmente com muitos objetivos. Como mostrado em [47] e [81] a técnica CDAS apresenta bons resultados quando aplicada a problemas com muitos objetivos e produz resultados melhores que outras estratégias como o *Average Ranking* [23] e a relação favour [34]. Em nosso trabalho a influência dessa técnica em um algoritmo MOPSO é analisada. As próximas seções apresentam a técnica CDAS, Seção 4.1.1, e descreve os passos da aplicação do CDAS no MOPSO na Seção 4.1.2.

### 4.1.1 Controle da área de dominância das soluções

Este método propõe o controle do grau de contração ou expansão da área de dominância das soluções através de um parâmetro definido pelo usuário, $S_i$. Os valores dos objetivos da solução são modificados através de relações trigonométricas definidas pelo valor de $S_i$. O vetor de objetivos é transladado no espaço de objetivos e assim a relação de dominância muda com essa contração ou expansão. Assim, soluções que originalmente eram não dominadas passam a ser dominadas por outras e vice-versa. A modificação da área de dominância é definida pela Equação 4.1 e a representação desta modificação é apresentada na Figura 4.1, tal que:

$$f_i'(\overrightarrow{x}) = \frac{r \cdot sin(\omega_i - S_i\pi)}{sin(S_i\pi)} \tag{4.1}$$

onde $\overrightarrow{x}$ é uma solução no espaço de busca, $\overrightarrow{f(\overrightarrow{x})}$ é o vetor objetivo, $r$ é a norma de $\overrightarrow{f(\overrightarrow{x})}$ e $\omega_i$ é o grau entre $f_i(\overrightarrow{x})$ e $\overrightarrow{f(\overrightarrow{x})}$. A Equação 4.1 realiza a operação trigonométrica que translada cada valor objetivo, $f_i(x)$, para os novos valores, $f_i'(\overrightarrow{x})$. Esta operação pode ser observada através da Figura 4.1. Observe que dado o ângulo $\varphi = S_i\pi$, algumas operações

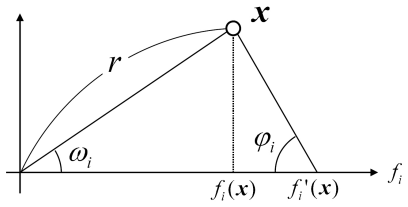
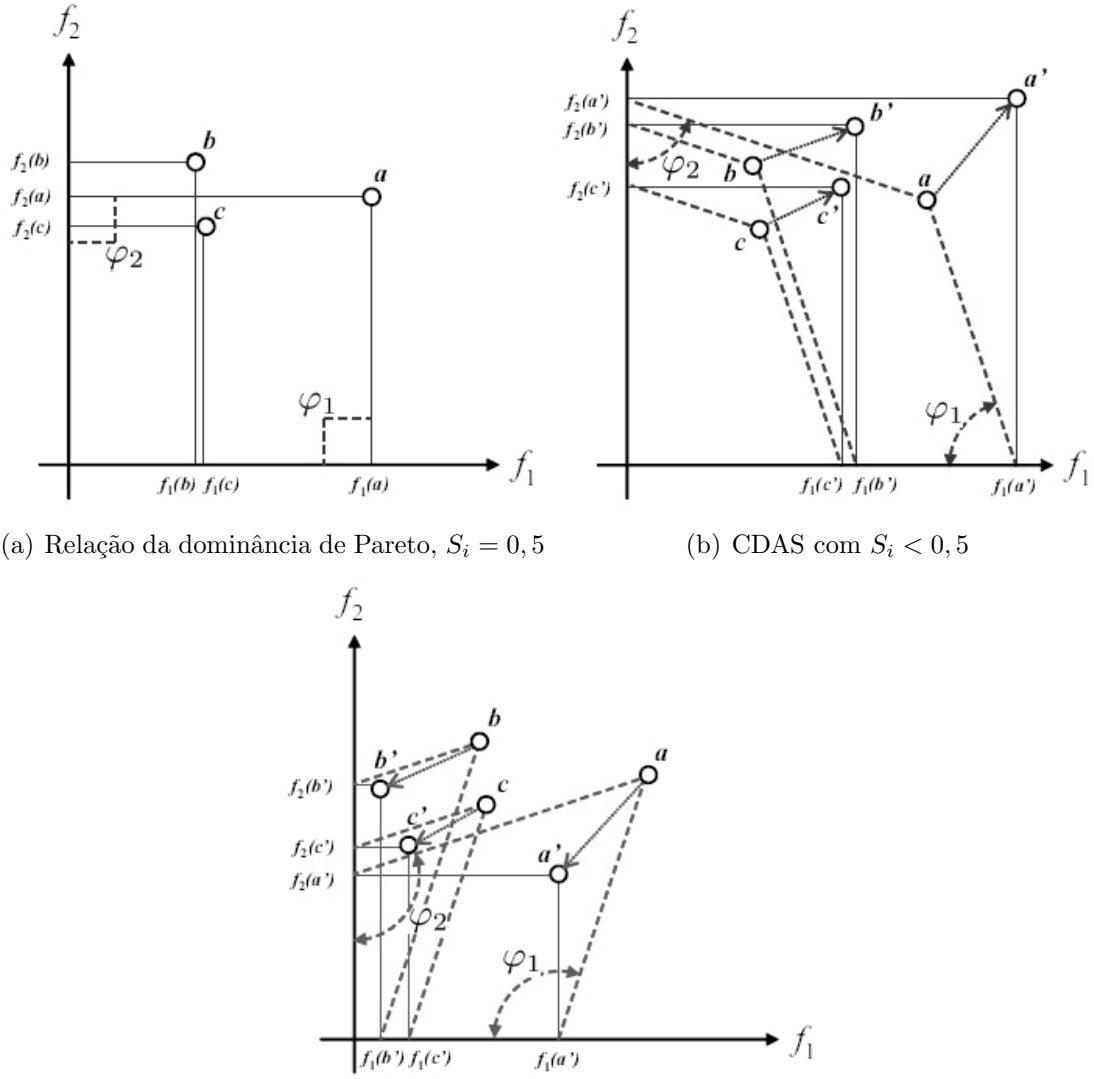



Figura 4.1: Modificação da relação de dominância com o CDAS [81].

trigonométricas são realizadas com os valores de $r$ e $\omega_i$ e é definida a translação de $f_i(\overrightarrow{x})$ para $f_i'(\overrightarrow{x})$.

A Figura 4.2 apresenta um exemplo do funcionamento do CDAS para dois objetivos [81]. Os círculos $a$, $b$ e $c$ são vetores objetivos de três soluções distintas. Se $S_i = 0,5$, $\varphi = \frac{\pi}{2}$ então $f_i'(\overrightarrow{x}) = f_i(\overrightarrow{x})$ e não ocorre modificação da relação de dominância. Na Figura 4.2(a) o ponto $c$ é dominado pelo ponto $a$. Se $S_i < 0,5$ então $f_i'(\overrightarrow{x}) > f_i(\overrightarrow{x})$, há um aumento em cada valor do vetor de objetivos e será produzido um subconjunto das soluções não dominadas. A Figura 4.2(b) mostra a alteração da dominância de Pareto quando se usa $S_i < 0,5$, percebe-se que os valores dos objetivos são alterados para se modificar a área de dominância e o ponto $b$ passar a ser dominado por $a$. Por outro lado, se $S_i > 0,5$ então $f_i'(\overrightarrow{x}) < f_i(\overrightarrow{x})$, logo a relação de dominância é relaxada e as soluções que normalmente são dominadas passam a ser não dominadas. Na Figura 4.2(c) percebe-se que com a diminuição da área de dominância das soluções, o ponto $c$ passa a ser não dominado. Os resultados apresentados em [81] mostraram que é possível obter melhor desempenho, em termos de convergência e diversidade, para problemas com muitos objetivos quando se controla a área de dominância. Os experimentos foram conduzidos utilizando o algoritmo NSGA-II [28] aplicado ao problema da mochila multiobjetivo. Embora seja possível utilizar o parâmetro $S_i$ com valores $> 0,5$, na Otimização com Muitos Objetivos estamos interessados em reduzir o número de soluções não dominadas. Logo, em nossa tese a escolha de $S_i$ será limitada a valores menores do que $0,5$. Além disso, trabalhos prévios do autor, apresentados em [25] e [14], mostraram que ao executar o CDAS com valores de $S_i < 0,5$, obtém-se resultados melhores do que ao se utilizar $S_i > 0,5$. Outros trabalhos da literatura, como o trabalho apresentado em [47], também desconsideram a

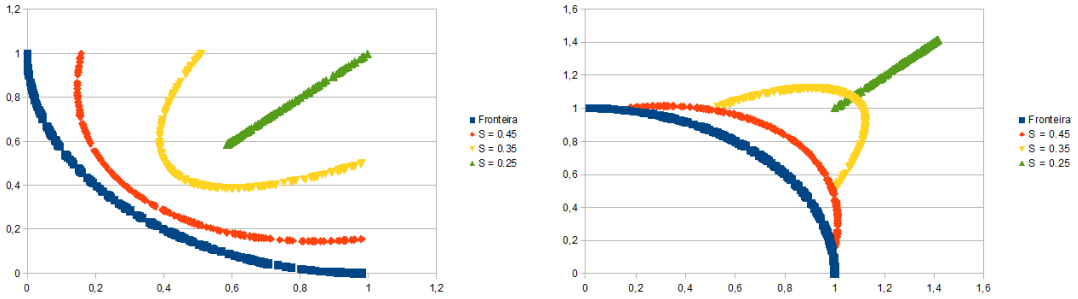



(a) Relação da dominância de Pareto, $S_i = 0,5$

(b) CDAS com $S_i < 0,5$

(c) CDAS com $S_i > 0,5$

Figura 4.2: Exemplo do funcionamento do CDAS para duas funções objetivo [81].



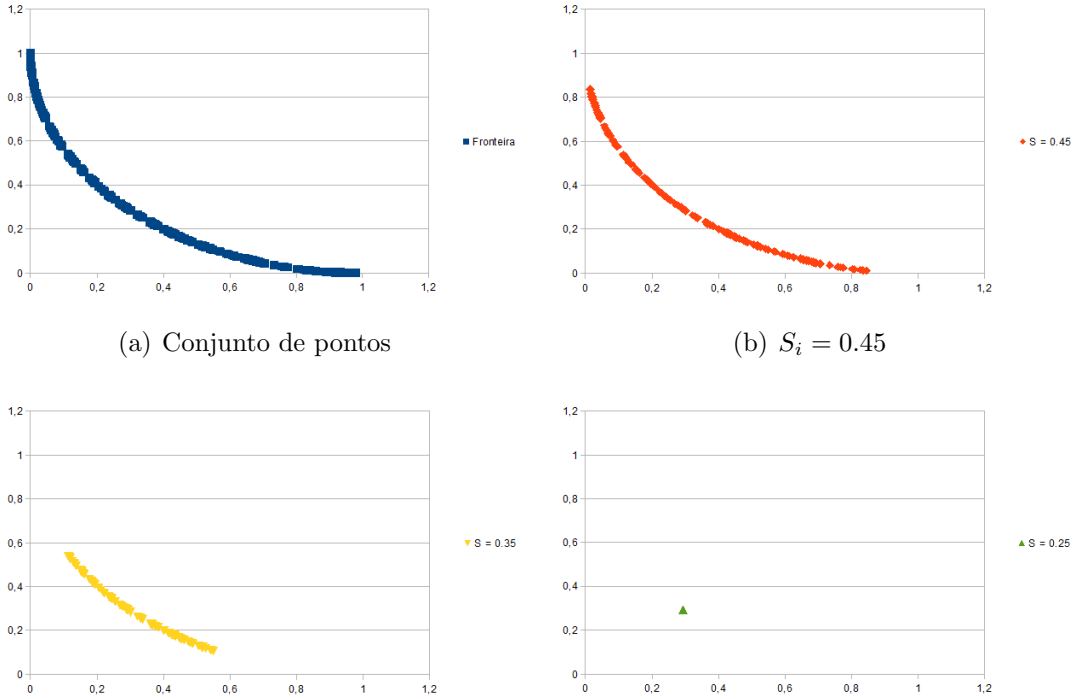

(a) CDAS num problema de minimização (b) CDAS num problema de maximização

Figura 4.3: Exemplo da translação de um conjunto pontos efetuada pelo CDAS.

execução do CDAS com valores de $S_i > 0,5$. Por fim, a extensão do CDAS, chamada de S-CDAS [80] é baseada na execução do CDAS com valores de $S_i < 0,5$.

A Figura 4.3 mostra o exemplo da translação de um conjunto de pontos para problemas de minimização e maximização, Figuras 4.3(a) e 4.3(b), respectivamente. Cada curva contém um total de 100 pontos. A curva maior representa uma fronteira de Pareto genérica para problemas de minimização e maximização. As demais curvas representam os pontos da fronteira transladados com diferentes valores de $S_i$, variando entre $S_i = 0,45, 0,35$ e $0,25$. É possível notar que a localização dos pontos do espaço de objetivos é alterada e que quanto menor é o valor de $S_i$ maior é essa alteração.

Para se ter um melhor entendimento da técnica CDAS a Figura 4.4 mostra um exemplo do cálculo da técnica com diferentes valores $S_i$. A Figura 4.4(a) mostra um conjunto de pontos que representa a fronteira de Pareto para um problema de minimização. As Figuras 4.4(b), 4.4(c) e 4.4(d) mostram as soluções não dominadas após o cálculo do CDAS de todos os pontos do conjunto original (Figura 4.4(a)) com valores de $S_i = 0,45$, $0,35$ e $0,25$, respectivamente. Nessas figuras, cada ponto é transladado utilizando a técnica CDAS e em seguida é checada a relação de dominância entre os novos vetores. São apresentados somente os pontos que possuem os vetores modificados não dominados. Percebe-se que quanto menor é o valor de $S_i$, o conjunto de pontos é mais reduzido e são privilegiados os pontos localizados no centro da curva. Quanto mais próximo o valor de $S_i$ é de $0,5$, maior é o número de pontos e o conjunto é semelhante ao conjunto original, porém sempre descartando os pontos nos extremos.




(a) Conjunto de pontos

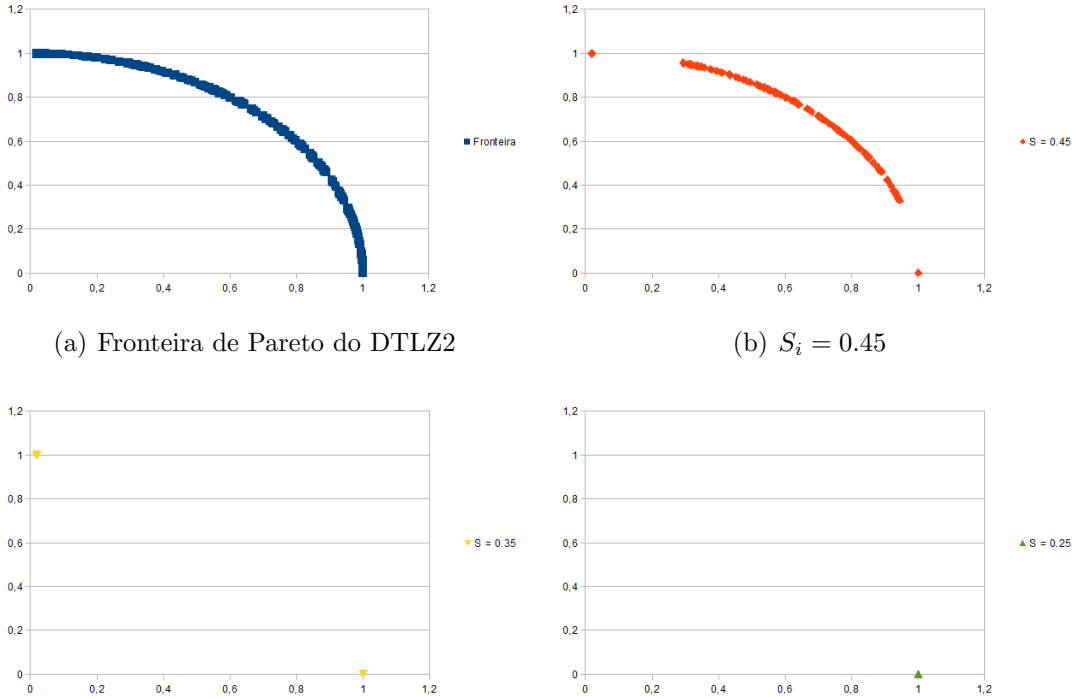
(b) $S_i = 0.45$

(c) $S_i = 0.35$

(d) $S_i = 0.25$

Figura 4.4: Pontos não dominados após a execução do CDAS para um problema de minimização.

A Figura 4.5 apresenta um cenário diferente. Na Figura 4.5(a) é apresentada a fronteira de Pareto do problema DTLZ2 [31]. O DTLZ2 é um problema de minimização e para duas funções objetivo é definido pela função de um quadrante de uma esfera de raio 1 e centro no ponto $(0, 0)$. Na aplicação do CDAS com valores de $S_i = 0,45$, $0,35$ e $0,25$ (Figuras 4.5(b), 4.5(c) e 4.5(d), respectivamente) é observado que quanto menor o valor de $S_i$, menor é o número soluções não dominadas. Diferentemente dos outros cenários, as soluções extremas são privilegiadas. Percebe-se que os pontos dos extremos nunca são dominados e com a diminuição do valor de $S_i$, os pontos mais próximos aos extremos são dominados. Logo, nessa situação os pontos não dominados que permanecem são os pontos nos extremos e os pontos localizados ao centro do conjunto, sendo que quando o valor de $S_i$ for definido com um valor baixo (entre $S_i = 0,25$ e $0,35$) somente restam as soluções no extremo.




(a) Fronteira de Pareto do DTLZ2 (b) $S_i = 0.45$

(c) $S_i = 0.35$ (d) $S_i = 0.25$

Figura 4.5: Pontos não dominados após a execução do CDAS para o problema DTLZ2.

### 4.1.2 Algoritmo CDAS-MOPSO

A primeira contribuição desta tese refere-se à aplicação da técnica do controle da área de dominância das soluções em algoritmos PSO multiobjetivo [13] [14] [16]. No MOPSO a convergência e a diversidade são controladas através da cooperação entre as partículas, ou seja, é controlada através da seleção e escolha dos líderes. Os líderes guiam o exame para as melhores áreas no espaço de busca, então dependendo de como é feita essa escolha a solução pode convergir para uma pequena área da fronteira de Pareto ou executar uma busca mais diversificada, buscando cobrir uma região maior da fronteira de Pareto.

O CDAS modifica a relação de dominância, selecionando um subconjunto das soluções não dominadas, de acordo com o comportamento discutido na seção anterior. Essa técnica é incorporada na busca do MOPSO de uma maneira simples: o passo que atualiza o repositório com as soluções não dominadas é modificado e agora aplica a nova relação de dominância definida pela Equação 4.1.

A partir dessa nova relação de preferência, foi desenvolvido um novo algoritmo, cha-

mado de CDAS-MOPSO. É importante ressaltar que as demais características do MOPSO permanecem inalteradas, assim é possível combinar a técnica CDAS com diferentes métodos de arquivamento, escolha do líder, equações de velocidade, entre outros. O Algoritmo 4 apresenta um descrição do algoritmo proposto. O algoritmo segue os passos do algoritmo básico MOPSO, apresentado no Capítulo 3. O primeiro passo é a inicialização do algoritmo MOPSO. Após a população ter sido iniciada, devem ser definidas as soluções não dominadas. Então, é aplicado o método CDAS com o valor $S_i$ passado como parâmetro pelo usuário. Com os novos vetores objetivo modificados, as soluções não dominadas são definidas e é feito o arquivamento dessas soluções no arquivo externo. Após a definição dos líderes, as partículas da população escolhem seus líderes globais e é iniciado o laço evolutivo. Nesse laço, para cada partícula $p_i$ da população, a primeira etapa é o movimento da partícula pelo espaço de busca. Em seguida, deve ser feito o cálculo das funções objetivo da nova posição e efetuada a modificação desse vetor através do método CDAS. Em seguida, a partícula (com o vetor de funções objetivo já modificado) é submetida ao arquivador. Na comparação de dominância entre a nova solução e as demais soluções do arquivo externo é sempre utilizado o vetor de objetivos modificado pelo CDAS. Ao final do movimento das partículas é utilizado algum método para a definição dos líderes para todos os elementos da população. Por fim, ao final do laço evolutivo o retorno do algoritmo são as partículas do arquivo externo.

O CDAS MOPSO é baseado no algoritmo SMPSO. Assim, ele utiliza as mesmas definições apresentadas na Seção 3.3.1, ou seja, uso da equação de velocidade com constrição, mutação polinomial em 15% das soluções, escolha dos líderes por distância de *Crowding* e uso do Arquivador *Crowding Distance*.

O desenvolvimento do algoritmo CDAS-MOPSO em diferentes algoritmos MOPSO da literatura gerou algumas publicações em conferências [13] [14] e em revistas especializadas da área[16], [25].

**Algoritmo 4** CDAS-MOPSO

1: Procedimento de inicialização.
2: Definir as soluções não dominadas através do método CDAS usando valor $S_i$
3: Arquivamento das soluções não dominadas e escolha dos líderes.
4: Laço evolutivo
5: Para cada partícula $p_i$ da população
6: Atualizar a velocidade e população de $p_i$
7: Avaliar os objetivos e modificação da área de dominância de $p_i$ através do método CDAS usando valor $S_i$
8: Arquivamento de $p_i$
9: Escolha dos líderes para as partículas da população
10: Retornar as partículas do arquivo externo.

## 4.2 Explorando métodos de ranking em algoritmos MOPSO

A segunda etapa da exploração de novas relações de preferências em algoritmos MOPSO refere-se ao uso de métodos de ranking. Esses métodos buscam construir uma função de fitness através da associação dos valores do vetor de funções objetivo. Métodos de ranking foram inicialmente propostos na década de 90 em [4], porém só foram explorados em Problemas de Otimização com Muitos Objetivos em trabalhos mais recentes [23] [47] [40].

Nesta tese, os rankings são utilizados de três formas: primeiro é proposto um novo algoritmo MOPSO que utiliza métodos de rankings da literatura como função de fitness; a segunda contribuição é a proposta de um novo método chamado de *Balanced Ranking*; por fim, é proposta uma nova abordagem que visa combinar diferentes métodos de ranking através de um algoritmo MOPSO.

Essa seção discute as três abordagens propostas. Com esse intuito, inicialmente o modelo geral para a obtenção de ranking a partir do vetor de objetivos, bem como os métodos *Average* Ranking e *Maximum* Ranking são discutidos na Seção 4.2.1. O algoritmo Ranking-SMPSO é proposto na Seção 4.2.2. Por fim, o método Balanced Ranking é apresentado na Seção 4.2.3 e o algoritmo Combinação de Rankings na Seção 4.2.4.

### 4.2.1 *Average* ranking e *maximum* ranking

Os métodos *Average Ranking* (AR) e *Maximum Ranking* (MR) foram propostos em [4]. Essas técnicas são relações de preferência que induzem uma ordem de preferência para

um conjunto de soluções. Esses métodos são aplicados sobre um conjunto de soluções e definem um ordenamento entre elas.

O AR e o MR compartilham um modelo geral para geração de um ranking. Eles possuem o mesmo funcionamento, diferenciando-se somente no final do cálculo do ranking. Em geral, dado um conjunto de soluções $CS \subset \Omega$, um método de ranking computa um valor, $ranking(\overrightarrow{x}, CS)$ de cada solução $\overrightarrow{x} \in CS$. Assim, o primeiro passo é ordenar todas as soluções de acordo com cada função objetivo independentemente. Através desse ranking por função objetivo é possível seguir diferentes estratégias. O método AR define o ranking de uma solução $\overrightarrow{x}$ como sendo a soma de todos os rankings individuais da solução $\overrightarrow{x}$, enquanto o MR é definido pelo melhor ranking encontrado para $\overrightarrow{x}$. O AR e o MR podem ser definidos através das Equações (4.2) e (4.3), respectivamente.

$$AR(\overrightarrow{x}, CS) = \sum_{1<i<m} ranking(f_i(\overrightarrow{x}), CS) \tag{4.2}$$

$$MR(\overrightarrow{x}, CS) = MIN(ranking(f_i(\overrightarrow{x}), CS)) \tag{4.3}$$

onde $CS$ é um conjunto de soluções, $\overrightarrow{x}$ é a solução para qual está sendo calculado o ranking, $m$ é o número de objetivos e $ranking(f_i(\overrightarrow{x}), CS)$ é o ranking de $\overrightarrow{x}$ para o $i$-ésimo objetivo. A Tabela 4.1 apresenta um exemplo para ilustrar o funcionamento dos dois métodos. Por convenção, quanto menor é o ranking melhores são os valores dos objetivos. Inicialmente, cada solução é ordenada por cada objetivo, por exemplo, a solução $C$ possui o melhor ranking para o primeiro objetivo (ranking 1). Após o cálculo de cada ranking, os valores de cada método são calculados. O método AR soma todos os rankings de cada objetivo, enquanto o MR é definido pelo valor do melhor ranking. No exemplo da Tabela 4.1, percebe-se que soluções que possuem valores extremos, por exemplo $A$ e $D$, possuem um melhor valor do que uma solução com valores dos objetivos mais equivalentes, solução $B$. O MR age de forma semelhante que o AR, porém dá um peso maior ainda para as soluções com valores extremos. No exemplo da Tabela 4.1, todas as soluções com o melhor valor em pelo menos um objetivo recebe o melhor ranking possível. Para esse

Tabela 4.1: Exemplo do *Average* Ranking e *Maximum* Ranking.

| # | ($f_1$,$f_2$,$f_3$) | $f_1$ | $f_2$ | $f_3$ | **AR** | **MR** |
|---|---|---|---|---|---|---|
| A | (6, 1, 3) | 4 | 1 | 2 | 7 | 1 |
| B | (3, 2, 5) | 3 | 2 | 4 | 9 | 2 |
| C | (1, 11, 7) | 1 | 3 | 5 | 9 | 1 |
| D | (2, 20, 1) | 2 | 5 | 1 | 8 | 1 |
| E | (5, 14, 4) | 5 | 4 | 3 | 12 | 3 |

método, a solução $B$ fica atrás de três soluções no ranking.

Os métodos AR e MR apresentam bons resultados na literatura [23] [47]. Porém, apesar dos bons resultados, os métodos de rankings apresentam algumas limitações. Como discutido em [47], estes métodos privilegiam soluções que possuem bons rankings em poucos objetivos, normalmente um objetivo. Assim, essas soluções são localizadas nos extremos da fronteira de Pareto. Por exemplo, a solução B possui rankings médios para todos os objetivos, porém é superada por várias outras soluções, se considerarmos os rankings AR e MR. Esta limitação do AR e o MR é a motivação para a criação do método *Balanced Ranking* (BR), apresentado na Seção 4.2.3.

### 4.2.2 Ranking-SMPSO

O algoritmo Ranking-MOPSO incorpora métodos de ranking como função de fitness de um algoritmo MOPSO. Ele tem como base o algoritmo SMPSO [70]. Esse algoritmo é chamado de Ranking-SMPSO. A proposta desse algoritmo foi publicada em [25].

O Ranking-SMPSO utiliza uma relação de preferência diferente da dominância de Pareto. Para isto, é definida a relação $\prec_{Ranking}$, onde *Ranking* é o método de ranking escolhido para execução do algoritmo. A relação $\prec_{Ranking}$ é definida da seguinte maneira. Uma solução $\overrightarrow{x}$ é melhor que uma solução $y$ com respeito à relação *Ranking*, denotada por $\prec_{Ranking}$, se o somente se $Ranking(\overrightarrow{x}) < Ranking(\overrightarrow{y})$.

O algoritmo Ranking-SMPSO utiliza a relação de dominância $\prec_{Ranking}$ para melhorar os resultados do algoritmo SMPSO para problemas com muitos objetivos. O Algoritmo 5 apresenta uma breve descrição do Ranking-SMOPSO. Primeiro, um procedimento de inicialização é efetuado. Nesse procedimento, todos os componentes das partículas são

**Algoritmo 5** Ranking-SMPSO

1: Procedimento de inicialização.
2: Avaliar as partículas e cálculo do ranking.
3: Inicializar o repositório e escolha dos líderes.
4: Laço evolutivo
5: Atualizar a posição e velocidade das partículas
6: Avaliar os objetivos e cálculo dos rankings.
7: Mutação de 15% da população
8: Atualizar o repositório através da relação $\prec_{Ranking}$.
9: Calcular a distância de *crowding* (CD).
10: Arquivamento dos líderes e poda do arquivo externo(Melhor ranking e CD).
11: Seleção dos líderes.
12: Retornar as partículas do repositório.

iniciados (posição, velocidade, líder local, etc.). Após, todos os objetivos são avaliados e é calculado o ranking de cada partícula. Então, o repositório com as melhores soluções deve ser iniciado. Aqui, para efetuar uma pressão inicial em direção às melhores soluções, somente 10% das soluções com os melhores valores de ranking preenchem o repositório. Após este passo, é efetuada a escolha dos líderes através do método do torneio binário com distância de *crowding*.

O próximo passo do Ranking-SMPSO é o laço evolutivo. Nesse laço, primeiro as partículas são movimentadas pelo espaço de busca através do operador de velocidade. Em seguida é aplicado aleatoriamente um operador de mutação em 15% da população. O arquivamento das melhores soluções no arquivo externo é a principal diferença entre o Ranking-SMPSO e o SMPSO. Neste passo, a relação $\prec_{Ranking}$ é aplicada. Uma solução entra no repositório se o somente se ela é melhor pela relação $\prec_{Ranking}$ que qualquer outra partícula no repositório. Em seguida, a distância de *Crowding* (CD) [28] é calculada para todas as soluções do repositório. Como apresentando no Capítulo 3, o algoritmo SMPSO limita o número de soluções do repositório. Então, deve ser feita uma poda neste repositório. Este procedimento de poda mantém no repositório as soluções com os melhores valores de ranking. Se o valor de ranking for igual, então é utilizado o valor da CD para definir qual é a melhor solução. Por fim é feita a seleção dos líderes. Para evitar que soluções dominadas influenciem a busca para regiões ruins do espaço de objetivos, antes dessa etapa é feita a retirada de soluções dominadas do repositório.

### 4.2.3 *Balanced* ranking

Outra estratégia proposta neste trabalho é uma nova relação de preferência chamada de *Balanced Ranking* (BR). O BR tem como base o mesmo princípio adotado pelos métodos AR e MR propostos por [4], porém busca evitar a geração de soluções nos extremos da fronteira de Pareto [47]. A Figura 2.1 mostra um exemplo de uma fronteira de Pareto em um espaço de objetivos bidimensional com os valores extremos e o joelho da fronteira de Pareto.

O método proposto privilegia soluções em que os objetivos possuam valores semelhantes, ou seja, privilegiar soluções que não contenham valores extremos em relação aos demais e sejam mais próximas ao joelho. Os passos iniciais para o cálculo do BR são os mesmos necessários para o cálculo do AR e do MR. Assim, dado um conjunto $CS$ de soluções, o BR de uma solução $\overrightarrow{x}$ é calculado da seguinte maneira. Primeiro, é calculado um ranking independente para cada função objetivo. Para o cálculo do ranking é tomada a convenção de quanto melhor é a solução, mais baixo é o ranking. Após, é feita a soma de todos estes rankings. Em seguida, é calculada uma ponderação que é obtida através da diferença entre o maior e o menor ranking de uma solução. Essa diferença, é normalizada pelo tamanho de $CS$ (o maior ranking possível), assim, quanto maior a diferença entre os rankings, pior será o BR da solução e quanto menor a diferença, melhor será o BR. O BR pode ser definido através da Equação (4.4).

$$BR(\overrightarrow{x}, S) = \frac{MAX_{ranking}(x) - MIN_{ranking}(\overrightarrow{x})}{|CS|} * \sum_{1<j<m} ranking(f_j(\overrightarrow{x}), S) \qquad (4.4)$$

onde $m$ é o número de objetivos, $\overrightarrow{x}$ a solução a ser ranqueada, $CS$ o conjunto de soluções e $ranking(f_j(\overrightarrow{x}), CS)$ é o raking de $\overrightarrow{x}$ para o $j$-ésimo objetivo. A Tabela 4.2 apresenta um exemplo do funcionamento do BR. É apresentado também o cálculo do AR e do MR. Primeiro, cada solução é rankeada para cada objetivo, por exemplo, a solução $C$ possui o melhor ranking (ranking 1) para o primeiro objetivo. Após, é obtido para cada

Tabela 4.2: Exemplo do *balanced* ranking.

| # | ($f_1$,$f_2$,$f_3$) | **ranking**($f_1$) | **ranking**($f_2$) | **ranking**($f_3$) | **MAX** | **MIN** | **BR** | **AR** | **MR** |
|---|---|---|---|---|---|---|---|---|---|
| A | (9, 1, 3) | 5 | 1 | 2 | 5 | 1 | 5,3 | 8 | 1 |
| B | (3, 2, 5) | 3 | 2 | 4 | 4 | 2 | 3 | 9 | 2 |
| C | (1, 7, 7) | 1 | 4 | 5 | 5 | 1 | 6,66 | 10 | 1 |
| D | (2, 8, 1) | 2 | 5 | 1 | 5 | 1 | 5,33 | 8 | 1 |
| E | (7, 5, 8) | 4 | 3 | 6 | 6 | 3 | 7 | 14 | 3 |
| F | (10, 9, 4) | 6 | 6 | 3 | 6 | 3 | 7,5 | 15 | 3 |

solução os valores máximo e mínimo. Por fim, é feita a soma dos rankings e esta soma é ponderada pela diferença dos rankings normalizada. Através das duas primeiras soluções pode-se perceber a diferença entre o AR e o BR. Utilizando o AR a melhor solução é a $A$, pois possui um AR igual a 8. Porém, percebe-se que ela possui o melhor ranking para um objetivo (ranking 1 para o segundo objetivo), mas apresenta um ranking ruim para outros (ranking 5 para o primeiro objetivo). Utilizando o BR essa diferença é penalizada e a melhor solução passa ser a $B$, que apesar de não ser melhor em nenhum objetivo possui rankings semelhantes em todos.

### 4.2.4 Combinação de rankings

A última estratégia proposta é chamada de Combinação de Rankings e visa juntar aspectos positivos de diferentes esquemas de ranking. Como discutido anteriormente, métodos como o AR e o MR tendem a gerar soluções com um objetivo com melhores valores que os demais, assim, o conjunto de aproximação gerado tende a ser localizado nos extremos da fronteira de Pareto. Em contrapartida, o método proposto na seção anterior, BR, privilegia soluções com valores dos objetivos semelhantes e assim evita a geração do conjunto de aproximação nos extremos. Porém, os dois métodos apresentam deficiências, pois em algumas situações é preferível que o conjunto de aproximação fique próximo ao joelho [47], mas também é importante obter diversidade sobre toda a fronteira de Pareto.

A Combinação de Rankings utiliza uma ideia de reduzir um número grande de objetivos para poucos objetivos, normalmente 2 ou 3. No método proposto, os objetivos de uma solução passam a ser os valores de diferentes rankings escolhidos. Os passos da Combinação de Rankings são (ver Figura 4.6): dada uma solução $\overrightarrow{x}$ e o seu vetor de objetivos $\overrightarrow{f(\overrightarrow{x})} = f_1(\overrightarrow{x}), f_2(\overrightarrow{x})..., f_m(\overrightarrow{x}))$, onde $m$ é o número de objetivos; calcular o

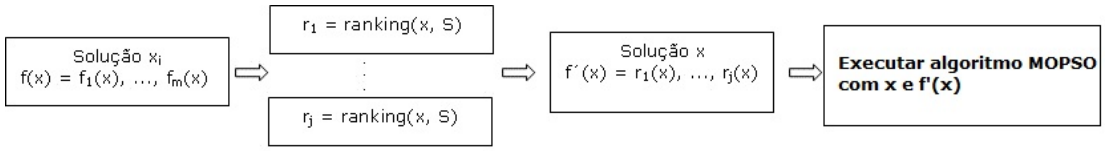



Figura 4.6: Esquema do método combinação de rankings.

valor de cada ranking $r_j = ranking_j(\overrightarrow{x}, CS)$, onde $CS$ é o conjunto de soluções, $ranking_j$ é um método de raking e $2 \leq j \leq 3$ é o número de rankings escolhidos. É importante que o número de rankings não seja maior do que 3 para que não haja deterioração na busca do algoritmo; definir os valores dos rankings como objetivos de $\overrightarrow{x}$, $\overrightarrow{f'(\overrightarrow{x})} = r_1, ..., r_j$; executar iteração do algoritmo evolutivo multiobjetivo com os novos valores $\overrightarrow{f'(\overrightarrow{x})}$.

Por exemplo, utilizando os rankings AR e BR, pode-se executar um MOEA tradicional (NSGA-II [28], SMPSO [70]) e otimizar tanto um objetivo que gere soluções nos extremos, quanto um objetivo que gere soluções no centro da fronteira de Pareto. A hipótese da Combinação de Rankings é que juntando diferentes rankings como vetor objetivo de uma solução pode-se obter um melhor conjunto de aproximação do que a aplicação de cada método separado. Nesta tese, a técnica da Combinação de Rankings é aplicada ao algoritmo SMPSO. A validação desse método é discutida no Capítulo 7.

## 4.3 Considerações finais

A definição de novas relações de preferência é um dos caminhos utilizados por alguns trabalhos na literatura para reduzir a deterioração de MOEAs na Otimização com Muitos objetivos [23] [47] [81]. Em geral, os métodos propostos na literatura apresentam bons resultados, porém limitam-se a estratégias baseadas em abordagens evolucionárias.

Nesta tese duas novas relações de preferências são aplicadas em algoritmos MOPSO: a técnica CDAS [81] e a exploração de métodos de ranking [23]. Com esse objetivo, novos algoritmos MOPSO são propostos buscando obter bons resultados em Problemas de Otimização com Muitos Objetivos. Os algoritmos CDAS-MOPSO [16] e Ranking-SMPSO [25] incorporam essas novas relações de preferência e modificam aspectos de um algoritmo MOPSO, como a escolha e a seleção do conjunto de líderes. Além dos novos

algoritmos se destaca também a proposta de novas abordagens baseadas em ranking: método *Balanced* Ranking e a Combinação de Rankings.

No Capítulo 7 é apresentado um conjunto de experimentos que avalia os métodos propostos utilizando-se problemas com muitos objetivos. Além disso, também e apresentado um conjunto de experimentos que compara o desempenho dos algoritmos CDAS-MOPSO e do Ranking-SMPSO.

# CAPÍTULO 5

# EXPLORANDO MÉTODOS DE ARQUIVAMENTO

Em problemas de otimização multiobjetivo não há somente uma melhor solução, mas sim um conjunto de soluções. Logo, a metaheurística da Otimização por Nuvem de Partículas Multiobjetivo deve definir quais partículas irão guiar o enxame. Nesse contexto, a escolha e a seleção dos líderes são importantes características de um algoritmo MOPSO. Os métodos para definir quais soluções não dominadas serão líderes candidatos e como cada partícula irá escolher o seu líder global influenciam características como a convergência e a diversidade da busca do algoritmo.

Conforme discutido no Capítulo 3, existem diferentes métodos para a escolha do líder por cada partícula do enxame. Em geral, esses métodos definem um critério, por exemplo, escolher um líder numa região menos povoada, e percorrem os líderes candidatos com o intuito de encontrar o líder mais habilitado para a partícula. Para isso, esses métodos efetuam outros cálculos com o conjunto de líderes, como o cálculo da distância de *Crowding* [28], cálculo dos vetores sigma [67], entre outros. Porém, se esse conjunto de líderes for grande, a busca de um líder por uma partícula pode ficar tão complexa quanto a própria resolução do problema.

Para evitar que o conjunto de líderes influencie negativamente na eficiência do algoritmo MOPSO, esse conjunto de líderes é normalmente limitado com um tamanho máximo $N$ [64]. Isto é, o arquivo externo possui tamanho máximo $N$. Devido a esse limite, possíveis líderes serão descartados pelo algoritmo quando o arquivo estiver cheio. Nesse contexto, métodos de arquivamento se tornam importantes por disponibilizar o melhor conjunto de líderes possíveis. Além disso, as soluções presentes no arquivo serão a solução do problema.

O Capítulo 3 apresentou um conjunto de métodos de arquivamento que podem ser utilizados em algoritmos MOPSO. Esses métodos buscam definir um conjunto de líderes com

características específicas, tal como privilegiar soluções mais espalhadas pelo espaço de objetivos ou selecionar apenas soluções em regiões predeterminadas. Este capítulo apresenta os métodos de arquivamento das soluções não dominadas propostos nesta tese. O objetivo principal da proposta desses métodos é obter bons resultados com um algoritmo MOPSO em problemas com muitos objetivos. Os métodos propostos exploram diferentes abordagens: uma abordagem que utiliza o vetor ideal para introduzir maior convergência na busca, chamada de Arquivador Ideal; duas outras abordagens que buscam espalhar as soluções do arquivo externo utilizando como base as soluções nos extremos de cada eixo de coordenadas, chamadas de Arquivador Distribuído e Arquivador Distribuído pela Busca; e uma abordagem que utiliza um ponto de referência escolhido pelo usuário, chamada de Arquivador Hiperplano. Todos os arquivadores propostos seguem o esquema do Arquivador Preciso, descrito no Algoritmo 2 do Capítulo 3.

Esse capítulo está organizado da seguinte maneira: o Arquivador Ideal é descrito na Seção 5.1. Os dois arquivadores que utilizam como base as soluções nos extremos, Arquivador Distribuído e Arquivador Distribuído pela Busca são apresentados na Seção 5.2. Em seguida, o Arquivador Hiperplano é discutido na Seção 5.3 e por fim, a Seção 5.4 apresenta as considerações finais desse capítulo.

## 5.1 Arquivador ideal

O Arquivador Ideal tem como objetivo introduzir maior convergência na busca de um algoritmo MOPSO. Para isso, esse método utiliza a hipótese de que concentrar as soluções em uma região específica do espaço de objetivos introduz maior convergência em direção à fronteira de Pareto. A região escolhida nesse método é definida através do vetor ideal. O vetor ideal é um vetor utópico e corresponde a um vetor com os melhores valores de cada função objetivo [20].

Esse método privilegia as soluções do arquivo que estejam mais próximas do vetor ideal no espaço de objetivos [20]. De forma simples, o Arquivador Ideal seleciona as soluções mais próximas ao ponto ideal como consequência o conjunto de aproximação gerado pelo algoritmo é mais próximo da fronteira de Pareto (maior convergência) e as soluções se

**Algoritmo 6** Função $filtro(A^{t-1}, x)$ do arquivador ideal.

1: Adicionar $\overrightarrow{x}$ a $A^{t-1}$ $(A^{t-1} \cup \overrightarrow{x})$
2: Calcular o vetor ideal das soluções em $A^{t-1} \cup \overrightarrow{x}$
3: Calcular a distância euclidiana entre o vetor de objetivos das soluções em $A^{t-1} \cup \overrightarrow{x}$ e o vetor ideal
4: Remover a solução com maior distância Euclidiana

localizam em uma boa área do espaço de objetivos (próximo ao joelho da fronteira).

O Arquivador Ideal segue o esquema no Arquivador Preciso (Algoritmo 2) [22]. Nesse modelo, cada método de arquivamento define uma função que é aplicada quando o arquivo está cheio e não há espaço para uma nova solução não dominada $\overrightarrow{x}$. O arquivador deve decidir quais das $N + 1$ soluções do conjunto $A^{t-1} \cup \overrightarrow{x}$ devem permanecer no arquivo externo, através da função $filtro(A^{t-1}, \overrightarrow{x})$. A função $filtro(A^{t-1}, \overrightarrow{x})$ do Arquivador Ideal é apresentada no Algoritmo 6. Assim o arquivo externo possui somente as soluções mais próximas ao vetor ideal. Nesse método, inicialmente a nova solução $\overrightarrow{x}$ é adicionada ao arquivo externo. Em seguida, é calculado o vetor ideal a partir dos vetores objetivos das soluções no arquivo. Após a obtenção do vetor ideal, é calculada a distância entre todos os vetores objetivos e o vetor ideal. Por fim, a solução com maior distância é removida do arquivo.

A Figura 5.1 mostra um exemplo do Arquivador Ideal num espaço de objetivos de duas dimensões. Os pontos verdes indicam as soluções do conjunto $(A^{t-1} \cup \overrightarrow{x})$. O vetor ideal é obtido com o melhor valor para cada função objetivo. Após, a solução mais distante desse vetor, no espaço de objetivos, é removida. A idea do Arquivador Ideal é remover as soluções que estejam longe do ponto ideal e aumentar a convergência guiando a busca para uma área menor do espaço de objetivos. É desejado que essa área seja localizada próximo ao joelho da fronteira de Pareto.

## 5.2 Arquivador distribuído e arquivador distribuído pela busca

O segundo conjunto de arquivadores desenvolvido também tem como objetivo guiar a busca para alguma região da fronteira de Pareto, porém, diferentemente do Arquivador Ideal, são escolhidas diferentes regiões definidas por um conjunto de pontos. Com esse

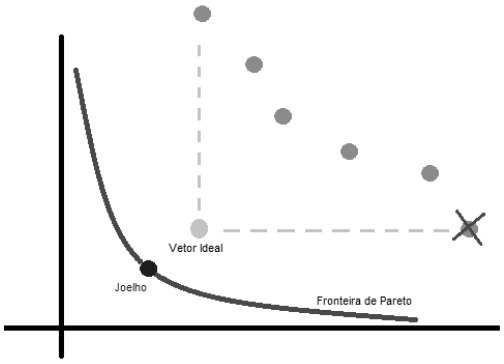



Figura 5.1: Arquivador ideal.

objetivo, são propostos o Arquivador Distribuído e o Arquivador Distribuído pela Busca. Esses métodos focam na convergência através de um conjunto de pontos de referência, porém para não perder diversidade seleciona esses pontos em diferentes áreas do espaço de objetivo. Os pontos de referência usados em ambos os métodos são os pontos extremos de cada dimensão e o vetor ideal (um conjunto de $m + 1$ pontos).

A estratégia do Arquivador Distribuído é fazer uma distribuição igualitária das soluções do arquivo entre os pontos de referência. Ele também segue o padrão do Arquivador Preciso e a função $filtro(A^{t-1}, \overrightarrow{x})$ é apresentada no Algoritmo 7. Nessa função os $A^{t-1} \cup \overrightarrow{x}$ pontos devem ser distribuídos para cada ponto de referência, e o ponto que estiver mais longe desses pontos de referência é removido. Para efetuar a distribuição, inicialmente são obtidos os pontos de referência: os pontos nos extremos de cada eixo de coordenadas (conjunto de pontos $E_j$) e o vetor ideal, calculado entre as soluções do arquivo. Após, é definido o número de soluções que são alocadas para cada ponto de referência. Esse número é igual a $num\_ref = N/(m + 1)$, o tamanho do arquivo dividido pelo número de pontos de referência. Caso a divisão não seja exata, o ponto de referência ideal recebe um número maior de soluções que os demais (o resto da divisão).

Após a definição dos pontos de referência é feita a distribuição das soluções do arquivo nos pontos de referência. Para evitar que a seleção dos pontos tenda em direção de alguma dimensão, o algoritmo percorre os pontos de referência de forma aleatória. Então, aleatoriamente é escolhido um ponto de referência. Em seguida é calculada a distância euclidiana entre os vetores objetivo das soluções do arquivo e esse ponto de referência.

**Algoritmo 7** Função $filtro(A^{t-1}, x)$ do Arquivador Distribuído

1: Adicionar $\overrightarrow{x}$ a $A^{t-1}$ ($A^{t-1} \cup \overrightarrow{x}$) e criar um arquivo temporário ($A^{temp}$)
2: Calcular o vetor ideal das soluções em $A^{t-1} \cup \overrightarrow{x}$ e os pontos extremos de cada dimensão
3: $num\_ref = N/(m+1)$
4: Enquanto `existir ponto de referência ainda não considerado`
5: Escolher ponto de referência aleatoriamente
6: Calcular a distância euclidiana entre o vetor de objetivos das soluções em $A^{temp} \cup x$ e o ponto de referência
7: $mais\_proximas\_referencia = num\_ref$ soluções com menor distância Euclidiana
8: Adicionar as $mais\_proximas\_referencia$ com menor distância a $A^t$
9: Remover $mais\_proximas\_referencia$ de $A^{temp}$

Em seguida, são adicionadas ao arquivo as $num\_ref$ soluções mais próximas ao ponto de referência. Ao final do laço o próximo ponto de referência é escolhido. O algoritmo efetua esse laço para todos os pontos de referência e ao final o arquivo está definido, sendo a solução não escolhida é descartada.

Arquivador Distribuído pela Busca é semelhante ao Arquivador Distribuído, porém não fixa o número de pontos para cada ponto de referência. Os pontos de referência utilizados são os mesmos, os pontos nos extremos e o vetor ideal entre as soluções do arquivo. Nesse método, ao invés de fixar a escolha das soluções do arquivo de forma distribuída para cada ponto de referência, essa distribuição varia durante a busca. O Algoritmo 8 apresenta a função $filtro(A^{t-1}, \overrightarrow{x})$ desse arquivador. Os primeiros passos são a definição dos pontos candidatos ao arquivamento $A^{t-1} \cup \overrightarrow{x}$ e a obtenção dos pontos de referência. Em seguida, é calculada a distância entre todos os pontos do conjunto $A^{t-1} \cup \overrightarrow{x}$ e todos os pontos de referência. Para cada ponto em $A^{t-1} \cup \overrightarrow{x}$ é associada uma distância ($dist_{ref}$) que é a menor distância encontrada entre o vetor de objetivos de $\overrightarrow{x}$ e um ponto de referência. Então, é descartada a solução que possui maior ($dist_{ref}$). Esse método tem como objetivo identificar quais soluções estão mais próximas dos pontos de referência e então guiar a seleção do arquivo para essas regiões do espaço de objetivo. Para esse método são utilizados dois métodos de cálculo de distância: distância Euclidiana e distância de Tchebycheff.

**Algoritmo 8** Função $filtro(A^{t-1}, x)$ do Arquivador Distribuído pela Busca

1: Adicionar $\overrightarrow{x}$ a $A^{t-1}$ ($A^{t-1} \cup \overrightarrow{x}$)
2: Calcular o vetor ideal das soluções em $A^{t-1} \cup \overrightarrow{x}$ e os pontos extremos de cada dimensão
3: Calcular a distância entre todos os pontos de referência e cada ponto em $A^{t-1} \cup \overrightarrow{x}$
4: Para cada ponto em $A^{t-1} \cup \overrightarrow{x}$ definir $dist_{ref}$ como a menor distância encontrada
5: Remover a solução com maior $dist_{ref}$

## 5.3 Arquivador hiperplano

Esse método de arquivamento utiliza um ponto de referência escolhido pelo usuário com o objetivo de guiar a busca do algoritmo para uma região próxima a este ponto. O Arquivador Hiperplano define o ponto de referência através de um conjunto de pontos, escolhido pelo usuário através de um hiperplano, baseado na estratégia inicialmente proposta em [27]. Diferentemente dos arquivadores discutidos anteriormente, ao invés de um conjunto de pontos nos extremos e somente o ponto ideal, esse hiperplano define um conjunto de pontos que cobrem uma área maior do espaço de objetivos. Esse hiperplano é construído de forma automática, sendo necessários somente pontos que definem os extremos e um valor que define o espaçamento entre os pontos de referência. A figura 5.2 apresenta um exemplo de hiperplano com 51 pontos de referência para 3 funções objetivo, os pontos extremos utilizados foram: $(1, 0, 0)$, $(0, 1, 0)$ e $(0, 0, 1)$.

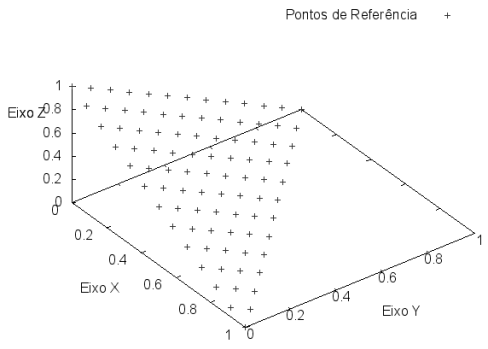



Figura 5.2: Hiperplano com 51 pontos de referência.

Assim, o Arquivador Hiperplano utiliza um ponto de referência, escolhido através de um hiperplano, e efetua a seleção das soluções no arquivo. O Algoritmo 9 apresenta a função $filtro(A^{t-1}, \overrightarrow{x})$ do Arquivador Hiperplano. Esse método é igual ao algoritmo do

**Algoritmo 9** Função $filtro(A^{t-1}, x)$ do Arquivador Hiperplano

1: Adicionar $\overrightarrow{x}$ a $A^{t-1}$ ($A^{t-1} \cup \overrightarrow{x}$)
2: Obtém o ponto de referência do hiperplano
3: Calcular a distância euclidiana entre o vetor de objetivos das soluções em $A^{t-1} \cup \overrightarrow{x}$ e o ponto de referência
4: Remover a solução com maior distância Euclidiana

Arquivador Ideal, a única diferença é na seleção do ponto de referência.

O Arquivador Hiperplano é definido através da construção do hiperplano e da escolha do ponto de referência. Logo, diferentes abordagens podem ser utilizadas: construir o hiperplano no começo da busca ou construir o hiperplano a cada execução do arquivador.

A primeira abordagem constrói o hiperplano no começo da busca, utilizando valores nos extremos pré-definidos. Para essa abordagem, deve se ter um conhecimento do problema, porém já se constrói um bom conjunto de referência desde início. Com isso, pode-se aumentar a convergência do algoritmo e pode-se direcionar a busca para regiões predefinidas. Nessa abordagem, pode-se escolher o ponto de referência utilizado para guiar o arquivamento de diferentes formas:

- **Ponto pré-definido**: Nesse método o ponto é escolhido pelo usuário no começo da execução guiando a busca para uma região desejada. Utilizando o hiperplano é possível definir um conjunto de pontos de referência no espaço de objetivos e então escolhem um dos pontos como guia. A construção do hiperplano é importante. pois limita a escolha do ponto de referência a uma região promissora.

- **Ponto definido por regiões**: Outra possibilidade é a escolha de regiões do hiperplano ao invés de somente um ponto referência. Ao invés da definição de um ponto no começo da busca, é possível escolher uma região no hiperplano para onde se deseja que o algoritmo convirja. Assim, é escolhido um subconjunto de pontos localizados em regiões específicas, tal como, o centro do hiperplano ou uma região próxima a um extremo. Dado esse subconjunto de pontos o arquivador escolhe um ponto aleatoriamente e utiliza-o como referência na seleção do arquivo.

Na segunda abordagem o hiperplano é construído a cada iteração, obtendo-se os pontos

extremos entre os pontos presentes no arquivo. Nesse caso a escolha pode ser feita através de duas possibilidades:

- **Ponto definido por regiões**: Da mesma forma que na outra abordagem é possível escolher uma região do hiperplano. Mesmo que não se tenha conhecimento prévio sobre hiperplano usado em cada execução do hiperplano, é possível escolher regiões com o intuito que ao final da busca as soluções estejam localizadas em regiões semelhantes da fronteira de Pareto.

- **Ponto escolhido aleatoriamente**: É a abordagem mais simples. O ponto de referência é escolhido aleatoriamente entre todos os pontos do hiperplano, a cada execução. Nessa abordagem o arquivador privilegia diversidade ao invés de convergência.

O Arquivador Hiperplano é validado nos experimentos conduzidos no Capítulo 7, porém, nesta tese somente é explorada a construção do hiperplano no começo da busca com escolha dos pontos definidos por regiões.

## 5.4 Considerações finais

Os métodos de arquivamento são importantes na busca de algoritmos multiobjetivo, em especial em algoritmos MOPSO. Nesse contexto, o arquivo externo define quais são os líderes candidatos para as partículas e também define quais são as soluções resultantes da busca. Porém, em geral, esse arquivo externo é limitado para que não haja prejuízo no desempenho da busca. Devido a esse limite, a maneira que serão selecionados os possíveis líderes no arquivo externo influencia características como convergência ou diversidade da busca.

Assim, métodos de arquivamento podem ser explorados visando atenuar os problemas da Otimização com Muitos Objetivos. Uma hipótese que pode ser explorada é limitar a região do espaço de objetivo em que os líderes podem estar localizados. Esse caminho explora a ideia de que limitar a região do espaço de busca dos líderes aumenta a

convergência das partículas em direção à fronteira de Pareto. Porém, essa limitação do espaço deve definir boas regiões da fronteira, como pontos próximos ao joelho ou regiões pré-definidas pelo usuário.

Visando explorar essas ideias foram propostos os métodos Arquivador Ideal, Arquivador Distribuído, Arquivador Distribuído pela Busca e Arquivador Hiperplano. Esses métodos guiam a busca através da seleção de bons líderes. Eles definem métodos que privilegiam convergência, outros que privilegiam diversidade e métodos que guiam a busca para regiões da fronteira definidas pelo usuário. Os métodos de arquivamento discutidos nesse capítulo foram inicialmente propostos nas publicações [9] e [7] do autor.

# CAPÍTULO 6

# EXPLORANDO MÚLTIPLOS ENXAMES E MÉTODOS DE ARQUIVAMENTO

Diferentes técnicas têm sido propostas na Otimização com Muitos Objetivos [46]. A maior parte dessas técnicas visa reduzir a deterioração na busca dos algoritmos multiobjetivo. Um dos caminhos seguidos por essas técnicas constitui na definição de uma nova ordenação das soluções não dominadas, com o intuito de reduzir o tamanho deste conjunto. Em geral, esses métodos definem regiões de preferência no espaço de objetivo ou ordenam a solução de acordo com as diferentes funções objetivo.

Porém, como discutido em [1], na maioria das técnicas só uma pequena parte da fronteira de Pareto é coberta, não havendo diversidade. Esse é o caminho dos métodos propostos nos capítulos anteriores, bem como diversos outros métodos apresentados na literatura como em [47], [40], [77], [81], entre outros.

Focando o aumento da diversidade na busca, sem perder a convergência, este capítulo explora algoritmos MOPSO com múltiplos exames. A ideia é que dividir a busca através de diferentes *sub-swarms* rodando em paralelo reduz a complexidade da busca em problemas menores [36]. Além disso, é possível definir que cada enxame execute a busca em diferentes regiões do espaço de busca com o objetivo de aumentar a diversidade e a distribuição do conjunto de aproximação gerado. Os múltiplos enxames são explorados de diferentes maneiras para problemas multi-objetivo. Pode-se definir que cada *sub-swarm* irá otimizar um objetivo ou um subconjunto de objetivos [73] ou otimizar todas as funções objetivo em cada *sub-swarm* e utilizar múltiplos enxames para reduzir a complexidade da busca em outros aspectos [76], por exemplo utilizar menos partículas em cada enxame.

Os algoritmos *multi-swarm* desenvolvidos neste capítulo seguem o esquema de otimizar todo o vetor de funções objetivo em cada enxame. Os múltiplos enxames são utilizados para compor diferentes métodos de arquivamento, com o objetivo de obter bons resultados

em termos de convergência e diversidade. Esses algoritmos utilizam métodos de arquivamento apresentados nos capítulos anteriores. Como discutido na Seção 3.2.1, existem diferentes arquivadores propostos na literatura e em geral estes arquivadores apresentam características específicas, como privilegiar diversidade ou convergência.

A principal ideia é executar a busca em diferentes populações, cada uma utilizando um método de arquivamento diferente, buscando otimizar a convergência e a diversidade da busca através da comunicação entre os *sub-swarms*. A hipótese é que ao combinar métodos de arquivamento através de algoritmos *multi-swarm*, obtém-se um melhor conjunto de aproximação que o uso individual de cada arquivador. É importante ressaltar que os algoritmos propostos aqui não exploram conceitos de programação paralela ou distribuída. No algoritmo, cada *sub-swarm* é executado separadamente de forma sequencial e não há preocupação com o sincronismo ou troca de mensagens.

Através desse conceito três algoritmos *multi-swarm* são propostos: *Multi-Swarm* Baseado em Arquivadores, *Multi-Swarm* com Arquivo Externo Compartilhado e *Iterated Multi-Swarm*, I-Multi. O *Multi-Swarm* Baseado em Arquivadores combina diferentes métodos de arquivamento e troca de informações entre os *sub-swarms*. O *Multi-Swarm* com Arquivo Externo Compartilhado executa a comunicação entre os *sub-swarms* através de um arquivo externo compartilhado. O algoritmo I-Multi combina duas estratégias: compor convergência e diversidade através de arquivadores e utilizar *sub-swarms* para explorar diferentes regiões do espaço de busca. Os algoritmos apresentados nesse capítulo foram desenvolvidos em parceria com a professora Sanaz Mostaghim, durante o período de doutorado sanduíche do autor no Karlsruhe Institute of Technology, Karlsruhe, Alemanha.

Neste trabalho, todos os algoritmos utilizam como base o SMPSO [70]. Cada algoritmo possui população e arquivo externo independentes, bem como os demais parâmetros do MOPSO (equação de velocidade, escolha dos líderes, etc.). O método de arquivamento varia em cada método proposto, pois esse é o objetivo dos algoritmos explorados neste capítulo.

Este capítulo está organizado da seguinte forma: os algoritmos propostos *Multi-Swarm*

Baseado em Arquivadores, *Multi-Swarm* com Arquivo Externo Compartilhado e *Iterated Multi-Swarm* são apresentado na Seções 6.1, 6.2 e 6.3, respectivamente. Apesar desta tese propor três novos algoritmos baseados em múltiplos enxames, somente o I-Multi será validado através de uma análise empírica no próximo capítulo. A avaliação dos demais algoritmos será abordada em trabalhos futuros. Por fim, as considerações finais são discutidas em 6.4.

## 6.1 *Multi-Swarm* baseado em arquivadores

O primeiro algoritmo proposto é chamado de *Multi-Swarm* Baseado em Arquivadores e propõe uma estratégia simples para compor diferentes métodos de arquivamento através de múltiplos enxames. Os artigos [9] e [64] discutem diversos métodos de arquivamento e mostram que uns métodos introduzem maior diversidade na busca, enquanto outros introduzem maior convergência. A discussão sobre métodos de arquivamento também é explorada nesta tese nos Capítulos 5 e 7.

Assim, o objetivo desse algoritmo *multi-swarm* é combinar diferentes métodos de arquivamento através de diferentes populações que se comunicam durante a busca. Para que haja uma melhora nos resultados de cada arquivador, o algoritmo busca combinar métodos que apresentam bons resultados em termos de convergência com outros métodos que possuem bons resultados em termos de diversidade.

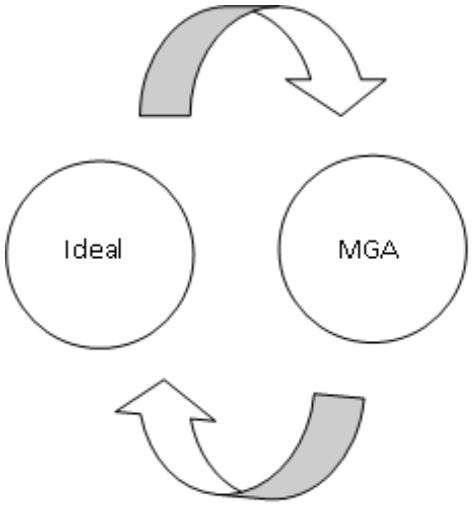
O algoritmo *Multi-Swarm* Baseado em Arquivadores é definido por um conjunto de *sub-swarms*, cada uma sendo executada em paralelo e utilizando um método de arquivamento. Em um determinado momento da execução há troca de informações entre cada *sub-swarm*. A comunicação entre as diferentes populações é um ponto importante do algoritmo. Nesse ponto deve ser definida a topologia que define a vizinhança entre cada *sub-swarm* e deve ser definido que informação deve ser trocada entre as populações. Ao final do algoritmo, o conjunto de aproximação gerado pelo algoritmo é a união das soluções não dominadas

Para o algoritmo proposto foi escolhida a topologia em anel. Como o objetivo é compor arquivadores diferentes, a topologia em anel é utilizada para permitir que haja a

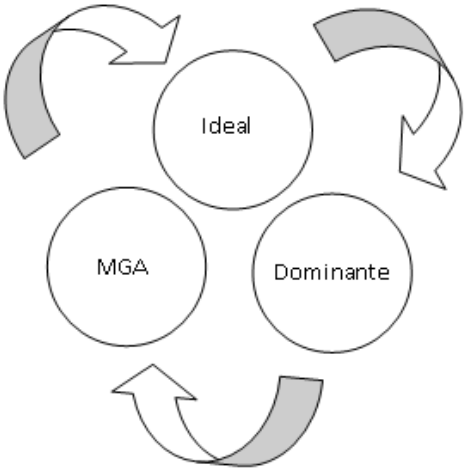
troca de informação entre um *sub-swarm* que privilegia convergência com um *sub-swarm* que otimiza diversidade. Baseado nos resultados apresentados num trabalho prévio do autor em [9] e na análise dos métodos de arquivamento apresentada no Capítulo 7, três métodos de arquivamento foram selecionados. O Arquivador Ideal apresenta os melhores resultados em termos de convergência, enquanto os métodos Arquivador Dominante e Arquivador MGA produziram uma fronteira de Pareto aproximada com maior diversidade. A Figura 6.1 mostra exemplos de diferentes composições de arquivadores através de múltiplos enxames numa topologia em anel. O número de enxames e o arquivador de cada *sub-swarm* são parâmetros do algoritmo. É importante ressaltar que esse algoritmo busca sempre maximizar a troca de informações entre diferentes tipos de arquivamento. É mais interessante que a informação passe entre diferentes tipos de arquivadores para que haja um ganho tanto de convergência quanto de diversidade.

Outra característica do algoritmo refere-se a troca de informação entre os *sub-swarm*. Nessa situação, duas questões a serem respondidas são: que informações trocar? Quando efetuar essa troca?

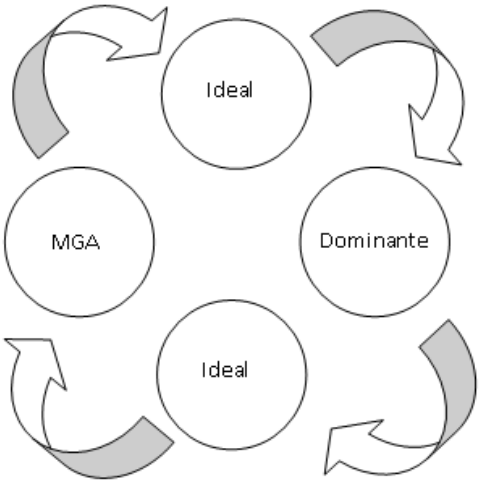
Em relação ao que os enxames devem trocar há duas possibilidades, a troca de partículas da população ou a troca de soluções do arquivo externo. Como o algoritmo proposto explora os métodos de arquivamento, é escolhido que deve haver a troca das soluções do arquivo externo. Nesse cenário, a população de cada *sub-swarm* permanece a mesma durante a busca, o que é alterado é o conjunto dos líderes candidatos. Esse conjunto de operações focam na diversidade e na convergência em diferentes momentos da busca. Assim, em um determinado momento da busca, dado um conjunto de soluções que foram selecionadas pelo Arquivador Ideal, a tendência é que essas soluções estejam concentradas em uma pequena região do espaço de objetivos, porém já próxima à fronteira de Pareto. Quando há a troca de informações entre *sub-swarms* esse conjunto passa para um arquivador que estimula diversidade (MGA ou Dominante). Com o desenvolvimento da busca algumas soluções do arquivo serão sobrepostas por novas soluções não dominadas que se localizam em regiões menos povoadas e é efetuada uma maior distribuição das soluções no arquivo. O inverso, isto é, passar as soluções de um arquivador que privilegia




(a) Anel com dois *sub-swarms*




(b) Anel com três *sub-swarms*




(c) Anel com quatro *sub-swarms*

Figura 6.1: Exemplo da composição de diferentes métodos de arquivamento através de múltiplos enxames numa topologia em anel.

diversidade para um que privilegia convergência, também é verdade, as soluções do arquivo vão perder diversidade mas vão chegar mais próximo da fronteira. Outra questão refere-se a quais soluções serão repassadas. Nesse caso há a possibilidade de se trocar todo o conjunto ou somente uma seleção de soluções. No algoritmo proposto é escolhida a troca de todo conjunto para que haja uma maior interação entre os arquivadores.

No processo de troca de informações entre cada *sub-swarm* o momento em que essa troca ocorre possui grande importância. No algoritmo *Multi-Swarm* Baseado em Arquivadores esse momento é definido através das iterações do algoritmo. Se o número de iterações for pequeno, pode não haver influência do arquivador no novo conjunto de soluções e portanto pode não existir nem convergência e nem diversidade. Por outro lado, se o número de iterações for grande a troca de informações entre os *sub-swarms* não tem efeito. Como estamos buscando a composição de diferentes arquivadores, o número de iterações para a troca de informação entre os *sub-swarms* deve ser otimizado.

O Multi-Swarm Baseado em Arquivadores é apresentado no Algoritmo 10. Além dos parâmetros básicos do SMPSO (ver Seção 3.3.1), o algoritmo tem como entrada o número de *swarms*, a lista dos métodos de arquivamento utilizado em cada *sub-swarm*, o número de iterações necessário para os swarms trocarem informações e o número de elementos na população e no arquivo externo de cada *sub-swarm*. O primeiro passo é a inicialização de cada *sub-swarm*. Após isso, o laço evolutivo é efetuado, até que uma condição seja alcançada. Esse algoritmo utiliza como condição um número máximo de iterações definido pelo usuário. Nesse laço, inicialmente cada *sub-swarm* efetua uma iteração do algoritmo PSO, movimentando as partículas e executando o arquivamento. Ao final do movimento das partículas é checado se a iteração corrente é uma iteração de troca. Caso sim, cada *sub-swarm* passa o seu arquivo para o próximo *sub-swarm*, sendo que o último da lista passa para o primeiro (topologia em anel). Ao final do laço evolutivo o conjunto de aproximação gerado é o conjunto da união de todas as soluções não dominadas de cada *sub-swarm*.

**Algoritmo 10** Algoritmo Multi-swarm baseado em arquivadores

**Entrada** $num_{swarms}$: Número de *sub-swarms*
  $arquivadores$ : lista dos métodos de arquivamento
  $num_{troca}$: número de iterações para a troca de informações
  $pop_{swarm}$: Tamanho da população de cada *sub-swarm*
  $arq_{swarm}$: Tamanho do arquivo externo de cada *sub-swarm*

Iniciar cada *sub-swarm* com o método de arquivamento passado como parâmetro
Enquanto `condição de término não alcançada`
  Para cada *sub-swarm* $s_i$ faça
    Executar o movimento das partículas de $s_i$
    Arquivamento dos líderes de $s_i$
  Se `for uma iteração de troca`
    *Sub-swarm* $i$ passa as soluções do seu arquivo para *sub-swarm* $i + 1$
    *Sub-swarm* $num_{swarms}$ passa as soluções do seu arquivo para *sub-swarm* 1
  Definir os líderes de todas as partículas de cada *sub-swarm*
Retorno: Melhores soluções encontradas pelos sub-swarms

## 6.2 *Multi-Swarm* com arquivo externo compartilhado

O segundo algoritmo desenvolvido, *Multi-Swarm* com Arquivo Externo Compartilhado, também é baseado na ideia da composição de diferentes métodos de arquivamento, porém busca simplificar a iteração entre cada *sub-swarm*. Nesse método, a comunicação ocorre através de um arquivo externo compartilhado entre todos os diferentes arquivadores. Assim, um *sub-swarm* insere e escolhe líderes de um mesmo conjunto de soluções.

Esse algoritmo busca construir um bom conjunto de líderes através da execução de diferentes métodos, uns privilegiando convergência e outros diversidade. A Figura 6.2 mostra a representação de diferentes métodos de arquivamento se comunicando através de um arquivo externo compartilhado. O método não limita o número de enxames, nem quais arquivadores serão escolhidos, sendo possível a repetição de um mesmo método, caso alguma qualidade queira ser ressaltada.

O ponto forte do método é derivado do fato que diferentes arquivadores irão introduzir diferentes soluções no arquivo, aumentando a diversificação de soluções do arquivo. Além disso, uma partícula de um *sub-swarm* pode escolher como líder um solução que originalmente foi alcançada por uma outra população e com isso evitar que a busca fique presa em ótimos locais. Porém, alguns problemas podem ocorrer. Como os arquivadores são

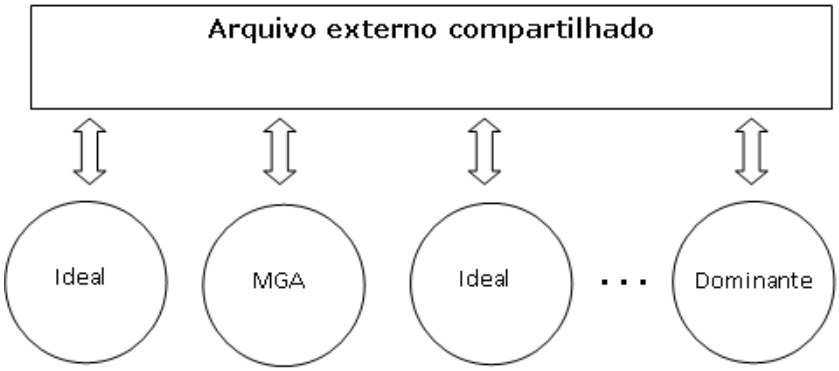



Figura 6.2: Representação do esquema de comunicação do *Multi-Swarm* com arquivo externo compartilhado.

baseados no esquema do Arquivador Preciso [22], caso o arquivo esteja cheio, uma solução só entra se alguma outra for removida. Nesse sentido, a busca pode ficar estagnada pois pode acontecer de um arquivador sempre remover do arquivo soluções inseridas por outro método.

Como discutido anteriormente, os métodos propostos não são baseados em programação paralela ou distribuída. Todos são executados num mesmo processador de forma sequencial, o que torna simples o uso de uma memória compartilhada. Em trabalhos futuros, os algoritmo com múltiplos enxames serão explorados em ambientes paralelos, o que irá introduzir novas desafios ainda não explorados, tais como sincronia e atomicidade das operações no arquivo.

O Algoritmo 11 apresenta uma descrição do *Multi-Swarm* com Arquivo Externo Compartilhado. Os parâmetros são semelhantes aos do algoritmo *Multi-Swarm* Baseado em Arquivadores. Além dos parâmetros base do SMPSO, deve ser definido o número de *sub-swarms*, o método de arquivamento escolhido para cada um e o tamanho das subpopulações. Além disso, deve ser definido o tamanho do arquivo externo. Em geral, quanto maior é o tamanho do arquivo externo, melhor é o resultado do algoritmo PSO [9], porém um arquivo externo muito grande ou ilimitado prejudica o desempenho da busca, como discutido no capítulo anterior. Porém, o tamanho do repositório pode influenciar a comunicação entre os múltiplos enxames e este parâmetro será explorado na avaliação desse algoritmo. O primeiro passo é o início de cada *sub-swarm* e a definição dos líderes iniciais. Após é executado o laço evolutivo. Esse laço segue os passos básicos de um algoritmo

**Algoritmo 11** Algoritmo com Arquivo Externo Compartilhado

1: **Entrada** $num_{swarms}$: Número de *sub-swarms*
2: $arquivadores$ : lista dos métodos de arquivamento
3: $pop_{swarm}$: Tamanho da população de cada *sub-swarm*
4: $arq_{swarm}$: Tamanho do arquivo externo compartilhado
5:
6: Iniciar cada *sub-swarm* com o método de arquivamento passado como parâmetro
7: Enquanto `condição de término não alcançada`
8: Para cada *sub-swarm* $s_i$ faça
9: Executar o movimento das partículas de $s_i$
10: Arquivamento dos líderes de $s_i$ no arquivo compartilhado
11: Definir os líderes de todas as partículas de cada *sub-swarm*
12: Retorno: Soluções presentes no arquivo externo

PSO com múltiplos enxames: movimento das partículas de cada *sub-swarm*, definição e escolha dos líderes. O diferencial é que cada *sub-swarm* insere um elemento no arquivo utilizando diferentes métodos. Ao final, o conjunto de aproximação gerado são as soluções presentes no arquivo externo compartilhado.

## 6.3 I-Multi: *Iterated Multi-Swarm*

O algoritmo *Iterated Multi-Swarm*, I-Multi, apresenta duas características: compor diferentes métodos de arquivamento e reduzir a complexidade da busca através da subdivisão do espaço das variáveis. Esse algoritmo não busca somente produzir um conjunto de *sub-swarms* que se comunicam trocando informações sobre os líderes através de métodos de arquivamento. Cada *sub-swarm* é definido através da limitação da busca em uma subregião do espaço de busca. Assim, não somente o uso de diferentes métodos de arquivamento irá buscar uma melhora na convergência e diversidade, mas também cada *sub-swarm* irá focar sua busca em menores regiões, espalhadas pelo espaço.

O I-Multi é baseado no inicialmente apresentado em [68]. Esse trabalho divide o espaço da busca entre diferentes *sub-swarms*. O algoritmo apresenta duas fases distintas: uma execução inicial para a geração de um primeiro conjunto de aproximação e uma execução de um conjunto de iterações com o objetivo de explorar diferentes regiões do espaço de busca. A primeira etapa busca construir um conjunto base bem localizado no espaço de objetivos. Após a definição de um conjunto inicial é feita a divisão da busca entre

múltiplos enxames. Essa fase tem como objetivo identificar um conjunto de partículas sementes a partir do conjunto de aproximação inicial. Essas partículas irão definir as regiões que serão exploradas e a geração da população de cada *sub-swarm*. É definida uma área de busca em torno de cada partícula semente que é o limite da busca de cada *sub-swarm*. Assim, dois aspectos importantes podem ser ressaltados dessa fase, o método de escolha das sementes e o processo de definição para as subregiões do espaço de busca. Após cada *sub-swarm* terminar a busca, todas as soluções não dominadas são unidas e o processo de encontrar as sementes e dividir a região de busca é repetido. Esse processo é repetido um número de iterações definido pelo usuário.

O algoritmo I-Multi utiliza as vantagens do algoritmo proposto em [68] com o objetivo de explorar os métodos de arquivamento com múltiplos enxames. O algoritmo proposto utiliza o princípio de inicialmente executar um método de arquivamento que privilegia diversidade para em seguida executar múltiplos enxames que privilegiem convergência. Um esquema do comportamento do I-Multi é apresentado na Figura 6.3. Para isso, o I-Multi é dividido em duas fases: fase de diversidade e busca *multi-swarm*. O I-Multi é descrito no Algoritmo 12.

A fase de diversidade corresponde à busca inicial do algoritmo com o objetivo de se gerar o primeiro conjunto de aproximação. Esse primeiro conjunto de aproximação deve ser bem diversificado para que seja possível executar os *sub-swarms* em diferentes regiões do espaço de busca. Assim, é utilizado um algoritmo MOPSO (baseado no SMPSO [70]) com o método de arquivamento MGA [61]. Ao final dessa fase, é gerado um conjunto de aproximação inicial que é utilizado como base para a definição da busca com múltiplos enxames, chamado de conjunto base.

A segunda fase do algoritmo corresponde à busca *multi-swarm* com objetivo de introduzir convergência em direção à fronteira de Pareto ao algoritmo. Essa fase é definida por um processo que é iterativo. Essas iterações, são chamadas de iterações de particionamento e repetem o seguinte processo: escolha das sementes, divisão dos *sub-swarms*, definição da região de busca, construção dos enxames, execução dos algoritmos e junção das soluções não dominadas.

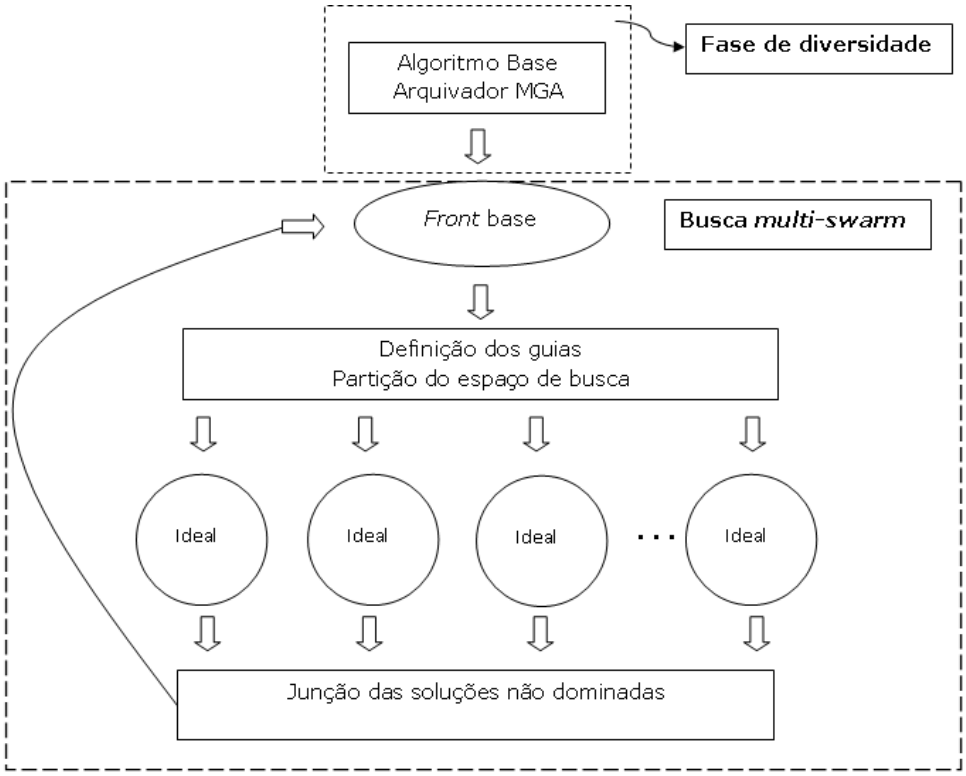



Figura 6.3: Representação do esquema do algoritmo I-Multi.

Nessa etapa um conjunto de *sub-swarms* é utilizado, cada um explorando uma área específica do espaço de busca e executando um arquivador que introduza maior convergência. Para definição da região de cada *sub-swarm* o algoritmo escolhe um conjunto de partículas sementes (escolha das sementes).

Uma partícula semente representa um *sub-swarm* e é o centro da área de busca selecionada. A partir dessa posição central, para cada dimensão, é definida uma região limitada por um valor superior e inferior (definição da região de busca). Esses valores, são definidos através de um intervalo passado como parâmetro. A Figura 6.4 mostra um exemplo da divisão de um espaço de busca com duas dimensões. Uma partícula semente é representada como um ponto, os retângulos menores indicam as subregiões de busca, enquanto a região do espaço de busca é limitada pelo retângulo maior. Além disso, o intervalo de busca é atualizado durante as iterações do algoritmo, variando entre um intervalo inicial e um intervalo final, ambos passados por parâmetro.

Após a definição das sementes e definição dos intervalos, cada *sub-swarm* deve ser iniciado (construção dos enxames). Nessa etapa, a população é iniciada e o arquivo

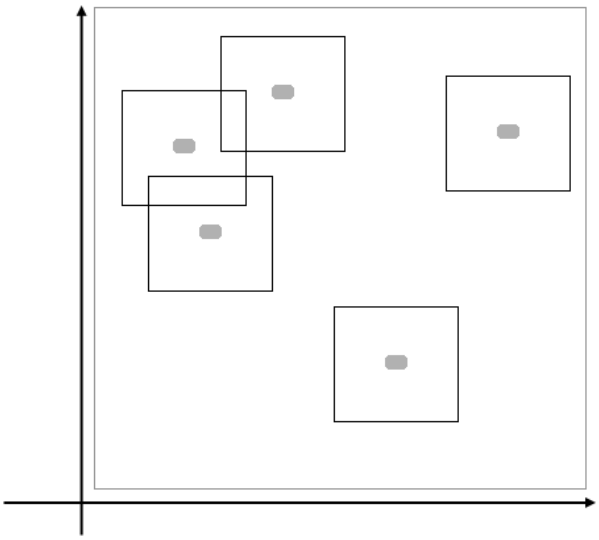


Figura 6.4: Representação das subregiões definidas pelas partículas sementes.

externo é preenchido. Em seguida, é executado o algoritmo SMPSO com o Arquivador Ideal em cada subregião (execução dos algoritmos). A hipótese é que cada *sub-swarm* irá se aproximar de uma região da fronteira de Pareto, porém como existem vários *sub-swarms* espalhados pelo espaço de busca vai haver também diversidade na busca. Por fim, ao final da execução dos algoritmos as soluções não dominadas encontradas são unidas (junção das soluções não dominadas). Esse processo é iterativo, e o conjunto base utilizado para escolha das sementes passa a ser o conjunto das novas soluções não dominadas e o processo é repetido por um número de iterações escolhido pelo usuário. Em uma nova iteração, são definidas novas sementes para *sub-swarms*. Como o espaço explorado é modificado pela escolha de uma nova semente, a população dos múltiplos enxames é reiniciada dentro dos novos limites. Porém, o arquivo externo de cada *sub-swarm* é mantido durante a busca. Nesse algoritmo a comunicação entre os *sub-swarms* ocorre de forma indireta. Primeiro, um *sub-swarm* pode escolher como semente uma solução do conjunto base que tenha sido encontrada por outro *sub-swarm*. Além disso, a cada nova definição de novas sementes o arquivo externo é atualizado com soluções do conjunto base que sejam semelhantes à semente (esse processo será discutido nas próximas seções). Assim, na fase de construção do enxame, um *sub-swarm* pode adicionar soluções de outros *sub-swarm* em seu arquivo externo.

Nessa fase da busca *multi-swarm* diferentes ações podem ser tomadas em cada fase.

**Algoritmo 12** Iterated Multi-Swarm

1: **Entrada** $num_{swarms}$: Número de *sub-swarms*
2: $num_{split}$: Número de iterações de particionamento
3: $pop_{swarm}$: Tamanho da população de cada *sub-swarm*
4: $rep_{swarm}$: Tamanho do arquivo de cada *sub-swarm*
5: $int_{inicial}$: Intervalo inicial
6: $int_{final}$: Intervalo final
7: *escolha_semente*: Método de escolha das sementes
8:
9: **Fase de diversidade - Arquivador MGA**
10: *conjunto_base* = Executar o algoritmo base MGA-SMPSO
11: **Busca Multi-Swarm - Arquivador Ideal**
12: $intervalo\_busca = int_{inicial}$
13: Para 0 até $num_{split}$ faça
14: sementes = Definir a semente de cada *sub-swarm* utilizando o *conjunto_base* através do método *escolha_semente*
15: Definir a região de busca de cada *sub-swarm* com intervalo *intervalo_busca*
16: swarms = Construir os $num_{swarms}$ *sub-swarm*
17: Limpar a lista *conjunto_base*
18: Para cada *sub-swarm* $s_i$ faça
19: $swarm\_front$ = Executar o *sub-swarm* $s_i$(Algoritmo: Ideal-SMPSO)
20: adicionar as soluções $swarm\_front$ ao *conjunto_base*
21: Atualizar intervalo de busca *intervalo_busca*
22: Retornar *conjunto_base*

Os detalhes do desenvolvimento dessas etapas são discutidos a seguir.

### 6.3.1 Escolha das partículas semente

A escolha das sementes é um dos passos mais importantes no algoritmo I-Multi. O objetivo desse algoritmo é combinar os métodos de arquivamento MGA e Ideal. O MGA é utilizado para construir um conjunto de soluções inicial diversificado e o Ideal para aumentar a convergência da busca. Porém, se as sementes não forem bem escolhidas pode não haver um bom espalhamento dos múltiplos enxames pelo espaço de busca. Isso acarretará numa boa convergência, obtida pelo Arquivador Ideal, porém não haverá diversidade e o esforço inicial do Arquivador MGA será perdido.

Neste trabalho, três diferentes abordagens de escolha das sementes são exploradas: I-Multi Centroid, I-Multi Dimensões e I-Multi Aleatório.

**I-Multi Centroid:** Essa abordagem foi apresenta em [68] e encontra as sementes

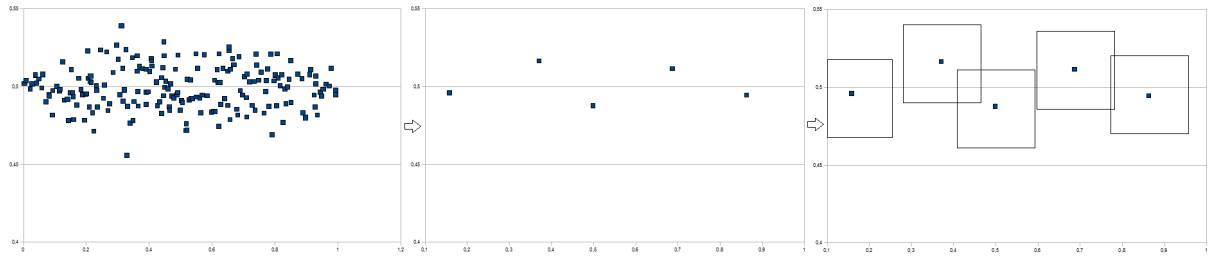


Figura 6.5: Escolha das sementes efetuada pelo I-Multi Centroid.

através de um algoritmo de agrupamento [50]. O agrupamento é feito com o conjunto de vetores objetivo e os centróides definem as sementes.

Cada semente deve representar um enxame, logo são definidos $num_{swarms}$ sementes. Dado o conjunto base, é executado o algoritmo $K$-Means para encontrar $num_{swarms}$ grupos nesse conjunto de vetores. O centroide de cada grupo é definido como a semente de cada *sub-swarm*. A Figura 6.5 mostra um exemplo da execução do I-Multi Centroid. O conjunto de pontos à esquerda representa o conjunto de aproximação gerado pela fase de diversidade. O algoritmo foi aplicado ao problema DTLZ2, com 2 variáveis de decisão. A primeira parte da figura representa todas as soluções traçadas no espaço das variáveis de decisão bidimensional. Em seguida, são plotados somente os centroides encontrados pelo algoritmo $K$-Means (5 grupos). Por fim, é apresentado um exemplo da definição da subregião do espaço de busca, que será discutida na próxima seção.

**I-Multi Aleatório:** É o método mais simples. Dado o conjunto base para a escolha sementes, o algoritmo escolhe $num_{swarms}$ partículas sementes de forma aleatória. A Figura 6.6 mostra um exemplo dessa seleção. A parte esquerda da Figura 6.5 mostra um conjunto de soluções plotadas num espaço com duas dimensões. Desse conjunto, são escolhidas $num_{swarms}$ sementes, nesse exemplo $num_{swarms} = 5$. Por fim, é definida a região de busca para cada semente.

**I-Multi Extremos:** A segunda abordagem tem como objetivo selecionar as sementes mais próximas dos pontos extremos de cada dimensão do espaço de objetivos. Um vetor com valores extremos para o função objetivo $j$, $E_j$, possui valores de várias funções objetivos baixo (minimização) e valor para uma função objetivo alto. Em geral, esses pontos definem extremos da fronteira de Pareto e estão espalhados por diferentes regiões

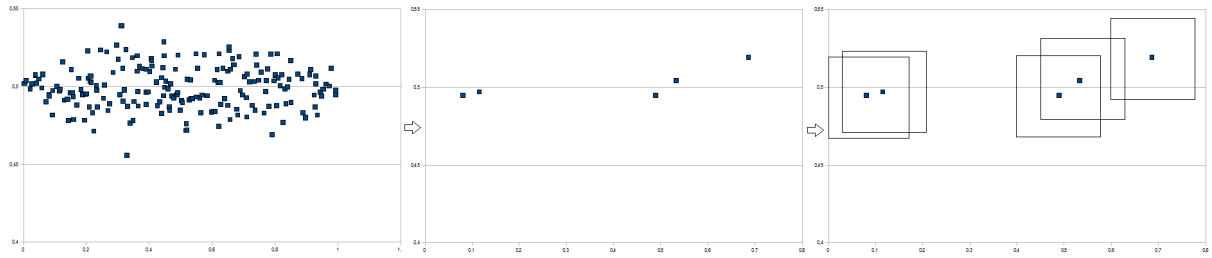


Figura 6.6: Escolha das sementes efetuada pelo I-Multi Aleatório.

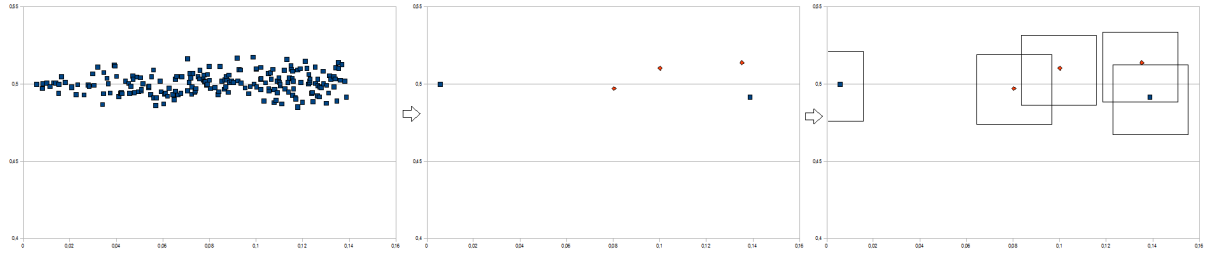


Figura 6.7: Escolha das sementes efetuada pelo I-Multi Extremos.

do espaço de objetivo.

Na abordagem proposta, inicialmente são identificados esses pontos para cada dimensão do espaço de objetivos. Após a definição dos pontos, eles são escolhidos como sementes de $m$ *sub-swarms*. Se o número de funções objetivo ($m$) for menor que o número de *sub-swarms* ($num_{swarms}$), as demais sementes são escolhidos de forma aleatória (mesmo esquema do I-Multi aleatório). Caso $num_{swarms}$ seja maior que $m$, algumas dimensões não serão contempladas pelo algoritmo. Para evitar que sempre as primeiras funções do objetivo sejam sempre escolhidas para a definição dos pontos extremos, as dimensões escolhidas são definidas de forma aleatória. Após a definição das sementes são definidas as subregiões de busca no espaço das variáveis de decisão.

A Figura 6.7 mostra o exemplo do I-Multi Extremos. Dado um conjunto de soluções num espaço de dimensões com duas dimensões, primeiro são escolhidas as soluções nos extremos do espaço de objetivos. Essas soluções estão marcadas através de quadros azuis, na segunda parte da figura. No exemplo são definidos 5 enxames, assim para completar as demais sementes são escolhidas de forma aleatória. Por fim, são definidas as regiões de busca para cada semente.

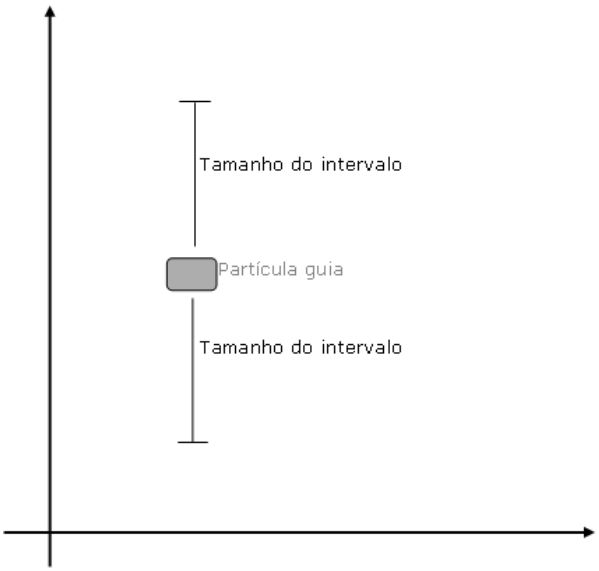



Figura 6.8: Exemplo da definição da região de busca para uma dimensão do espaço de busca.

### 6.3.2 Definição das subregiões de busca

Após a definição das partículas semente, a região de busca de cada *sub-swarm* deve ser definida. Para isso, é executado um algoritmo simples. Dado o vetor com as variáveis de decisão da partícula semente, para cada dimensão desse vetor é adicionado e diminuído um valor do intervalo que define a nova região de busca. A Figura 6.8 mostra o exemplo da definição dessa região para uma das dimensões. Dada a partícula semente é obtido um valor superior e um valor inferior.

Uma opção do algoritmo I-Multi é a atualização desse intervalo a cada iteração de particionamento do I-Multi que define a seguinte estratégia: são definidos dois valores pelo usuário, intervalo inicial ($int_{inicial}$) e intervalo final ($int_{final}$). Em seguida, é calculado um incremento que leva o valor do intervalo inicial até o intervalo final durante as iterações de particionamento ($num_{particoes}$). O incremento é calculado da seguinte maneira:

$$incremento = \frac{|int_{inicial} - int_{final}|}{num_{particoes}} \tag{6.1}$$

Definido o valor do incremento, o primeiro passo, antes da execução do laço com as iterações de partição no Algoritmo 12, é a definição do *intervalo_busca* como sendo o valor de $int_{inicial}$. Em seguida, ao final de cada iteração esse intervalo é atualizado através do

**Algoritmo 13** Atualização do intervalo das subregiões de busca

1: **Entrada** *intervalo_busca*: Intervalo corrente da busca
2: $inc_{inicial}$ - Intervalo inicial
3: $inc_{final}$ - Intervalo final
4: *incremento* - Incremento
5:
6: Se $inc_{inicial} > inc_{final}$
7: $intervalo\_busca = intervalo\_busca$ - $incremento$
8: Caso contrário $intervalo\_busca = intervalo\_busca + incremento$

procedimento descrito no Algoritmo 13.

### 6.3.3 Construção dos enxames

A etapa da construção dos enxames é definida pela inicialização da população e preenchimento do arquivo externo de cada *sub-swarm*. Após a definição das sementes e da região da busca, cada *sub-swarm* deve iniciar uma nova população. No I-Multi, em cada nova iteração, novas populações são criadas para cada *sub-swarm* de forma aleatória.

Nesse processo de criar novas partículas, há perda de informação, pois partículas bem localizadas no espaço de busca são descartadas. Porém, o I-Multi mantém informação entre as iterações do algoritmo através do processo de arquivamento. Isso ocorre de duas formas: primeiro, o arquivo é utilizado como base para a definição das sementes e segundo, a cada iteração o arquivo externo de cada *sub-swarm* é atualizado com as soluções do conjunto base. Além disso, os arquivos externos dos *sub-swarms* não são apagados entre cada iteração e são mantidos durante toda a busca.

Esse processo de atualizar cada arquivo externo com soluções do conjunto base evita que cada *sub-swarm* utilize somente soluções de sua busca, pois é possível que durante o processo de atualização seja escolhida uma solução encontrada por outro enxame. Nesse processo há uma comunicação indireta entre os múltiplos enxames que têm como objetivo aumentar a diversidade da busca. É importante ressaltar, que um *sub-swarm* pode conter em seu arquivo externo uma solução que esteja localizada fora de sua região de busca. Caso uma dessas soluções seja escolhida como líder irá guiar a partícula para uma região fora do região de busca, porém, as partículas não extrapolam esse limite.

O processo de atualização dos arquivos externos através das soluções do conjunto base é definido através do método de escolha das sementes. As soluções escolhidas são relacionadas à semente escolhida. As soluções são escolhidas da seguinte maneira:

Se o método de escolha das sementes for o **I-Multi Centroid**, é utilizado o agrupamento de dados. Todas as soluções agrupadas no mesmo centroide utilizado como semente são submetidas ao arquivo externo. O Arquivador Ideal decide quais soluções irão entrar ou não no arquivo externo.

Se o método de escolha das sementes for o **I-Multi Extremos**, as soluções mais próximas ao ponto extremo escolhido são submetidas ao Arquivador Ideal. São submetidas as $rep_{swarm}$ (tamanho do arquivo externo do *sub-swarm*) mais próximas ao ponto extremo escolhido.

Se o método de escolha das sementes for o **I-Multi Aleatório**, o processo de atualização do arquivo externo é simples. São escolhidas $rep_{swarm}$ soluções do conjunto base de forma aleatória e essas soluções são submetidas ao Arquivador Ideal.

## 6.4 Considerações finais

O uso de múltiplos enxames é uma importante ferramenta para se reduzir a complexidade da busca de algoritmos PSO, em especial na Otimização Multiobjetivo. Esse método já foi explorado por trabalhos da literatura que exploraram diferentes ideias tais como, a exploração de diferentes dimensões em cada *sub-swarm* [73], a divisão do espaço de busca em regiões menores [68], entre outros [76] [36].

O principal objetivo desta tese é explorar Problemas de Otimização com Muitos Objetivos. Um das estratégias abordadas refere-se ao uso de métodos de arquivamento, que buscam ressaltar características como convergência ou diversidade da busca, visando reduzir a deterioração de MOEAs nesse contexto. Porém, a maioria dos métodos propostos até então para a Otimização com Muitos Objetivos somente se preocupam em explorar uma das características, quando se aumenta a convergência se perde diversidade e vice-versa.

Assim, essa tese propõe algoritmos com múltiplos exames que compõem diferentes métodos de arquivamento, para que haja uma combinação da exploração da convergência

e da diversidade. Para isso, são utilizados diferentes métodos de arquivamento, para juntar as vantagens de todos os métodos. Com esse objetivo são propostos *Multi-Swarm* Baseado em Arquivadores, *Multi-Swarm* com Arquivo Externo Compartilhado e *Iterated Multi-Swarm*. O próximo capítulo irá apresentar a validação do I-Multi, irá testar diferentes configurações dos parâmetros e irá confrontá-lo com algoritmos da literatura em cenários com problemas com muitos objetivos. A validação dos outros dois algoritmos é um dos trabalhos futuros deste doutorado.

# CAPÍTULO 7

# AVALIAÇÃO EMPÍRICA DOS MÉTODOS PROPOSTOS

Os capítulos anteriores desta tese discutiram a Otimização por Nuvem de Partículas Multiobjetivo aplicada na Otimização com Muitos Objetivos. Baseados nesses conceitos e na análise de trabalhos da literatura, foram propostos novos métodos que têm como principal objetivo melhorar os resultados de algoritmos MOPSO quando utilizados em Problemas com Muitos Objetivos.

Neste trabalho foram exploradas três abordagens: novas relações de preferência (Capítulo 4), métodos de arquivamento (Capítulo 5) e uso de múltiplos enxames (Capítulo 6). Todas essas novas técnicas buscam melhorar o desempenho dos algoritmos MOPSO em termos de convergência e diversidade em relação à fronteira de Pareto.

Para medir a qualidade e avaliar o desempenho dessas novas técnicas e algoritmos em Problemas de Otimização com Muitos Objetivos este capítulo apresenta um conjunto de estudos empíricos. Esses experimentos utilizam diferentes problemas com muitas funções objetivo e um conjunto de indicadores de qualidade. Para cada técnica estudada, é avaliada a influência dos parâmetros, bem como é feita a comparação dos algoritmos propostos entre si e com alguns métodos da literatura identificando qual técnica apresenta o melhor desempenho.

Este capítulo está organizado da seguinte forma: A Seção 7.1 discute a metodologia utilizada na condução dessa análise empírica. Em seguida, as Seção 7.2 e 7.3 discutem a avaliação de cada método proposto. Por fim, a Seção 7.4 apresenta as considerações finais dos experimentos.

## 7.1 Metodologia

A análise empírica apresentada neste capítulo está dividida em três partes: análise das novas relações de preferência, análise dos métodos de arquivamento e análise dos métodos

de múltiplos enxames. Para cada conjunto de experimentos, inicialmente é feita a análise dos parâmetros de cada método, buscando-se encontrar as melhores configurações. Outro objetivo é confrontar os diferentes métodos propostos, tanto entre si, quanto com outros algoritmos da literatura.

O desenvolvimento desta tese seguiu a ordem da apresentação dos capítulos. Primeiro foram exploradas as relações de preferência, em seguida os métodos de arquivamento e por fim os algoritmos com múltiplos enxames. Assim, os experimentos também foram conduzidos nessa sequencia. Então, os melhores resultados das relações de preferência já eram conhecidos na comparação dos métodos de arquivamento e assim por diante. Logo, por clareza e para facilitar a comparação, os métodos desenvolvidos nas etapas finais da tese só são comparados com os melhores resultados das etapas anteriores.

Esse conjunto de experimentos mede a qualidade dos resultados dos métodos desenvolvidos em diferentes problemas com muitos objetivos. Para isso foi utilizada a família de Problemas de Otimização com Muitos Objetivos DTLZ [31] que permite a criação de problemas artificiais com qualquer número de funções objetivo. Esses problemas são apresentados na Seção 7.1.2. Para medir a qualidade dos resultados foi utilizado um conjunto de indicadores de qualidade, discutidos na Seção 7.1.3.

Os experimentos conduzidos se baseiam na medição dos indicadores de qualidade através do aumento do número de funções objetivo. Como discutido anteriormente, as técnicas da Otimização com Muitos Objetivos reduzem a deterioração dos MOEAs que ocorre quando o número de objetivos cresce. Assim, nos experimentos apresentados neste capítulo será medida a qualidade dos algoritmos para diferentes números de funções objetivo. Os algoritmos e os parâmetros utilizados são discutidos na próxima seção. Será utilizado um algoritmo previamente validado na literatura como base para avaliação dos resultados.

### 7.1.1 Algoritmos e parâmetros

O algoritmo SMPSO [70] foi utilizado como algoritmo base no desenvolvimento deste trabalho. Isso ocorreu, pois, de acordo com os resultados apresentados em [70] e [35]

Tabela 7.1: Parâmetros utilizados na execução do SMPSO

| **Parâmetro** | **Configuração** |
|---|---|
| $\varpi$ | dinâmico [0, 0,8] |
| $C_1$ e $C_2$ | dinâmico [1,5, 2,5] |
| $\phi_1$ e $\phi_2$ | dinâmico [0, 1] |
| Mutação | $p_{mut} = 1/n$ |
| Limite da velocidade | $[-5, +5]$ |

esse algoritmo apresenta os melhores resultados quando comparados a outros algoritmos MOPSO da literatura. É importante ressaltar que as técnicas desenvolvidas não são somente aplicadas ao SMPSO, visto que são utilizados conceitos bases comuns em todos os algoritmos MOPSO, tais como o uso de relações de preferência e métodos de arquivamento. Os algoritmos de múltiplos enxames propostos também utilizam o SMPSO, porém o importante é a maneira que os métodos trocam informações e não qual é o algoritmo utilizado.

Os parâmetros comuns do SMPSO utilizados em todos os experimentos são apresentados na Tabela 7.1. Esses parâmetros foram definidos através de trabalhos da literatura [70], [35], [67], [6] e através de alguns experimentos. Os demais parâmetros, tais como número de partículas, gerações ou número de avaliações, tamanho do arquivo, bem como os parâmetros específicos de cada método proposto são apresentados na seção dos resultados.

### 7.1.2 Problemas de *benchmark*

Para análise empírica desenvolvida neste capítulo, é utilizado um conjunto de problemas de *benchmark* que podem ser escalados com muitas funções objetivo. Esses problemas são utilizados por dois motivos: primeiro, é possível calcular a fronteira de Pareto e assim medir, através de indicadores de qualidade, aspectos como convergência e diversidade em relação às melhores soluções; segundo, problemas de *benchmark* podem ser projetados para analisar aspectos específicos da busca como convergência, diversidade ou a habilidade dos algoritmos quando trabalham com fronteira de Pareto desconexa.

Com intuito de comparar a escalabilidade de algoritmos multiobjetivo para muitos

objetivos, em [31] foi proposta a família de problemas de teste DTLZ. Esses problemas compartilham algumas características importantes para a nossa análise: a) esforço pequeno de implementação; b) podem ser escalados para qualquer número de objetivos ($m$) e número de variáveis de decisão ($n$); c) a fronteira de Pareto é conhecida analiticamente; d) aspectos como convergência e diversidade podem ser controlados.

Para cada problema, a complexidade da busca é definida pela variável $k$, onde $k = n-m+1$ ($n$ é o número de variáveis e $m$ é o número de objetivos) define a relação entre o número de variáveis de decisão e o número de objetivos, assim, quanto maior o valor de $k$ maior será a dimensão do espaço de busca. Além disso, cada problema utiliza o vetor $\left|\overrightarrow{\mathbf{x}_m}\right|$ = k, contendo os últimos $k$ valores de $\overrightarrow{f(\overrightarrow{x})}$. Esse vetor $\overrightarrow{\mathbf{x}_m}$ é utilizado como parâmetro de uma função $g$, que na definição dos problemas DTLZ, é utilizada para a geração do espaço de objetivos. Para a definição de $g$ podem ser utilizadas diferentes funções com o objetivo de se introduzir algumas dificuldades na busca. Logo, quanto maior o valor $k$, mais complexa será a função $g$ e mais difícil será a busca realizada pelo algoritmo.

Outro ponto importante na construção dos problemas DTLZ refere-se às equações paramétricas que associam os valores do vetor no espaço das variáveis de decisão e a construção do espaço das funções objetivo. A família DTLZ utiliza diferentes associações com diferentes objetivos, como por exemplo modificar a distribuição das soluções no espaço de objetivos. As diferentes funções DTLZ propostas buscam combinar fronteiras de Pareto em diferentes formatos, com diferentes funções de dificuldade $g$ e diferentes equações paramétricas.

Nesta tese, quatro destas funções foram escolhidas, DTLZ2, DTLZ4, DTLZ6 e DTLZ7. Essas funções apresentam diferentes dificuldades aos algoritmos de busca e são utilizadas para se obter uma análise complementar da qualidade dos algoritmos desenvolvidos. Além disso, estamos interessados em analisar o comportamento das novas estratégias para problemas com muitas funções objetivos. A Tabela 7.2 mostra o número de objetivos de variáveis de decisão utilizadas em nossos experimentos, $k$ foi definido com valor 10 para todos os experimentos. Esse valor foi definido, para que as dificuldades impostas pelos problemas escolhidos se sobressaiam e seja possível a análise dos métodos propostos em

Tabela 7.2: Números de objetivos e de variáveis de decisão utilizados nos experimentos com $k = 10$

| **Objetivos** $m$ | **Variáveis de decisão** $n$ |
|---|---|
| 2 | 11 |
| 3 | 12 |
| 5 | 14 |
| 10 | 19 |
| 15 | 24 |
| 20 | 29 |

diferentes cenários. A seguir, cada problema é detalhado.

Problema DTLZ2:

Esta função é utilizada para investigar a habilidade dos algoritmos em escalar seu desempenho para um número grande de objetivos. Com este problema pode-se observar a habilidade dos algoritmos em convergir para a fronteira de Pareto. O DTLZ2 é definido por:

$min f_{x\in\Omega}(\overrightarrow{x})$

$\Omega = \{\overrightarrow{x} | 0 \leq x_i \leq 1, \forall i = 1, ..., n\}$

$$\begin{cases} f_1(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(x_1.\pi/2))...cos(x_{m-2}.\pi/2)).cos(x_{m-1}.\pi/2)) \\ f_2(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(x_1.\pi/2))...cos(x_{m-2}.\pi/2)).sin(x_{m-1}.\pi/2)) \\ . \\ . \\ f_m(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).sin(x_1.\pi/2) \\ g(\overrightarrow{\mathbf{x}_m}) = \sum_{x_i \in \overrightarrow{\mathbf{x}_m}} (x_i - 0.5)^2 \\ Pareto\ otimo : x_i = 0.5, \forall i \in \overrightarrow{\mathbf{x}_m}. \end{cases}$$

onde $\Omega$ é o conjunto com todas as soluções possíveis e $\overrightarrow{\mathbf{x}_m}$ é um vetor, onde $|\overrightarrow{\mathbf{x}_m}| = k$ e contém os $k$ últimos valores de $x$. O DTLZ2 é definido através da equação de uma esfera, assim para 3 funções objetivo a fronteira de Pareto é definida por um octante da esfera. A Figura 7.1 mostra a equação que define a fronteira de Pareto para o problema DTLZ2. Essa função é simples, pois não introduz nenhuma dificuldade na função $g$ nem na transformação paramétrica. Porém, devido a esse fator a sua análise é importante, pois

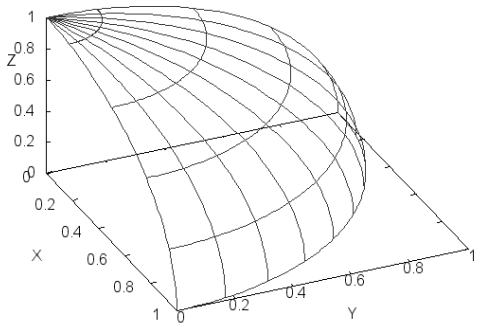



Figura 7.1: Fronteira de Pareto para o problema DTLZ2, 3 funções objetivo.

a maior característica é observar o comportamento de um algoritmo quando o número de objetivos cresce.

Problema DTLZ4:

Este problema é usado para investigar a habilidade dos algoritmos em manter uma boa distribuição das soluções. Pode-se observar a habilidade dos algoritmos obterem uma boa diversidade em relação à fronteira de Pareto. É definido por:

$min f_{x\in\Omega}(\overrightarrow{x})$

$\Omega = \{\overrightarrow{x} | 0 \leq x_i \leq 1, \forall i = 1, ..., n\}$

$$\begin{cases} f_1(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(x_1^\alpha.\pi/2))...cos(x_{m-2}^\alpha.\pi/2)).cos(x_{m-1}^\alpha.\pi/2)) \\ f_2(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(x_1^\alpha.\pi/2))...cos(x_{m-2}^\alpha.\pi/2)).sin(x_{m-1}^\alpha.\pi/2)) \\ . \\ . \\ f_m(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).sin(x_1^\alpha.\pi/2) \\ g(\overrightarrow{\mathbf{x}_m}) = \sum_{x_i \in \overrightarrow{\mathbf{x}_m}} (x_i - 0.5)^2 \\ Pareto\ otimo : x_i = 0.5, \forall i \in \overrightarrow{\mathbf{x}_m} \end{cases}$$

onde $\Omega$ é o conjunto com todas as soluções possíveis e $\overrightarrow{\mathbf{x}_m}$ é um vetor, onde $|\overrightarrow{\mathbf{x}_m}| = k$ e contem os $k$ últimos valores de $x$. O parâmetro $\alpha = 100$ é utilizado, como sugerido em [31]. O problema DTLZ4 é uma extensão do problema DTLZ2, porém com uma diferente equação paramétrica. A equação utilizada privilegia a geração de pontos em uma

região específica da fronteira de Pareto. Esse problema gera mais soluções no plano $f_m - f_1$. Assim, um algoritmo encontra mais facilmente pontos nessa região e tende a perder diversidade na busca. Uma busca que gere bons resultados em termos de diversidade para o DTLZ4, mostra que o algoritmo consegue explorar diferentes regiões do espaço. Em contrapartida, um bom resultado somente em termos de convergência não é o desejado no DTLZ4, pois o problema atrai a busca para uma região muito povoada. O formato da fronteira de Pareto para o DTLZ4 é o mesmo apresentado na Figura 7.1, porém a distribuição dos pontos não ocorre de forma uniforme.

Problema DTLZ6:

Este problema é utilizado para avaliar a habilidade dos algoritmos em escalar seu desempenho e para avaliar convergência. Possui uma função $g$ que torna a convergência mais complexa. O DTLZ6 definido por:

$min f_{x\in\Omega}(\overrightarrow{x})$

$\Omega = \{\overrightarrow{x} | 0 \leq x_i \leq 1, \forall i = 1, ..., n\}$

$$\begin{cases} f_1(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(\theta_1.\pi/2))...cos(\theta_{m-2}.\pi/2)).cos(\theta_{m-1}.\pi/2)) \\ f_2(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(\theta_1.\pi/2))...cos(\theta_{m-2}.\pi/2)).sin(\theta_{m-1}.\pi/2)) \\ f_3(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(\theta_1.\pi/2))...sin(\theta_{m-2}.\pi/2)) \\ . \\ . \\ f_{m-1}(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).cos(\theta_1.\pi/2).sin(\theta_2.\pi/2) \\ f_m(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).sin(\theta_1.\pi/2) \\ g(\overrightarrow{\mathbf{x}_m}) = \sum_{x_i \in \overrightarrow{\mathbf{x}_m}} x_i^{0.1} \\ 0 \leq x_i \leq 1, for i = 1, 2, ..., n \\ Pareto\ otimo : x_i = 0.5, \forall i \in \overrightarrow{\mathbf{x}_m}. \end{cases}$$

onde $\theta_i = \frac{\pi}{4(1+g(r))}(1+2g(r)x_i)$, para 1=1,2,3...(m-1), $\Omega$ é o conjunto com todas as soluções possíveis e $\overrightarrow{\mathbf{x}_m}$ é um vetor, onde $|\overrightarrow{\mathbf{x}_m}| = k$ e contem os $k$ últimos valores de $x$. Nesse problema a fronteira de Pareto é definida por somente uma curva e não por toda a região da esfera. A Figura 7.2 mostra a fronteira de Pareto do DTLZ6, para 3 funções

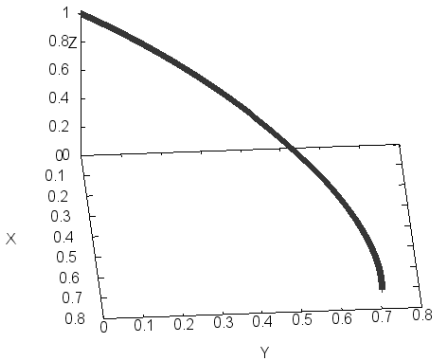



Figura 7.2: Fronteira de Pareto para o problema DTLZ6 com 3 funções objetivo.

objetivo. Duas características se destacam no DTLZ6. Primeiro, a função $g$ é uma função multimodal. Assim, esse problema apresenta ($3^k$-1) ótimos locais. Segundo, é possível plotar diferentes visões da fronteira de Pareto em duas dimensões. Devido a forma da fronteira, toda dimensão que for plotada com a última dimensão forma uma curva. As demais combinações formam uma reta. Com essa característica é possível verificar aonde está localizada a fronteira aproximada.

Problema DTLZ7:

Este problema possui $2^{m-1}$ regiões da fronteira de Pareto desconexas. Testa a habilidade dos algoritmos manterem subpopulações em diferentes regiões da fronteira de Pareto. O DTLZ7 é definido por:

$min f_{x\in\Omega}(\overrightarrow{x})$

$\Omega = \{\overrightarrow{x} | 0 \leq x_i \leq 1, \forall i = 1, ..., n\}$

$$
\begin{cases}
f_1(\mathbf{x}_1) = x_1 \\
f_2(\mathbf{x}_2) = x_2 \\
. \\
. \\
f_{m-1}(\mathbf{x}_{m-1}) = x_{m-1} \\
f_m(\overrightarrow{\mathbf{x}}) = (1 + g(\overrightarrow{\mathbf{x}_m})).h(f_1, f_2, ..., f_{m-1}; g) \\
g(\overrightarrow{\mathbf{x}_m}) = 1 + \frac{9}{|\overrightarrow{\mathbf{x}_m}|}.\sum_{x_i \in \overrightarrow{\mathbf{x}_m}} x_i \\
h(f_1, f_2, ..., f_{m-1}, g) = m - \sum_{i=1}^{m-1}[\frac{f_i}{1+g}.(1 + sin(3\pi f_i)] \\
Pareto\ otimo : x_i = 0, \forall i \in \overrightarrow{\mathbf{x}_m}.
\end{cases}
$$

onde $\Omega$ é o conjunto com todas as soluções possíveis e $\overrightarrow{\mathbf{x}_m}$ é um vetor, onde $|\overrightarrow{\mathbf{x}_m}| = k$ e contem os $k$ últimos valores de $x$. A principal característica do DTLZ7 são as fronteiras desconexas. Com esse problema é possível avaliar o desempenho do algoritmo em encontrar diferentes conjuntos de pontos espalhados pela espaço de objetivos. Um problema dessa algoritmo é que para muitas funções objetivo é que o número de subfronteiras é muito grande o que dificulta bastante a busca. Figura 7.3 ilustra a fronteira de Pareto do DTLZ7, para três funções objetivo.

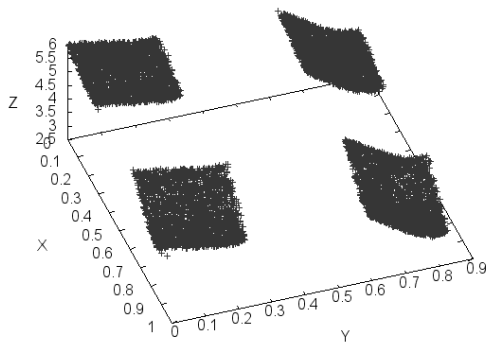



Figura 7.3: Fronteira de Pareto para o problema DTLZ7 com 3 funções objetivo.

As fronteira de Pareto de cada problema foram geradas de forma analítica de acordo com as restrições discutidas em [31]. O número de pontos de cada fronteira de Pareto

Tabela 7.3: Número de pontos guardados na fronteira de Pareto para cada problema DTLZ.

| | 2 | 3 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| **DTLZ2** | 9,999 | 9,901 | 7,381 | 49,203 | 49,173 | 1,571,171 |
| **DTLZ4** | 9,999 | 9,901 | 7,381 | 49,203 | 49,173 | 1,571,171 |
| **DTLZ6** | 10,000 | 10,000 | 10,000 | 10,000 | 100,000 | 100,000 |
| **DTLZ7** | 10,000 | 10,000 | 10,000 | 10,000 | 100,000 | 100,000 |

para os problemas DTLZ utilizados é apresentado na Tabela 7.3. Para os problemas DTLZ2 e DTLZ4 as fronteiras foram geradas de forma uniforme, espalhando os pontos por toda região da função que define os problemas. Para o DTLZ6 e DTLZ7 não houve um procedimento uniforme de geração dos pontos e gerou-se um número de pontos tal que se tenha uma boa formação da fronteira.

### 7.1.3 Indicadores de qualidade

A análise empírica visa medir aspectos como convergência e diversidade da busca de cada nova estratégia proposta em relação à fronteira de Pareto. A seleção dos indicadores de qualidade utilizados para medir o desempenho da busca de MOEAs é um grande obstáculo na Otimização com Muitos Objetivos. Para a medição desses aspectos são utilizados indicadores de qualidade. Indicadores de qualidade são funções que mapeiam $i$ conjuntos de soluções em um número real [89]. Normalmente, é definido $i = 1$, um indicador unário ou $i = 2$, indicador binário. Na literatura, diferentes indicadores de qualidade são utilizados [1] [82] [46], porém não existe consenso sobre qual conjunto de medidas possibilita uma melhor interpretação dos resultados. Dentre esses indicadores se destacam o Hipervolume, o *Epsilon* [89], o *Generational Distance* (GD), *Spacing* [84], *Inverted Generational Distance* (IGD) [20], entre outros.

Um dos maiores problemas na medição de desempenho de MOEAs em Problemas com Muitos Objetivos refere-se à dificuldade de visualização da fronteira aproximada gerada ($PF_{aprox}$), já que com mais de três funções objetivos não há uma maneira natural de se esboçar o conjunto de pontos obtido. Isso dificulta o processo de entendimento dos resultados dos indicadores de qualidade. Além disso, há dificuldade na interpretação de

alguns resultados esperados, por exemplo, algumas técnicas induzem a geração dos pontos para uma região mais próxima ao joelho da fronteira de Pareto.

Em nossos experimentos, já que estamos trabalhando com problemas de benchmark onde a fronteira de Pareto pode ser obtida de forma analítica, é utilizada uma combinação de um conjunto de métricas que buscam medir a distância em relação à fronteira de Pareto. O uso desse conjunto de métricas tem como objetivo observar se a busca dos algoritmos propostos se deteriora em termos de convergência e diversidade quando o número de objetivos cresce. Além disso, são utilizados indicadores de qualidade com o objetivo específico de identificar onde está localizada a $PF_{aprox}$ em relação a pontos de referência, como por exemplo, o joelho da fronteira de Pareto. Os indicadores utilizados são descritos a seguir:

***Generational Distance*** (GD) mede o quão próximo o conjunto de aproximação gerado ($PF_{aprox}$) está em relação à fronteira de Pareto real ($PF_{real}$). O GD é uma medida de minimização. Se o GD é igual a 0, todos os pontos do $PF_{aprox}$ pertencem à fronteira de Pareto. O GD permite observar se o algoritmo converge para alguma região da fronteira de Pareto. A Equação 7.1 define o GD, onde $n$ é o número de soluções pertencentes à $PF_{aprox}$ e $d_i^{a,r}$ é a menor distância Euclidiana entre o ponto $i$ pertencente à $PF_{aprox}$ e um ponto da $PF_{real}$. A Figura 7.4 mostra um exemplo do cálculo do GD. Para cada ponto do conjunto de aproximação é calculada a menor distância em relação à fronteira de Pareto, $d_i^{a,r}$, e é feita a soma dessas distâncias. O exemplo da figura apresenta um bom GD, pois o conjunto de aproximação está próximo à fronteira de Pareto, logo há uma boa convergência.

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^{a,r}}}{n} \tag{7.1}$$

***Convergence*** foi inicialmente apresentada em [40]. Esse indicador de qualidade mede a menor distância de um ponto da $PF_{aprox}$ a $PF_{real}$. Ele é utilizado para auxiliar a análise do GD para medir convergência. A menor distância de um ponto é a menor contribuição do cálculo do GD, assim é possível verificar se a busca alcançou a fronteira de Pareto em pelo menos uma região da fronteira.

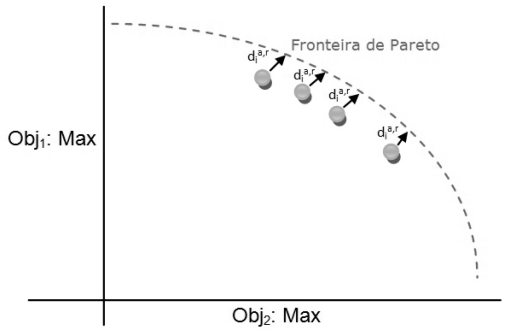



Figura 7.4: Exemplo do cálculo do GD.

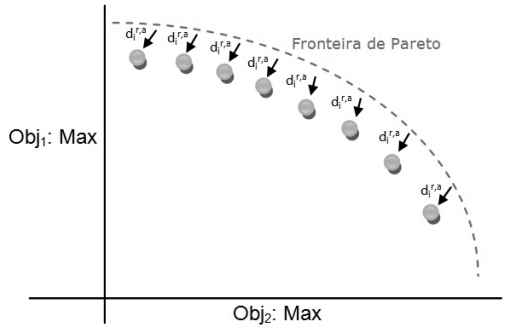



Figura 7.5: Exemplo do cálculo do IGD.

***Inverted Generational Distance*** (IGD) mede a distância mínima entre cada ponto da fronteira de Pareto real em relação ao conjunto de aproximação gerado. O IGD é uma medida de minimização. O IGD permite observar se $PF_{aprox}$ converge para a fronteira Pareto real e se este conjunto é diversificado. A Equação 7.2 define o IGD, onde $n$ é o número de soluções pertencentes à $PF_{aprox}$ e $d_i^{r,a}$ é a menor distância Euclidiana entre o ponto $i$ pertencente à $PF_{real}$ e um ponto da $PF_{aprox}$. A Figura 7.5 mostra um exemplo do cálculo do IGD. Para cada ponto da fronteira de Pareto é calculada a menor distância em relação ao conjunto de aproximação, $d_i^{a,r}$, e é feita a soma dessas distâncias. O exemplo da Figura 7.5 apresenta um bom IGD, pois o conjunto de aproximação está próximo à fronteira de Pareto e também bem distribuído, logo há uma boa convergência e uma boa diversidade.

$$IGD = \frac{\sqrt[2]{\sum_{i=1}^{n} d_i^{r,a}}}{n} \tag{7.2}$$

No entanto nem sempre é possível obter uma visão real da diversidade da busca utilizando somente o IGD. Em alguns casos, onde uma $PF_{aprox}$ está limitada a uma região

específica da fronteira e muito próximo ao $PF_{real}$ (o que implica num valor muito baixo de GD), o IGD tende também a ser pequeno, já que a contribuição dos pontos da $PF_{real}$ dessa região é muito pequena para o cálculo do IGD. Nessa situação, um valor baixo de IGD pode implicar num interpretação errada de boa diversidade. Assim, para ajudar na análise da diversidade da busca, é proposto um novo indicador de qualidade chamado de *Largest Distance*.

***Largest Distance*** (LD) é o oposto do indicador *Convergence*. Ao invés de se obter a menor distância de um ponto do $PF_{aprox}$ em relação à $PF_{real}$, é obtida a maior distância de um ponto da $PF_{real}$ em relação à $PF_{aprox}$. Essa medida indica a maior contribuição de um ponto no cálculo do IGD. Com essa medida, uma $PF_{aprox}$ concentrada numa pequena região da fronteira de Pareto irá gerar um alto valor de LD, já que estará longe de outras regiões da $PF_{real}$. Da mesma forma que, uma $PF_{aprox}$ bem diversificada irá gerar valores baixos de LD, já que está próxima de diferentes regiões da $PF_{real}$.

***Spacing*** mede o intervalo de variância entre soluções vizinhas em um conjunto de aproximação. É uma medida de minimização. Se o valor do *Spacing* é igual a 0, todas as soluções estão igualmente distribuídas. Se a $PF_{aprox}$ contiver uma ou duas soluções o valor é igual a zero e quanto menor o número de soluções, mais fácil o algoritmo consegue controlar o *Spacing* de seu conjunto de aproximação. A Equação 7.3 define o *Spacing*, onde $n$ é o número de soluções pertencentes à $PF_{aprox}$, $d_i = min_j(|f_1^i(x) - f_1^j(x)| + ... + |f_m^i(x) - f_m^j(x)|)$ $i, j = 1, 2, ..., n$ e $\overline{d}$ é a média de todos $d_i$, tal que:

$$Spacing = \sqrt{\frac{1}{n-1}\sum_{i+1}^{n}(\overline{d} - d_i)^2} \tag{7.3}$$

Além dos indicadores de qualidade baseados na distância entre as soluções geradas, esta tese utiliza dois indicadores com objetivo de auxiliar a visualização da $PF_{aprox}$. Esses indicadores medem a distribuição dos pontos da $PF_{aprox}$ em relação a pontos de referência. Assim, é possível observar como diferentes métodos estão distribuídos no espaço de objetivos.

**A distribuição da distância *Tchebycheff*** é utilizada para medir a distribuição da

$PF_{aprox}$ em relação ao joelho da fronteira de Pareto [20]. Joelho é o ponto localizado no maior arqueamento da curva da $PF_{real}$. Estudos apresentados em [24] e [47] afirmam que tomadores de decisão, em geral, preferem os pontos no centro da fronteira. Esse indicador calcula a distância de *Tchebycheff* entre todos os pontos da $PF_{aprox}$ e o joelho da $PF_{real}$.

A distância de *Tchebycheff* é uma métrica definida pela distância entre dois vetores. O cálculo da distância é obtido através da maior diferença entre todas as coordenadas das dimensões, definida pela Equação 7.4.

$$d(z, z^*, \lambda) = max_{1 \leq j \leq m}\{\lambda_j | z_j^* - z_j|\} \tag{7.4}$$

onde $z^*$ é o joelho da fronteira, $z$ é um ponto do $PF_{aprox}$, $m$ é o número de objetivos e $\lambda_j = 1/R_i$, onde $R_j$ é o intervalo para $j-$ésimo objetivo da fronteira de Pareto. Após o cálculo da distância de todos os pontos da $PF_{aprox}$ é gerado um histograma com a distribuição desses valores. Curvas que se localizam perto de pequenos valores da distância indicam uma distribuição dos pontos próximos ao joelho, enquanto curvas mais alongadas indicam uma distribuição mais diversificada. A Figura 7.6 mostra um exemplo com três distribuições. O eixo $x$ indica o valor da distância de *Tchebycheff*, o eixo $y$ a quantidade de soluções localizadas nessa distância. A primeira curva possui um pico em um valor com baixa distância, isso indica uma distribuição dos pontos perto do joelho. A segunda curva também possui os pontos concentrados em poucos valores de distância, porém localizados longe do joelho. A terceira curva é mais alongada, contendo valores em diferentes distâncias. Essa distribuição apresenta pontos mais diversificados.

Além da distribuição da distância de *Tchebycheff* é utilizada outra medida que busca mostrar a distribuição do $PF_{aprox}$, porém ao invés de ser utilizado o joelho da busca, é utilizado um ponto de referência.

**Distribuição dos pontos sobre um ponto de referência** busca medir como diferentes $PF_{aprox}$ geradas por diferentes algoritmos estão distribuídas sobre um ponto de referência. Assim, dadas diferentes $PF_{aprox}$ e um ponto de referência: primeiro é calculada a distância euclidiana entre todos os pontos, de todos os algoritmos, e o ponto de

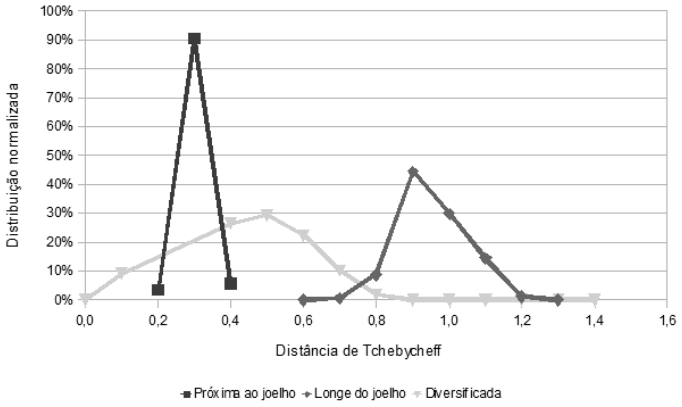



Figura 7.6: Exemplo da distribuição de *Tchebycheff*.

referência. Após, são obtidos o menor e o maior valores da distância. Em seguida, o intervalo variando entre o menor e o maior valor é dividido em 10 intervalos, correspondendo a regiões de proximidade ao ponto de referência. Por fim, o indicador conta para cada algoritmo quantos pontos da $PF_{aprox}$ estão localizados em cada região de proximidade. Algoritmos que gerarem mais pontos em intervalos menores (10% ou 20%) mais próximos do ponto de referência, estão mais próximos deste ponto. Da mesma forma que no cálculo da distância de *Tchebycheff* essas distribuições são traçadas num histograma e é feita a mesma análise apresentada no exemplo anterior.

Dentre os indicadores apresentados nessa seção, essa tese irá focar nos valores de GD e IGD. O GD será utilizado para medir a convergência dos algoritmos utilizados, enquanto o IGD será utilizados para medir a diversidade. Os demais indicadores serão utilizados para auxiliar a análise do GD e do IGD.

Na validação das técnicas propostas, cada algoritmo é executado várias vezes e os resultados dos indicadores de qualidade são comparados utilizando o teste de Friedman [32]. O teste de Friedman é um teste estatístico não paramétrico utilizado para a comparação de múltiplos conjuntos de dados. Nesse teste as observações entre os blocos podem ser rankeadas. Por exemplo, o conjunto de execuções de cada algoritmo gera um bloco de dados para cada medida que é independente e sem interação, pois a cada algoritmo é executado separadamente, e só pode ser obtido um rank entre os valores das medidas. No teste não se utilizam os valores dos dados na comparação, mas sim os rankings obtidos

para cada amostra no conjunto de dados. A hipótese nula é que não existe diferença entre os algoritmos analisados. Assim, o teste mostra se há uma diferença entre os conjuntos de dados analisados. Em nossa análise ele é utilizado com 5% de nível de significância. O pós-teste é efetuado através de funções da ferramenta R [78]. É utilizado o pacote pgirmess da ferramenta R. O teste de Friedman é executado através da função *friedman.test* que recebe como parâmetro o conjunto de dados. O pós-teste é executado através da função *friedmanmc*. O pós-teste indica se há diferença estatística entre os diferentes blocos do conjuntos de dados, para identificar quais conjuntos de dados obtiveram os melhores valores são utilizados gráficos *boxplots*. Um *boxplot* é um gráfico utilizado para descrever grupos de dados. Dado um conjunto de dados, este gráfico apresenta a mediana, os quartis inferiores e superiores (vigésimo quinto e septuagésimo quinto percentis, respectivamente), os valores limites do conjunto de dados (maiores e menor valores) e os possíveis *outliers* do conjunto.

## 7.2 Análise empírica

Esta seção apresenta os resultados da análise empírica executada para a validação das novas estratégias propostas para trabalhar com problemas com muitos objetivos. As novas técnicas são aplicadas em algoritmos MOPSO e são medidos aspectos como convergência e diversidade da busca. As próximas seções apresentam a discussão dos resultados obtidos.

### 7.2.1 Avaliação do algoritmo CDAS-MOPSO

O primeiro conjunto de experimentos refere-se à aplicação da técnica Controle da Área de Dominância das Soluções no algoritmo SMPSO, chamado de CDAS-MOPSO (ver Algoritmo 4). A técnica CDAS necessita da definição do parâmetro que controla a área de dominância das soluções, ($S_i$). Neste conjunto de experimentos, ele é definido com 5 valores diferentes, obtendo-se 5 diferentes configurações do CDAS-MOPSO. O parâmetro $S_i$ variou em intervalos de $0,05$, a mesma variação utilizada em [81]. $S_i$ variou no intervalo de $[0,25;\ 0,45]$. Nessa primeira comparação, o algoritmo CDAS-MOPSO é comparado

Tabela 7.4: Algoritmos utilizados na avaliação do CDAS-MOPSO.

| **Algoritmo** | **Descrição** |
|---|---|
| SMPSO | Algoritmo SMPSO original |
| 0,25 | Algoritmo MOPSO com CDAS, $S_i = 0,25$ |
| 0,3 | Algoritmo MOPSO com CDAS, $S_i = 0,3$ |
| 0,35 | Algoritmo MOPSO com CDAS, $S_i = 0,35$ |
| 0,4 | Algoritmo MOPSO com CDAS, $S_i = 0,4$ |
| 0,45 | Algoritmo MOPSO com CDAS, $S_i = 0,45$ |

com o SMPSO [70]. A Tabela 7.4 apresenta um resumo dos algoritmos utilizados na análise do CDAS no MOPSO. Tanto o CDAS-MOPSO e SMPSO foram executados com 200 partículas na população e 200 soluções no arquivo externo. Os algoritmos foram executados um total de 50000 avaliações de fitness em 20 execuções independentes.

Nessa análise empírica do CDAS-SMPSO são efetuados dois estudos. O primeiro utiliza os indicadores de qualidade GD, IGD e *spacing* para medir a convergência e a diversidade da busca dos algoritmos. Além disso, para auxiliar a interpretação dos indicadores, também é considerado o número total de pontos da fronteira aproximada de cada algoritmo. São utilizados os problemas DTLZ2, DTLZ4, DTLZ6 e DTLZ7. Os números de funções objetivo utilizados foram: 3, 5, 10, 15 e 20.

O segundo conjunto de experimentos busca identificar se as fronteiras aproximadas geradas por cada algoritmo estão localizadas próximas ao joelho da fronteira de Pareto. Neste estudo é utilizado somente o problema DTLZ2 com 3, 5, 10, 15 e 20 funções objetivo.

Os resultados das melhores configurações obtidas pelo teste de Friedman para todos os problemas são apresentados na Tabela 7.5 e o número de pontos das $PF_{aprox}$ na Tabela 7.6. Para auxiliar a interpretação dos resultados do teste de Friedman, são apresentados gráficos que mostram para cada algoritmo o valor médio de cada indicador para todos os números de funções objetivo analisados. Através desses gráficos é possível perceber se há uma variação no valor de cada indicador quando o número de objetivos cresce. Esses gráficos são apresentados nas Figuras de 7.7 a 7.20.

O primeiro problema é o DTLZ2. Para medir a convergência inicialmente é observado o resultado do GD. Os melhores resultados foram obtidos por diferentes configurações do CDAS-MOPSO, porém as configurações $S_i = 0,3$ e $0,35$ se destacaram, pois obtiveram os

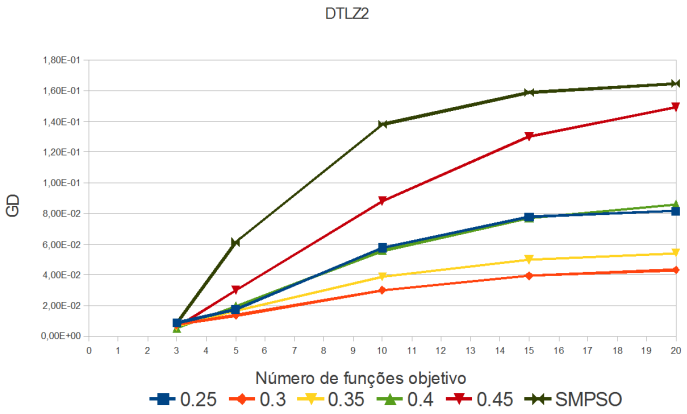



Figura 7.7: Valores médios de GD para o CDAS-SMPSO e SMPSO, DTLZ2.

melhores valores de GD para os maiores números de funções objetivo. O algoritmo SMPSO não conseguiu o melhor resultado em nenhum cenário. Observando a Figura 7.7, percebe-se que o valor do GD para SMPSO deteriorou quando o número de objetivos cresceu. Para o algoritmo CDAS-MOPSO essa deterioração é reduzida, sendo que as melhores configurações desse algoritmo obtiveram valores médios de GD semelhantes para poucas e muitas funções objetivo. Assim, para o problema DTLZ2 o algoritmo CDAS-MOPSO conseguiu melhores resultados em termos de convergência.

Através da medida IGD é possível observar diversidade na busca, porém também é possível identificar convergência. Para essa medida, novamente o CDAS-MOPSO se destacou. Todas as configurações conseguiram obter o melhor valor de IGD, de acordo com o teste de Friedman em pelo menos um cenário, exceto $S_i = 0,25$. Para essa medida a configuração $S_i = 0,35$ se destacou, obtendo bons valores tanto para poucas e muitas funções. Novamente a busca do SMPSO sofreu uma deterioração quando o número de objetivos cresceu. Para poucos objetivos, 3 funções, o SMPSO consegue ser competitivo, porém quando o número de objetivos cresceu o algoritmo perdeu desempenho também em termos de diversidade. É possível observar esse comportamento na Figura 7.8. Além disso, observa-se que o CDAS-MOPSO manteve um bom comportamento em termos de IGD para todos os números de funções. Outra característica que se pode destacar é o fato de algumas configurações do CDAS-MOPSO obterem um valor ruim de IGD para poucas funções. Isso é explicado através do comportamento do método CDAS, discutido

no Capítulo 4. Esse método limita a região de busca em pequenas partes do espaço, especialmente para valores muito baixos de $S_i$. Assim, a $PF_{aprox}$ gerada está localizada em uma pequena região, em geral, com um ou poucos pontos (ver Figura 4.5). Quando o número de objetivos cresce, o número de pontos não dominados gerados pelo CDAS também cresce e isso permite maior diversificação na busca.

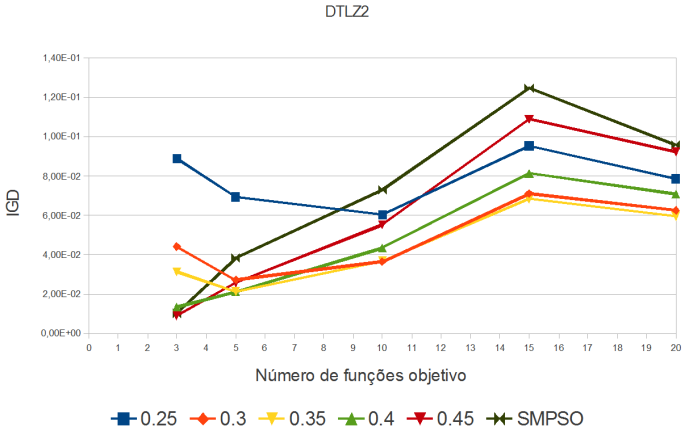



Figura 7.8: Valores médios de IGD para o CDAS-SMPSO e SMPSO, DTLZ2.

Por fim, o *spacing* é analisado com o objetivo de observar a distribuição entre as soluções geradas. O CDAS-MOPSO obteve a melhor distribuição, porém nesse caso ele foi motivado pelo baixo número de pontos gerados na $PF_{aprox}$. Quanto menor o número de pontos, o algoritmo controla mais facilmente essa distribuição. A Tabela 7.6 apresenta o número de pontos gerados na fronteira aproximada de cada algoritmo. As configurações que apresentaram os melhores valores de *spacing*, obtiveram o menor conjunto de pontos. Quando o conjunto de pontos é pequeno (um ou dois pontos) o valor do *spacing* é 0. Outra característica que pode ser observada é que os melhores valores de spacing foram obtidos pelas configurações com melhor GD. Assim, quanto menor é a área que o $PF_{aprox}$, melhor é o valor do *spacing*.

Em resumo, para o problema DTLZ2, que permite observar como um algoritmo escala o seu desempenho quando o número de objetivos cresce, o algoritmo CDAS-MOPSO apresentou os melhores resultados. Esse algoritmo obteve os melhores resultados tanto em termos de convergência quanto de diversidade. Além disso, não houve uma grande deterioração quando o número de objetivos cresceu. Por outro lado, o SMPSO sofreu uma

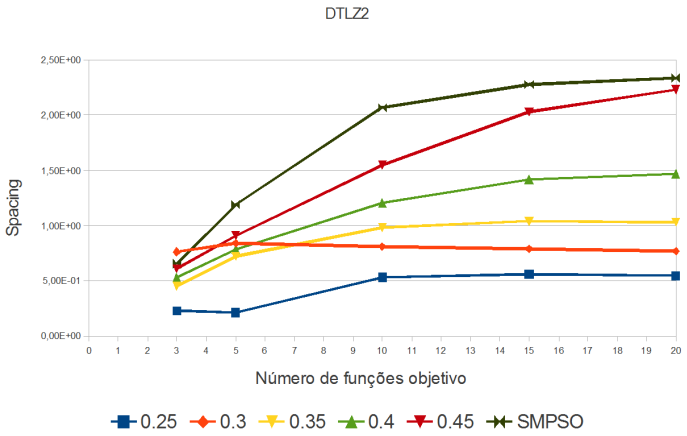



Figura 7.9: Valores médios de *spacing* para o CDAS-SMPSO e SMPSO, DTLZ2.

grande deterioração para um número de funções objetivo grande. Esse resultado afirma a suposição que MOEAs tradicionais enfrentam problemas na Otimização com Muitos Objetivos e que as técnicas propostas atenuam esses problemas.

Tabela 7.5: Melhores configurações do CDAS-SMPSO e SMPSO para todos os problemas e funções objetivos, de acordo com o pós-teste do teste de Friedman.

| **Prob** | **Obj** | **Melhores Configurações** | | |
|---|---|---|---|---|
| | | **GD** | **IGD** | ***Spacing*** |
| **DTLZ2** | **3** | 0,4 e 0,45 | 0,4 e 0,45 | 0,25, 0,35 e 0,4 |
| | **5** | 0,25, 0,3 e 0,35 | 0,35 e 0,4 | 0,25 e 0,35 |
| | **10** | 0,3 e 0,35 | 0,3, 0,35 e 0,4 | 0,25 e 0,3 |
| | **15** | 0,3 e 0,35 | 0,3 e 0,35 | 0,25 e 0,3 |
| | **20** | 0,3 e 0,35 | 0,3 e 0,35 | 0,25 e 0,3 |
| **DTLZ4** | **3** | 0,25 | 0,45 e SMPSO | 0,25, 0,3 |
| | **5** | 0,25 e 0,45 | 0,0,45 e SMPSO | 0,25 e 0,3 |
| | **10** | 0,25 | 0,4, 0,45 e SMPSO | 0,25 e 0,3 |
| | **15** | 0,25 e 0,45 | 0,4, 0,45 e SMPSO | 0,25 e 0,3 |
| | **20** | 0,25 e 0,45 | 0,35, 0,4 e 0,45 | 0,25 e 0,3 |
| **DTLZ6** | **3** | 0,3, 0,35, 0,4, 0,45 e SMPSO | 0,4, 0,45 e SMPSO | 0,25 e 0,3 |
| | **5** | 0,3, 0,35 e 0,4 | 0,4, 0,45 e SMPSO | 0,25 e 0,3 |
| | **10** | 0,3, 0,35 e 0,4 | 0,4 e 0,45 | 0,25 e 0,3 |
| | **15** | 0,3, 0,35 e 0,4 | 0,35, 0,4 e 0,45 | 0,3 e 0,35 |
| | **20** | 0,3 e 0,35 | 0,35, 0,4 e 0,45 | 0,3 e 0,35 |
| **DTLZ7** | **3** | 0,35, 0,4 e 0,45 | 0,45 e SMPSO | 0,25, 0,3 e 0,35 |
| | **5** | 0,45 e SMPSO | 0,45 e SMPSO | 0,25, 0,3 e 0,35 |
| | **10** | 0,45 e SMPSO | 0,45 e SMPSO | 0,25, 0,3 e 0,35 |
| | **15** | 0,45 e SMPSO | 0,45 e SMPSO | 0,25, 0,3 e 0,35 |
| | **20** | 0,45 e SMPSO | 0,45 e SMPSO | 0,25, 0,3 e 0,35 |

O segundo problema é o DTLZ4. Esse problema apresenta uma região da fronteira de Pareto mais povoada, que atrai a busca de MOEAs. Assim, quanto melhor é a diversidade da busca de um algoritmo, melhor é o desempenho desse algoritmo no DLTZ4. Os melhores algoritmos de acordo com o teste de Friedman para cada indicador de qualidade são apresentados na Tabela 7.5. Observando o GD, o CDAS-MOPSO se destaca, obtendo o melhor valor em todos os cenários, em especial para configuração com $S_i = 0,25$. Porém,

Tabela 7.6: Número médio de pontos na $PF_{aprox}$ geradas pelos algoritmos CDAS-MOPSO e SMPSO.

| Prob | Obj | Configurações | | | | | |
|---|---|---|---|---|---|---|---|
| | | **0,25** | **0,3** | **0,35** | **0,4** | **0,45** | **SMPSO** |
| **DTLZ2** | **3** | 4,15 | 16,5 | 75,3 | 198,9 | 199,7 | 199,6 |
| | **5** | 11,3 | 142,7 | 197,25 | 199,25 | 199,9 | 199,2 |
| | **10** | 30,05 | 195,3 | 199,7 | 199,7 | 199,85,0 | 200,0 |
| | **15** | 47,75 | 196,85 | 199,7 | 200,00 | 199,9 | 200,0 |
| | **20** | 48,05 | 199,35 | 198,85,7 | 197,25 | 199,9 | 200,0 |
| **DTLZ4** | **3** | 1,00 | 3,95 | 9,75 | 22,00 | 40,20 | 60,05 |
| | **5** | 1,00 | 6,40 | 15,75 | 29,60 | 52,95 | 74,00 |
| | **10** | 1,00 | 18,15 | 32,40 | 54,05 | 87,70 | 159,75 |
| | **15** | 1,00 | 25,99 | 49,60 | 83,85 | 142,50 | 198,65 |
| | **20** | 1,00 | 33,80 | 73,40 | 124,00 | 197,35 | 199,60 |
| **DTLZ6** | **3** | 1,10 | 199,85 | 200,00 | 200,00 | 200,00 | 200,00 |
| | **5** | 18,20 | 199,55 | 200,00 | 200,00 | 200,00 | 199,55 |
| | **10** | 8,40 | 200,00 | 200,00 | 200,00 | 200,00 | 189,90 |
| | **15** | 1,40 | 200,00 | 200,00 | 200,00 | 200,00 | 198,45 |
| | **20** | 15,80 | 199,85 | 199,65 | 200,00 | 200,00 | 197,80 |
| **DTLZ7** | **3** | 4,90 | 8,90 | 22,80 | 93,40 | 200,00 | 200,00 |
| | **5** | 3,70 | 4,30 | 10,60 | 30,20 | 199,10 | 200,00 |
| | **10** | 2,10 | 3,60 | 6,40 | 22,10 | 175,60 | 200,00 |
| | **15** | 2,20 | 3,10 | 6,30 | 12,10 | 101,30 | 200,00 |
| | **20** | 2,50 | 2,80 | 5,10 | 11,70 | 58,00 | 200,00 |

isso ocorreu devido ao pequeno número de soluções no conjunto de aproximação, observado a Tabela 7.6. Essa configuração gerou somente um ponto, em todos os números de funções objetivo. Assim, esse ponto conseguiu convergir, porém a cobertura da fronteira é praticamente nula. A configuração com $S_i = 0,45$ conseguiu bons resultados de GD em alguns números de objetivos e gerou um conjunto de pontos maior. A Figura 7.10 mostra o comportamento do GD. Para $S_i = 0,25$ o valor é sempre o mesmo, praticamente igual a zero, ou seja, o ponto alcançou a fronteira. Para as demais configurações, nota-se a deterioração do SMPSO e os melhores resultados obtidos pelas demais configurações do CDAS-MOPSO.

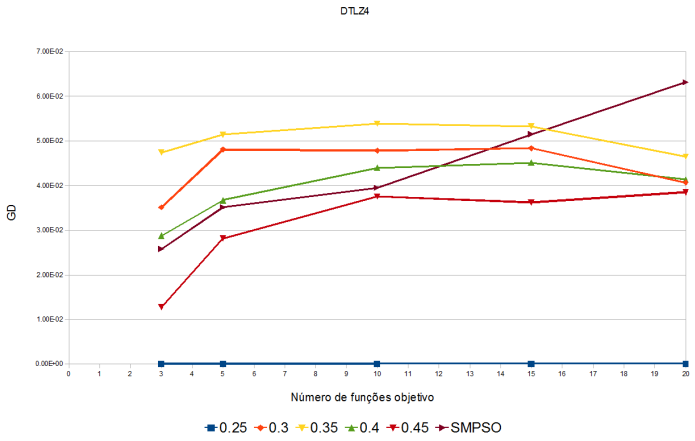



Figura 7.10: Valores médios de GD para o CDAS-SMPSO e SMPSO, DTLZ4.

O IGD é a medida mais importante para a análise do DTLZ4. Um algoritmo que possui ótimo valor de GD, porém apresenta um valor ruim de IGD, mostra que houve convergência, mas a busca ficou presa em uma pequena região da fronteira e não houve diversidade. Nesse sentido, a configuração com $S_i = 0,25$ obteve um resultado abaixo dos demais, justamente por produzir somente uma solução. Para essa medida, o SMPSO e a configuração $S_i = 0,45$ do CDAS-MOPSO obtiveram os melhores resultados para a maioria dos cenários, de acordo com o teste de Friedman. Esse comportamento pode ser visto na Figura 7.11. Para todos os números de funções objetivo, a deterioração é semelhante tanto para o CDAS-MOPSO quanto para SMPSO, exceto para 20 funções que o SMPSO apresentou um resultado pior. Embora alguns algoritmos tenham se destacados significativamente no teste de Friedman, os valores de IGD são semelhantes entre todos os métodos. Logo, em geral todos os algoritmos obtiveram um comportamento semelhante em termos de diversidade. Além disso, o valor de IGD da configuração com $S_i = 0,25$ não é muito menor que os demais algoritmos. Logo, isso sugere que mesmo gerando mais soluções, todos os algoritmos ficaram limitados numa mesma região da fronteira de Pareto.

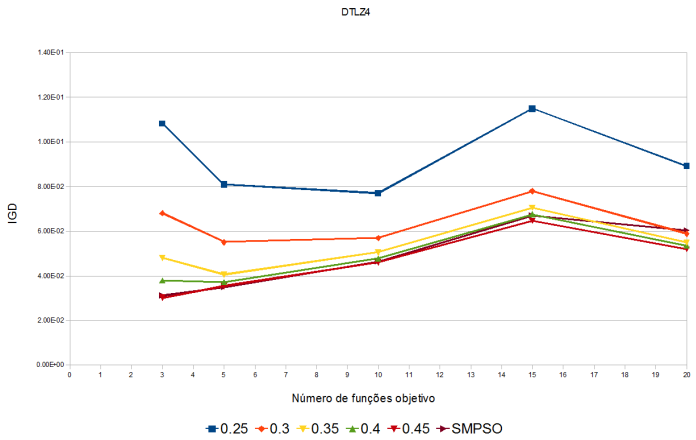



Figura 7.11: Valores médios de IGD para o CDAS-SMPSO e SMPSO, DTLZ4.

A Figura 7.12 mostra um esboço da $PF_{aprox}$ gerada pelo algoritmo SMPSO para 3 funções objetivo. A região da fronteira real está demarcada e os pontos do $PF_{aprox}$ são indicados por triângulos. Na figura percebe-se que a maioria dos pontos ficou presa em uma pequena parte da fronteira, próximos aos limites da esfera. Há alguns pontos espalhados por outras regiões, porém a maioria dos pontos concentram-se em uma mesma

região.

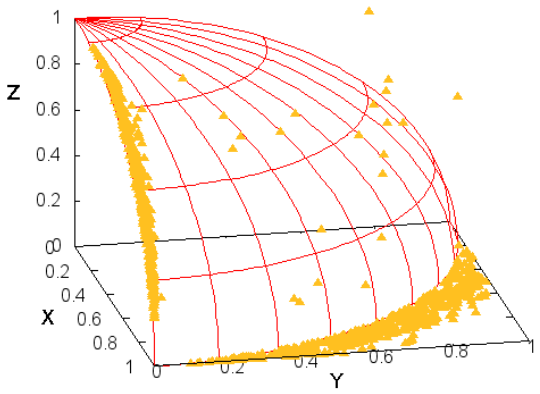



Figura 7.12: $PF_{aprox}$ gerada pelo algoritmo SMPSO sobre a fronteira de Pareto do DTLZ4.

Para o DTLZ4, o valor do *spacing* acabou sendo bastante influenciado pelo número de pontos gerados pelo algoritmo CDAS-MOPSO. As duas configurações com menor valor de $S_i$, $S_i = 0,25$ e $0,3$ obtiveram um conjunto muito pequeno de pontos na $PF_{aprox}$, logo isso gerou o melhor valor de spacing para todos os números de funções objetivo. Esse bom valor de *spacing* pode ser observado na Figura 7.13. A configuração $S_i = 0,25$ obteve o mesmo valor de *spacing* para todos os cenários. Os demais algoritmos sofreram deterioração na busca, porém o SMPSO apresenta os piores valores.

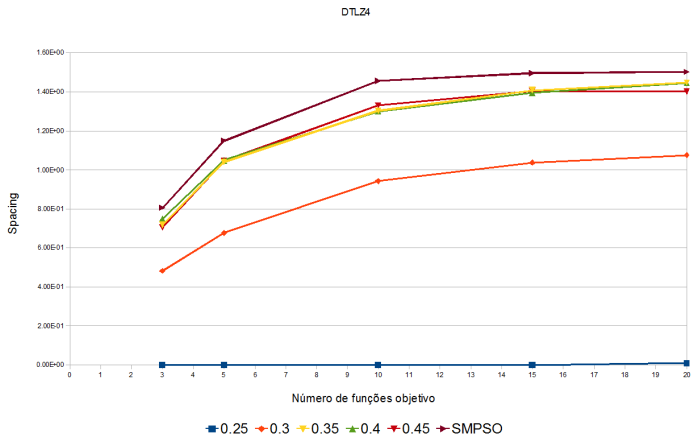



Figura 7.13: Valores médios de *spacing* para o CDAS-SMPSO e SMPSO, DTLZ4.

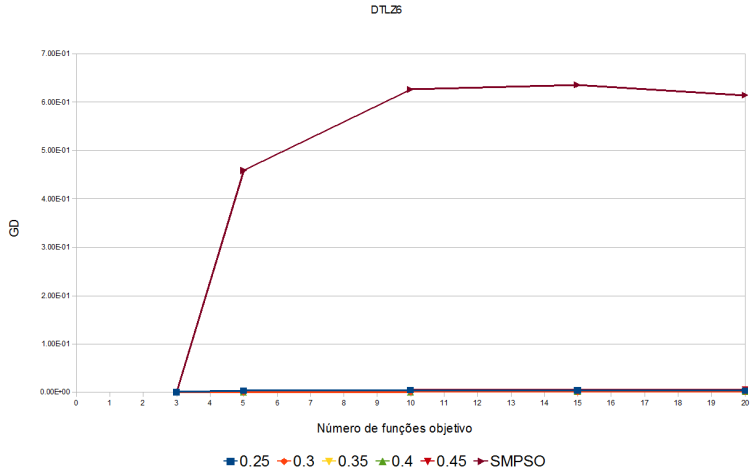
Em resumo, todos os algoritmo enfrentaram dificuldades no problema DTLZ4. Quem mais se destacou foi a configuração $S_i = 0,45$ do CDAS-MOPSO, que obteve os melhores

resultados de GD e IGD. Porém, os algoritmos não conseguiram obter um bom resultado em termos de diversidade e a busca ficou estacionada na região mais densa do espaço de objetivos. Os algoritmos SMPSO e $S_i = 0,45$ do CDAS-MOPSO ainda conseguiram espalhar mais suas soluções pelo espaço de busca, porém, em geral, houve uma concentração muito grande dessas soluções em uma pequena região.

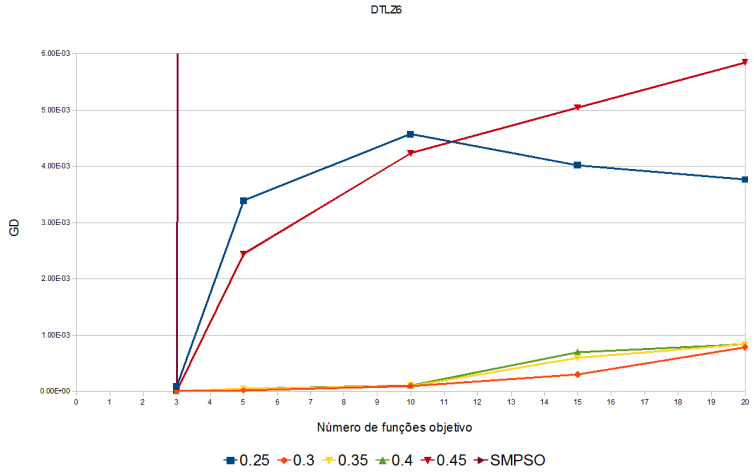
O problema DTLZ6 apresenta uma fronteira de Pareto mais simples que os problemas anteriores, somente uma curva e não toda a esfera, porém utiliza uma função de dificuldade que introduz diversos múltiplos locais. Para esse problema o CDAS-MOPSO se destacou, obtendo bons resultados para todos os indicadores. Os melhores algoritmos de acordo com o teste de Friedman são apresentados na Tabela 7.5. Para o GD, as configurações $S_i = 0,30$ e $0,35$ se destacaram obtendo o melhor valor em todos os números de funções objetivo. O SMPSO conseguiu ser equivalente ao CDAS-MOPSO para 3 funções objetivos, porém não conseguiu manter um bom resultado quando o número de objetivos cresceu. A Figura 7.14 mostra uma resumo desse comportamento. O SMPSO sofreu uma grande deterioração na sua busca quando foi submetido a cenários com muitos objetivos. Por outro lado, o CDAS-MOPSO conseguiu manter valores semelhantes de GD para todos os números de objetivos. A Figura 7.14 apresenta os resultados do GD em duas diferentes escala, pois os resultados do SMPSO crescem muito em relação aos do CDAS-MOPSO. Na Figura 7.14(b), percebe-se os bons valores de GD de algumas configurações do CDAS-MOPSO para todos os números de funções objetivo.

Para o IGD, novamente o CDAS-MOPSO apresentou bons resultados. As configurações $S_i = 0,4$ e $0,45$ obtiveram os melhores resultados para todos os números de objetivo. O SMPSO conseguiu bons resultados para poucas funções, porém perdeu desempenho para muitos objetivos. A Figura 7.15 mostra o comportamento do IGD. A partir de 5 funções objetivo, o SMPSO obteve uma piora no IGD e sofreu uma deterioração na diversidade da busca. Por outro lado, O CDAS-MOPSO conseguiu manter um valor semelhante de IGD em todos os cenários.

Por fim, o CDAS-MOPSO obtém os melhores valores de *spacing*. A configuração $S_i = 0,3$ obteve o melhor resultado em todos os números de funções objetivo. Pode




(a) Escala mais ampla




(b) Escala mais reduzida

Figura 7.14: Valores médios de GD para o CDAS-SMPSO e SMPSO, DTLZ6, com duas diferentes escalas.

ser destacado que essa configuração gerou um número de pontos na $PF_{aprox}$ semelhante ao número dos demais algoritmos, como visto na Tabela 7.6. A Figura 7.16 mostra que o SMPSO novamente sofreu uma deterioração na sua busca quando o número de objetivos cresceu. As demais configurações do CDAS-MOPSO conseguiram controlar a deterioração, obtendo valores semelhantes de spacing para todos os números de funções objetivo.

Em resumo, o CDAS-MOPSO obteve um resultado muito bom para o problema DTLZ6. Esse algoritmo não sofreu deterioração tanto em termos de convergência, quanto em diversidade. Pode-se destacar a configuração com $S_i = 0,4$ que conseguiu bons resul-

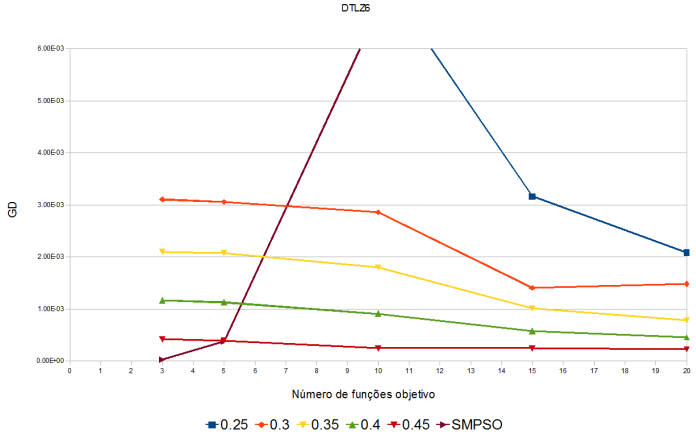



Figura 7.15: Valores médios de IGD para o CDAS-SMPSO e SMPSO, DTLZ6.

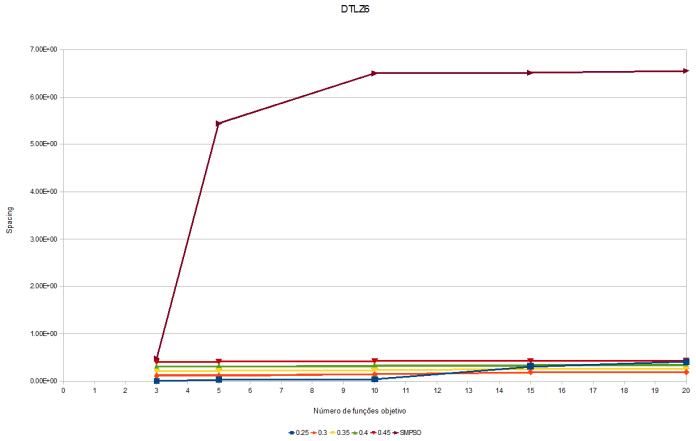
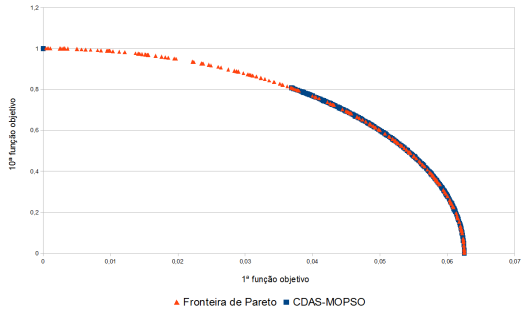



Figura 7.16: Valores médios de *spacing* para o CDAS-SMPSO e SMPSO, DTLZ6.

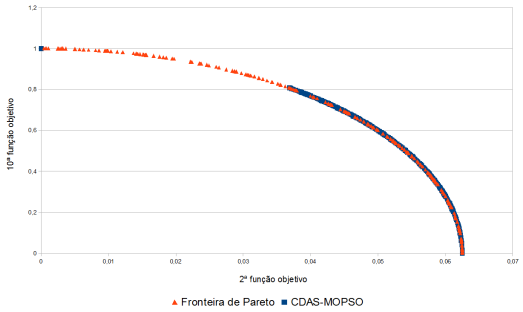
tados de GD e IGD, além de não sofrer uma deterioração grande do *spacing*. Uma característica do DTLZ6 é o fato que sua fronteira pode ser visualizada através da combinação das diferentes funções objetivo. Para a fronteira de Pareto desse problema, qualquer conjunto de valores gerados por uma função objetivo quando esboçados com os valores da última função objetivo gera uma curva.

A Figura 7.17 mostra a $PF_{aprox}$ gerada pelo algoritmo CDAS-MOPSO com $S_i = 0,4$ (obtido da primeira execução do algoritmo) e a fronteira de Pareto (gerada com 200 pontos) para 10 funções objetivo. Por clareza, só são mostradas as curvas das três primeiras funções objetivo, porém todas as demais funções seguem o mesmo comportamento nesse exemplo. A fronteira de Pareto é a curva completa, que alcança todo intervalo, enquanto a $PF_{aprox}$ cobre apenas uma parte dessa curva. Através dessas curvas percebe-se que o

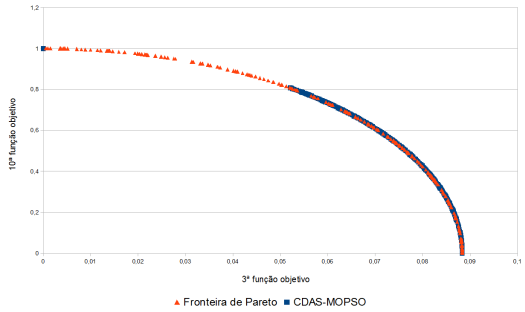
CDAS-MOPSO alcança uma região da fronteira de Pareto e cobre uma grande parte dessa fronteira. A parte não coberta é resultado do método CDAS, que privilegia a geração de pontos em algumas regiões do espaço de objetivos.




(a) 1 x 10




(b) 2 x 10




(c) 3 x 10

Figura 7.17: Gráfico da $PF_{aprox}$ e $PF_{real}$ para o DTLZ6 com 10 objetivos (somente três funções).

O último problema é o DTLZ7 que tem como característica possuir uma fronteira de Pareto desconexa. Para esse problema CDAS-MOPSO não conseguiu obter um bom resultado. Os resultados do teste de Friedman são apresentados na Tabela 7.5. Observando o GD, a configuração com $S_i = 0,45$ obteve um resultado equivalente ao SMPSO na maioria de números de funções objetivo. As demais configurações não obtiveram bons resultados. A Figura 7.18 mostra a média do GD para todos os algoritmos. O algoritmo CDAS-MOPSO sofreu uma deterioração maior do que o SMPSO, especialmente para valores de $S_i$ pequenos. Além disso, apesar de os algoritmos serem equivalentes estatisticamente a diferença entre as médias aumentou quando o número de objetivos cresceu, sendo que o SMPSO teve os melhores resultados.

Para o IGD o resultado é semelhante ao GD. O SMPSO e a configuração $S_i = 0,45$

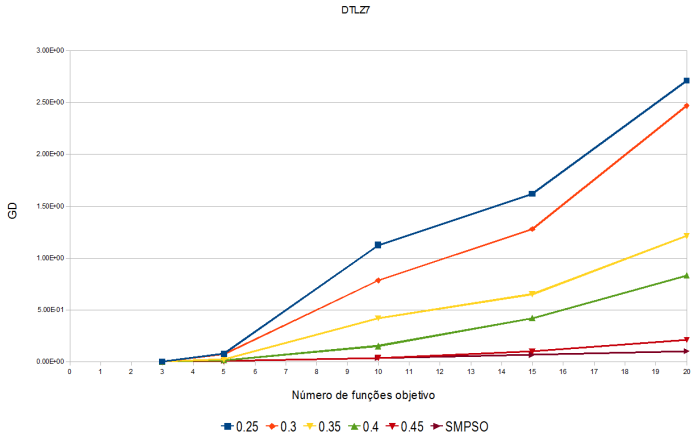



Figura 7.18: Valores médios de GD para o CDAS-SMPSO e SMPSO, DTLZ7.

do CDAS-MOPSO são equivalentes para todos os números de funções objetivo. Porém observando as Figura 7.19, os resultados do CDAS-MOPSO pioram em relação ao SMPSO. Novamente, as configurações com baixo valor de $S_i$ obtiveram os piores resultados. Esse mau comportamento do CDAS-MOPSO em termos de convergência e diversidade pode ser observado também no número de pontos gerado pelo algoritmo na Tabela 7.6. Quando o número de objetivos cresce o algoritmo passa a gerar $PF_{aprox}$ menores. Isso mostra, que a limitação do espaço efetuado pelo CDAS não é sempre eficiente, pois em alguns casos pode limitar a busca numa região que não seja válida para o problema.

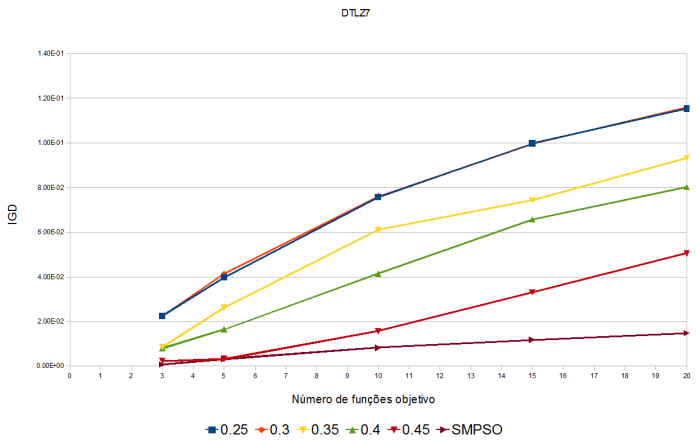



Figura 7.19: Valores médios de IGD para o CDAS-SMPSO e SMPSO, DTLZ7.

O *spacing* é definido de forma semelhante aos problemas DTLZ2 e DTLZ4. Os melhores resultados foram obtidos pelos algoritmos que geraram menos pontos. As configurações $S_i = 0,25$, $0,3$ e $0,35$ obtiveram os melhores resultados para todos os números de funções

objetivo. Como nos demais problemas, quanto menor é o número de pontos, melhor é o espaçamento do conjunto (ver Figura 7.20).

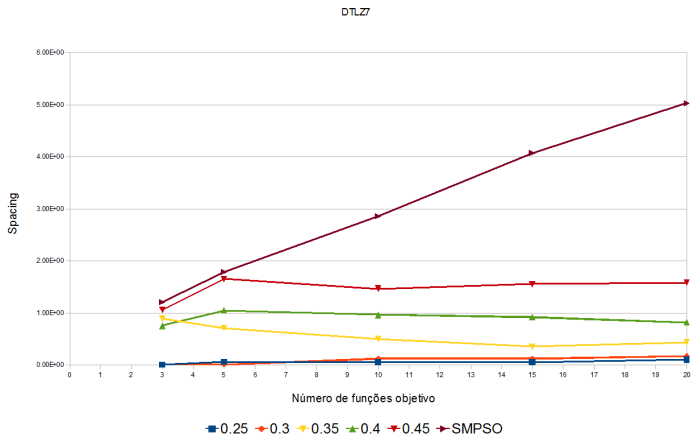



Figura 7.20: Valores médios de *spacing* para o CDAS-SMPSO e SMPSO, DTLZ7.

Em resumo, o CDAS-MOPSO não apresentou um bom resultado para o problema DTLZ7, especialmente em cenários com muitos objetivos. Quando o número de objetivos aumentou, o CDAS-MOPSO sofreu uma grande deterioração na busca, tanto em termos de convergência quando em diversidade. O fato de a técnica CDAS limitar a área de exploração da busca não foi eficiente para um problema em que a fronteira de Pareto é desconexa. Em geral, o SMPSO apresentou os melhores resultados, porém não se tem a garantia que ele obtém bons resultados para o DTLZ7. Para todas as medidas o SMPSO apresentou uma deterioração, além disso, a escala dos resultados dos indicadores obtidos pelo SMPSO é alta quando comparado aos resultados do CDAS-MOPSO em outros problemas. Isso indica que o SMPSO foi melhor que o CDAS-MOPSO no DTLZ7, mas que não se aproximou tanto da fronteira quanto o CDAS-MOPSO faz nos demais problemas.

Observando todos os problemas analisados, conclui-se que o CDAS-MOPSO apresenta resultados bons para Problemas com Muitos Objetivos, porém pode enfrentar algumas dificuldades dependendo da natureza do problema. Em geral, para um problema simples como o DTLZ2 o CDAS-MOPSO apresenta bons resultados tanto em termos de convergência quanto de diversidade. Além disso, o algoritmo não apresentou dificuldades em resolver problemas com múltiplos ótimos locais, obtendo inclusive os melhores resultados

em termos de diversidade nesse contexto. O problema DTLZ4 apresentou dificuldade na diversificação da busca de todos os algoritmos. Todos os algoritmos testados ficaram limitados na região mais densa da fronteira. Por fim, o DTLZ7 apresentou os maiores desafios ao CDAS-MOPSO. A técnica CDAS não se mostrou eficiente nesse problema e o algoritmo apresentou resultados ruins. Logo, o CDAS-MOPSO não é aplicável em problemas com fronteiras desconexas, especialmente com muitas funções objetivo.

A segunda etapa da análise do CDAS-MOPSO visa identificar se o conjunto de aproximação gerado está localizado próximo ao joelho da fronteira de Pareto [24]. Nessa etapa só foi utilizado o problema DTLZ2. A Figura 7.21 apresenta as distribuições da distância de Tchebycheff para todos os números de funções objetivo, para as configurações do CDAS-MOPSO para o SMPSO, $S_i = 0,5$. Para a geração dos gráficos de distribuição de cada configuração, foi calculada a distância de Tchebycheff de todas as soluções geradas (soluções de todas as execuções). Essas distâncias foram arredondadas para uma casa decimal e os gráficos com as distribuições mostram o número total de soluções normalizado entre 0 e 1.

Nesta análise, algoritmos que geram mais soluções em volta do joelho são preferíveis. Através destas distribuições é possível observar se as soluções se concentram próximo ao joelho (distribuição concentrada em valores baixo da distância) ou se as soluções são um conjunto mais diversificado. Para se ter uma base de comparação, em todos os gráficos a distribuição do SMPSO ($S_i = 0,5$) é também traçada.

Para o CDAS-MOPSO, quando o valor de $S_i$ é muito baixo o algoritmo converge para uma região pequena da fronteira de Pareto, porém nem sempre próxima ao joelho. As distribuições das configurações $S_i = 0,25$ e $0,3$ são representadas por somente um ponto ou uma pequena curva, ou seja, todas as soluções estão concentradas em uma mesma distância em relação ao joelho. As demais configurações $S_i = 0,35$, $0,4$ e $0,45$, concentram quase todas suas soluções em uma distância próxima ao joelho, mesmo para um número alto de objetivos. Para o SMPSO ($S_i = 0,5$), em todos os números de objetivos, é obtido uma $PF_{aprox}$ mas diversificada sobre o joelho da busca. Em todos os casos, a curva do SMPSO é mais alongada, alcançando diferentes valores da distância, sem a existência de

Tabela 7.7: Algoritmos utilizados na comparação entre os métodos de ranking.

| Algoritmo | Descrição |
|---|---|
| DP | Algoritmo SMPSO. |
| AR | Algoritmo Ranking-SMPSO com ranking AR. |
| MR | Algoritmo Ranking-SMPSO com ranking MR. |
| BR | Algoritmo Ranking-SMPSO com ranking BR. |
| AR_BR | Algoritmo SMPSO com a combinação AR e BR. |
| BR_MR | Algoritmo SMPSO com a combinação BR e MR. |

grandes picos em poucos valores.

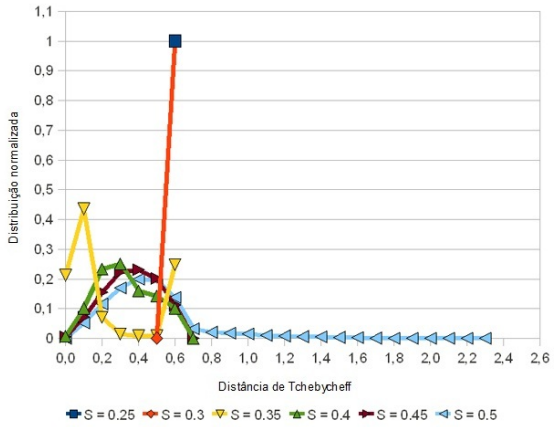
### 7.2.2 Comparação dos métodos de ranking e a Combinação de Rankings

O segundo conjunto de experimentos visa validar os métodos de ranking propostos no Capítulo 4, *Balanced Ranking* e a Combinação de Rankings. Da mesma forma que no conjunto de experimentos anterior, o objetivo é executar os algoritmos em diferentes problemas com muitos objetivos e observar aspectos como convergência e diversidade da busca. Nesse conjunto de experimentos são utilizados os problemas DTLZ2 e DTLZ4, o primeiro para medir a escalabilidade dos algoritmos e o segundo para medir a habilidade desses algoritmos em diversificar a busca, com 2, 3, 5, 10, 15 e 20 funções objetivos. Para observar os aspectos de convergência e diversidade foram utilizados os indicadores GD, IGD e *spacing*. Os algoritmos foram executados com 200 partículas na população, 200 soluções no repositório e o critério de para foi o número de gerações, definido com 100 iterações. Cada algoritmo foi executado 30 vezes. Os resultados desse conjuntos de experimentos foram publicados em [8].

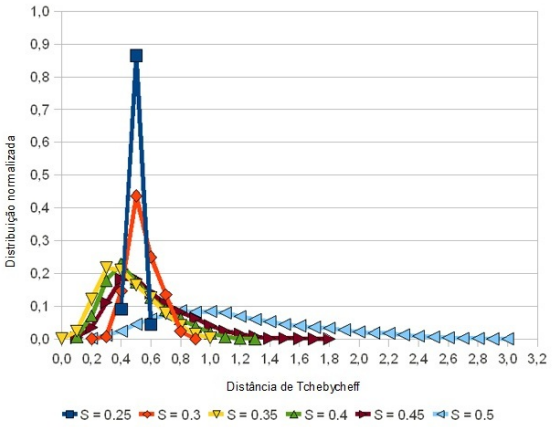
O *Balanced Ranking* e a Combinação de Rankings são comparados com diferentes algoritmos da literatura. A Tabela 7.7 apresenta um resumo dos algoritmos utilizados nessa análise.

A Tabela 7.8 e as Figuras 7.22 e 7.24 apresentam um resumo das comparações efetuadas. As Figuras mostram para cada método o valor médio de cada indicador para cada objetivo analisado. Para facilitar na interpretação do teste de Friedman são apresentados alguns *boxplots* com os melhores rankings obtidos, Figuras 7.23 e 7.25.

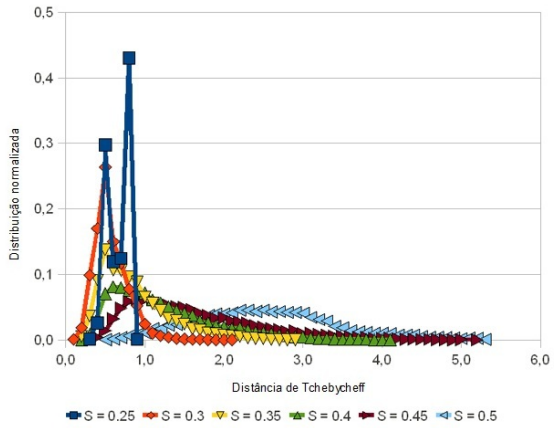
O primeiro problema analisado é o DTLZ2. Observando os valores de GD, percebe-



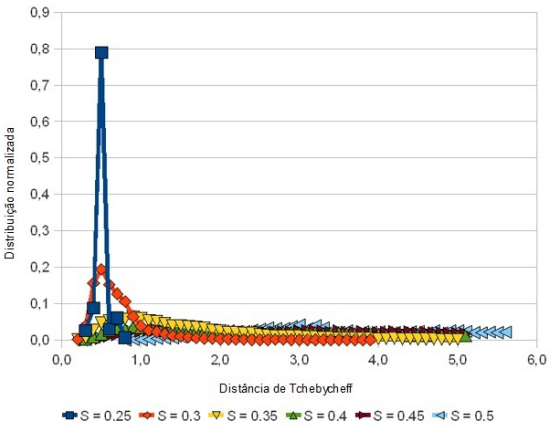

(a) 3 objetivos



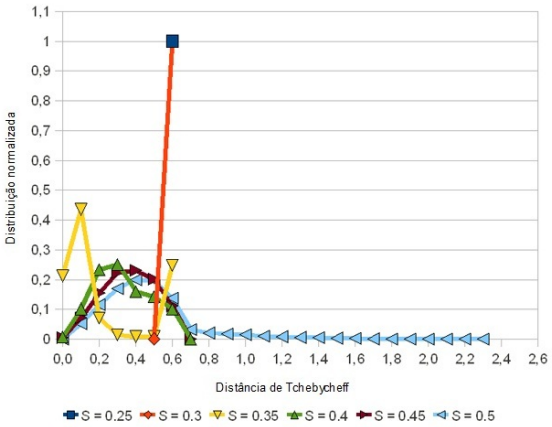

(b) 5 objetivos




(c) 10 objetivos




(d) 15 objetivos




(e) 20 objetivos

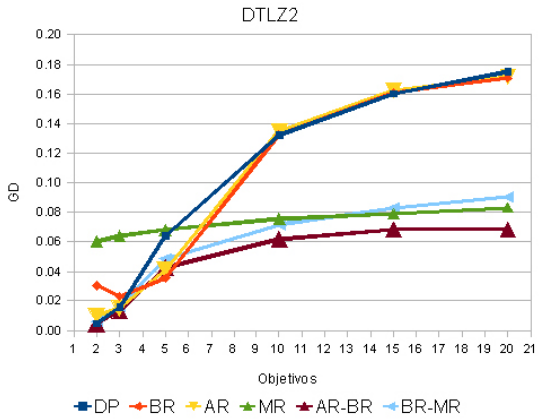
Figura 7.21: Distribuição da distância de Tchebycheff para o CDAS-MOPSO e SMPSO.

Tabela 7.8: Melhores métodos de ranking para cada objetivo e cada problema de acordo com o pós-teste do teste de Friedman.

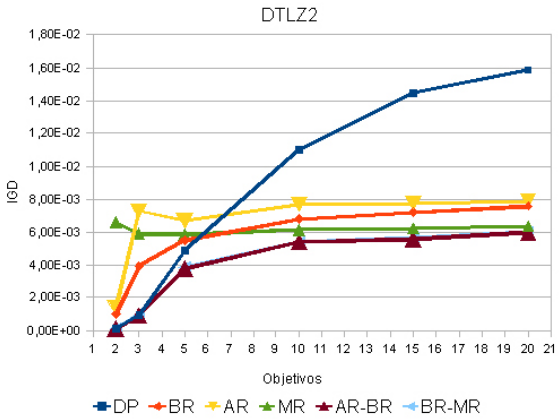
| Prob. | Obj. | Melhores métodos | | |
|---|---|---|---|---|
| | | GD | IGD | Spacing |
| DTLZ2 | 2 | DP, AR_BR, BR_MR | DP, AR_BR, BR_MR | DP, AR, MR, AR_BR, BR_MR |
| | 3 | DP, AR, AR_BR | DP, AR_BR, BR_MR | AR, BR |
| | 5 | AR, BR | AR_BR, BR_MR | AR, BR |
| | 10 | AR_BR, BR_MR | AR_BR, BR_MR | MR, AR_BR |
| | 15 | MR, AR_BR | MR, AR_BR, BR_MR | MR, AR_BR |
| | 20 | MR, AR_BR | MR, AR_BR, BR_MR | MR, AR_BR |
| DTLZ4 | 2 | BR | BR, AR_BR | BR, MR |
| | 3 | Todos | Todos | AR, BR, MR, AR_BR, BR_MR |
| | 5 | AR, BR, AR_BR | DP, AR, BR, AR_BR | AR, BR, AR_BR |
| | 10 | AR, BR, AR_BR | AR, BR, AR_BR | AR, BR, AR_BR |
| | 15 | AR, BR, AR_BR | AR, BR, AR_BR | AR, BR, AR_BR |
| | 20 | AR, BR, AR_BR | AR, BR, AR_BR | AR, BR, AR_BR |

se na Tabela 7.8 que a combinação de rankings obteve o melhor resultado para quase todos os objetivos. Tanto a combinação AR_BR quanto a BR_MR apresentaram bons resultados, sendo que a primeira foi melhor para um número maior de objetivos. Outro método que se pode destacar é o MR, que também obteve bons resultados para muitos objetivos. O boxplot da Figura 7.23(a) mostra que para 20 objetivos essas técnicas tiveram resultados equivalentes. O BR não obteve bons resultados para muitos objetivos, somente foi melhor para 5 objetivos. O algoritmo SMPSO com a dominância de Pareto (DP) conseguiu obter os melhores resultados para 2 e 3 objetivos, mas não conseguiu obter bons resultados quando o número de objetivos cresceu. A Figura 7.51(a) mostra um resumo desse comportamento. As curvas que representam os métodos de combinação de rankings e do MR têm um crescimento pouco acentuado quando o número de objetivos aumenta, obtendo bons valores de GD para todos os objetivos. Para os demais métodos há uma inclinação acentuada quando o número de objetivos cresce, ou seja, os algoritmos têm a busca deteriorada quando o número de objetivos cresce e não conseguem convergir para a fronteira de Pareto real.

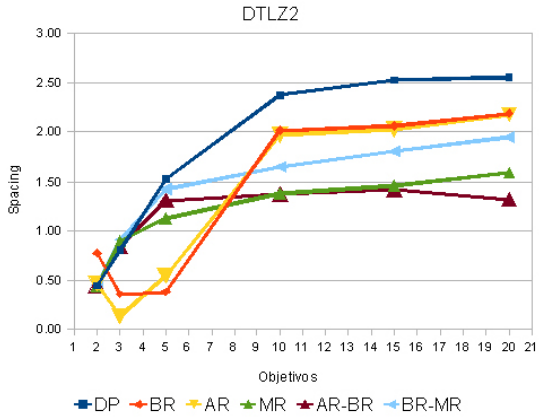
Para o IGD, o resultado é semelhante. A combinação de rankings (AR_BR e BR_MR) e o MR obtiveram os melhores resultados quando o número de objetivos cresceu (ver boxplot na Figura 7.23(b)). Novamente o BR não conseguiu bons resultados e não atingiu o melhor IGD em nenhum objetivo. O DP apresentou o mesmo comportamento, bons valores somente com poucos objetivos. Observando a Figura 7.22(b), nota-se o comportamento discutido anteriormente. Um crescimento pouco acentuado para os métodos de AR_BR,




(a) GD

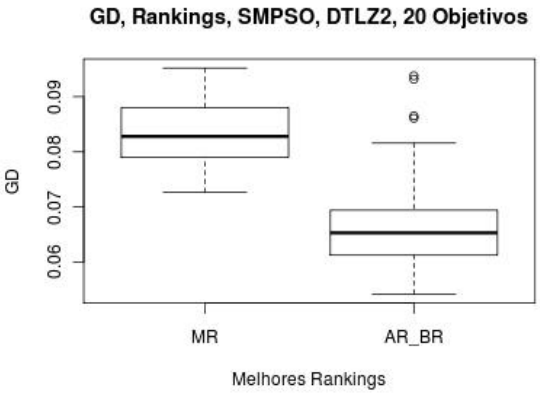



(b) IGD

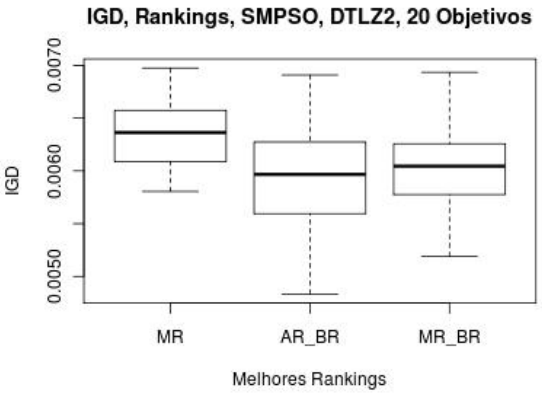



(c) *Spacing*

Figura 7.22: Média das medidas GD e IGD para o problema DTLZ2, para os métodos de ranking.




(a) GD




(b) IGD

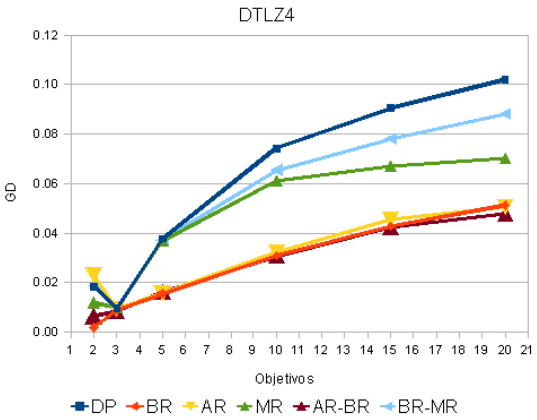
Figura 7.23: Boxplot com os melhores rankings para o GD e IGD. Problema DTLZ2 com 20 objetivos.

BR_MR e MR e uma alta deterioração para o SMPSO. Apesar de os métodos AR e BR não apresentarem os melhores resultados, o IGD destes métodos não deteriorou muito quando o número de objetivos cresceu. Assim, através do IGD, pode-se concluir que os métodos da combinação de rankings e o MR, além de terem uma boa convergência para a fronteira de Pareto real, conseguem produzir um conjunto de aproximação diversificado em torno desta fronteira, quando o número de objetivos cresce. Para o método BR, não houve uma grande deterioração da convergência e da diversidade quando o número de objetivos foi alto, porém em todos os casos o algoritmo teve dificuldade para convergir.

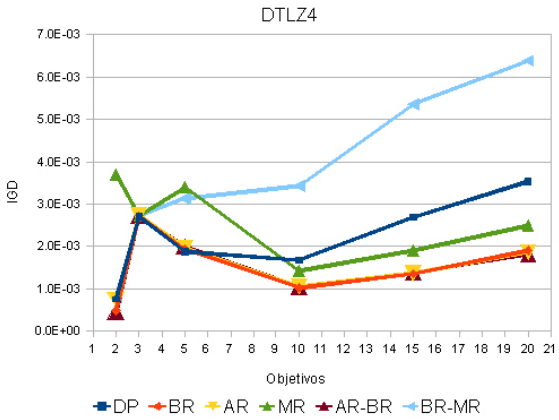
Na análise do *spacing*, os resultados também são semelhantes. A combinação de rankings e o MR obtiveram os melhores resultados para muitos objetivos. Quando o número de objetivos foi baixo, quase todos os métodos foram competitivos, em especial os métodos AR e BR. A Figura 7.22(c) mostra um resumo deste comportamento. A partir de 10 objetivos, os métodos que possuíam bons valores de *spacing* apresentam uma deterioração, enquanto os métodos MR e AR_BR apresentam valores constantes de *spacing* para todos os números de objetivos.

O segundo problema, DTLZ4, é utilizado para observar a habilidade dos algoritmos em manter uma boa distribuição das soluções. Neste problema a medida IGD é mais importante, pois também mede diversidade. Para o GD os métodos AR, BR e a combinação AR_BR obtiveram os melhores resultados. O *boxplot* da Figura 7.25(a) mostra o resultado equivalente dessas técnicas para 15 objetivos. Diferentemente do DTLZ2, neste problema os métodos AR e BR não tiveram sua convergência deteriorada quando número de objetivos cresceu. A Figura 7.24(a) mostra o resumo da evolução do GD quando o número de objetivos cresce. Novamente nota-se um bom resultado da combinação de rankings através do método AR_BR, que manteve um aumento suave quando o número de objetivos cresceu. Os métodos AR e BR apresentam um comportamento semelhante. Para o DP, novamente, há uma deterioração grande na convergência quando o número de objetivos é alto.

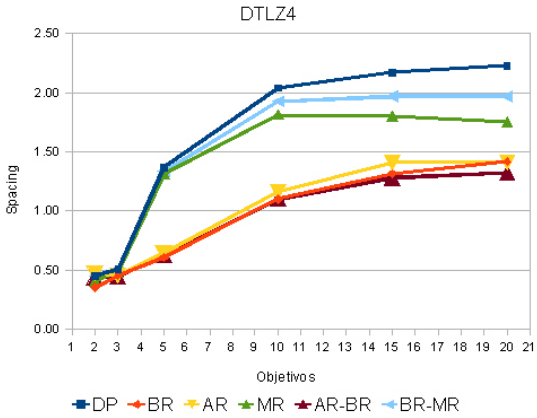
O IGD apresenta um resultado bastante semelhante ao GD para o DTLZ4. Os melhores resultados para muitos objetivos foram obtidos pelos métodos AR, BR e a combinação




(a) GD

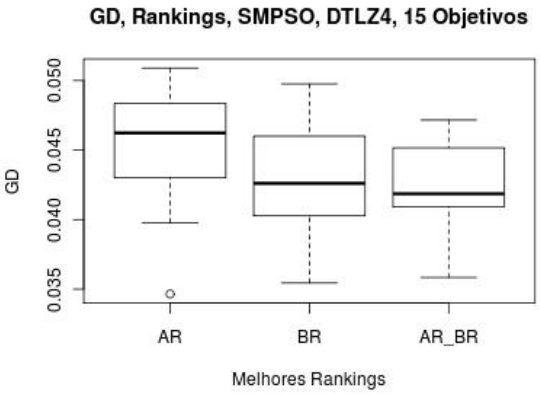



(b) IGD

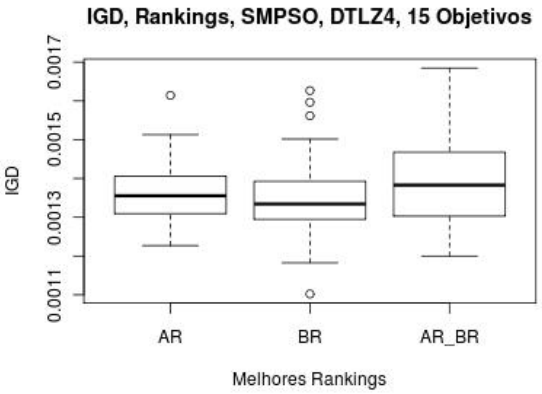



(c) *Spacing*

Figura 7.24: Média das medidas GD e IGD para o problema DTLZ4, para os métodos de ranking.




(a) GD




(b) IGD

Figura 7.25: *Boxplot* com os melhores rankings para o GD e IGD. Problema DTLZ2, 15 objetivos.

AR_BR (ver boxplot na Figura 7.25(b)). Observando a Figura 7.24(b), estes métodos apresentam valores semelhantes de IGD para quase todos os números de objetivos, ou seja, conseguiram controlar a diversificação das soluções mesmo trabalhando com muitos objetivos. É interessante notar, que o método BR_MR obteve os piores resultados, sendo pior que a execução dos métodos em separado e até pior que a dominância de Pareto para muitos objetivos.

Para os *spacing* os resultados são semelhantes aos demais indicadores: AR, BR e AR_BR com os melhores valores para muitos objetivos. Na Figura 7.24, pode-se observar que as curvas que representam esses algoritmos estão mais próximas de zero e que quando o número de objetivos cresce, os valores de *spacing* não pioram muito. Para os demais métodos é possível perceber que há uma maior deterioração da diversificação quando o número de objetivos cresce, em especial para a dominância de Pareto.

Em resumo, a combinação de rankings foi a melhor técnica dentre todas as analisadas, obtendo uma boa convergência em direção à fronteira de Pareto real e uma boa diversificação das soluções geradas, tanto em torno da fronteira real quanto em relação a distribuição das soluções. O método BR não conseguiu bons resultados para o problema DTLZ2 e apresentou problemas para convergir para a fronteira real quando o número de objetivos cresceu, porém obteve bons resultados para o DTLZ4 e mostrou que gera um conjunto de aproximação com boa diversificação.

### 7.2.3 Ranking-SMPSO X CDAS-MOPSO

O último conjunto de experimentos relacionado às novas relações de preferência confronta os resultados do algoritmo que utiliza métodos de ranking, Ranking-SMPSO, com o algoritmo baseado na técnica CDAS, CDAS-MOPSO. Nesse experimento, será utilizado o método de ranking *Average Ranking* (AR) [4] aplicado ao algoritmo Ranking-SMPSO (Algoritmo 5), chamado de AR-SMPSO. O outro algoritmo é o CDAS-SMPSO, com as mesmas configurações apresentadas na Tabela 7.4. Além disso, o SMPSO também é utilizado nessa comparação. Esse conjunto de experimentos foi conduzido independente dos experimentos anteriores. Alguns parâmetros, como por exemplo o número de avaliações,

Tabela 7.9: Valores de GD para as melhores configurações do CDAS-MOPSO, SMPSO ($S_i = 0,5$) e o AR-SMPSO, para cada objetivo, DTLZ2 e DTLZ4.

| Prob | Obj | 0,25 | 0,3 | 0,35 | 0,4 | 0,45 | SMPSO ($S_i = 0,5$) | AR |
|---|---|---|---|---|---|---|---|---|
| DTLZ2 | 3 | * | * | 3,17E-03 | 2,45E-03 | 3,64E-03 | 1,59E-02 | 1,52E-02 |
| | 5 | * | 1,23E-02 | 9,85E-03 | 1,37E-02 | * | 6,42E-02 | 4,47E-02 |
| | 10 | 1,94E-02 | 2,48E-02 | 3,73E-02 | * | * | 1,32E-01 | 6,22E-02 |
| | 15 | 2,24E-02 | 2,67E-02 | 5,04E-02 | * | * | 1,60E-01 | 8,68E-02 |
| | 20 | 1,85E-02 | 2,82E-02 | 5,32E-02 | * | * | 1,75E-01 | 1,11E-01 |
| DTLZ4 | 3 | 7,74E-06 | 3,51E-02 | 4,74E-02 | 2,87E-02 | 1,28E-02 | 2,58E-02 | 9,04E-02 |
| | 5 | 5,12E-06 | 4,81E-02 | 5,14E-02 | 3,67E-02 | 2,82E-02 | 3,51E-02 | 8,89E-02 |
| | 10 | 7,06E-06 | 4,78E-02 | 5,39E-02 | 4,40E-02 | 3,76E-02 | 3,95E-02 | 6,17E-02 |
| | 15 | 1,11E-05 | 4,84E-02 | 5,33E-02 | 4,51E-02 | 3,62E-02 | 5,15E-02 | 6,25E-02 |
| | 20 | 1,02E-05 | 4,07E-02 | 4,65E-02 | 4,13E-02 | 3,85E-02 | 6,32E-02 | 6,38E-02 |

são diferentes, logo os valores dos indicadores apresentados anteriormente não são os mesmos. O objetivo dessa análise é comparar CDAS com métodos de ranking em algoritmos MOPSO. A influência das técnicas utilizadas em algoritmos MOPSO foi discutida nas seções anteriores.

Esta análise também será dividida em duas etapas, a primeira compara os resultados dos indicadores de qualidade GD, IGD, *spacing* e o tempo de execução. A segunda etapa busca analisar a distância das soluções em relação ao joelho do problema, através da distância de Tchebycheff. Nas duas etapas são considerados dois problemas: DTLZ2 para observar a convergência dos algoritmos e o DTLZ4 para observar a diversidade na busca. Foram utilizados 3, 5, 10, 15 e 20 números de funções objetivo. Os algoritmos foram executados com 200 partículas na população, 200 soluções no repositório (mesmo tamanho da população) e o critério de parada foi o número de gerações, definido com 100 iterações. Cada algoritmo foi executado 30 vezes. Os resultados discutidos nesta seção foram publicados no trabalho apresentado em [25].

As Tabelas 7.9, 7.10, 7.11 e 7.12 apresentam os resultados das comparações entre o AR-SMPSO e o CDAS-SMPSO. Cada tabela apresenta o valor médio dos indicadores, para cada execução. Somente algumas configurações do CDAS-MOPSO foram utilizadas. Essas melhores configurações foram obtidas de um trabalho prévio do autor apresentado em [14] e [16]. As células marcadas com $*$ representam configurações que não participam da comparação.

Para o GD (ver Tabela 7.9), o CDAS-MOPSO superou o AR-SMPSO para todos os objetivos, em ambos os problemas. Observa-se que quando o número de objetivos cresceu a diferença entre as técnicas diminuiu. Além disso, o AR-SMPSO obteve um resultado

Tabela 7.10: Valores de IGD para as melhores configurações do CDAS-MOPSO, SMPSO ($S_i = 0,5$) e o AR-SMPSO, para cada objetivo, DTLZ2 e DTLZ4.

| Prob | Obj | 0,25 | 0,3 | 0,35 | 0,4 | 0,45 | SMPSO ($S_i = 0,5$) | AR |
|---|---|---|---|---|---|---|---|---|
| DTLZ2 | 3 | * | * | * | * | 7,77E-04 | 9,46E-04 | 1,84E-03 |
| | 5 | * | * | 2,85E-03 | 2,32E-03 | 2,47E-03 | 4,93E-03 | 4,56E-03 |
| | 10 | 4,46E-01 | * | * | * | * | 1,10E-02 | 5,54E-03 |
| | 15 | * | 3,24E-03 | 3,43E-03 | 4,12E-03 | * | 1,45E-02 | 5,77E-03 |
| | 20 | 4,12E-03 | 3,19E-03 | 3,37E-03 | 4,42E-03 | * | 1,59E-02 | 6,20E-03 |
| DTLZ4 | 3 | 1,08E-01 | 6,81E-002 | 4,80E-02 | 3,80E-02 | 3,02E-02 | 3,11E-02 | 1,62E-01 |
| | 5 | 8,09E-02 | 5,51E-02 | 4,05E-02 | 3,71E-02 | 3,54E-02 | 3,50E-02 | 1,20E-01 |
| | 10 | 7,70E-02 | 5,69E-02 | 5,05E-02 | 4,78E-02 | 4,61E-02 | 4,63E-02 | 7,31E-02 |
| | 15 | 1,15E-01 | 7,79E-02 | 7,06E-02 | 6,76E-02 | 6,46E-02 | 6,70E-02 | 1,06E-01 |
| | 20 | 8,92E-02 | 5,89E-02 | 5,50E-02 | 5.35E-02 | 5,19E-02 | 6,04E-02 | 8,44E-02 |

Tabela 7.11: Valores de *spacing* para as melhores configurações do CDAS-MOPSO, SMPSO ($S_i = 0,5$) e o AR-SMPSO, para cada objetivo, DTLZ2 e DTLZ4.

| Prob | Obj | 0,25 | 0,3 | 0,35 | 0,4 | 0,45 | SMPSO ($S_i = 0,5$) | AR |
|---|---|---|---|---|---|---|---|---|
| DTLZ2 | 3 | * | * | * | * | * | 8,08E-01 | 8,79E-01 |
| | 5 | 4,46E-01 | * | * | * | * | 1,53E+00 | 1,27E+00 |
| | 10 | 4,96E-01 | * | * | * | * | 2,39E+00 | 1,36E+00 |
| | 15 | 5,53E-01 | 7,57E-01 | * | * | * | 2,54E+00 | 1,82E+00 |
| | 20 | 5,52E-01 | 7,47E-01 | 9,84E-01 | * | * | 2,56E+00 | 2,28E+00 |
| DTLZ4 | 3 | 0,00E+00 | 4,82E-01 | 7,19E-01 | 7,50E-01 | 7,06E-01 | 8,06E-01 | 5,39E-01 |
| | 5 | 0,00E+00 | 6,76E-01 | 1,04E+00 | 1,05E+00 | 1,05E+00 | 1,15E+00 | 5,22E-01 |
| | 10 | 0,00E+00 | 9,42E-01 | 1,30E+00 | 1,30E+00 | 1,33E+00 | 1,46E+00 | 5,38E-01 |
| | 15 | 0,00E+00 | 1,04E+00 | 1,41E+00 | 1,39E+00 | 1,40E+00 | 1,50E+00 | 6,41E-01 |
| | 20 | 9,48E-03 | 1,08E+00 | 1.45E+00 | 1,45E+00 | 1,40E+00 | 1,50E+00 | 7,29E-01 |

Tabela 7.12: Tempo de execução (segundos) para todas as configurações do CDAS-MOPSO, SMPSO ($S_i = 0,5$) e o AR-SMPSO, para cada número de objetivo, DTLZ2 e DTLZ4.

| Prob | Obj | 0,25 | 0,3 | 0,35 | 0,4 | 0,45 | SMPSO ($S_i = 0,5$) | AR |
|---|---|---|---|---|---|---|---|---|
| DTLZ2 | 3 | 0,52 | 1,44 | 6,42 | 34,32 | 39,05 | 32,88 | 4,72 |
| | 5 | 1,60 | 33,65 | 66,56 | 475,11 | 160,80 | 68,85 | 8,50 |
| | 10 | 7,37 | 758,33 | 144,74 | 1091,48 | 962,58 | 157,86 | 15,72 |
| | 15 | 12,61 | 626,27 | 446,05 | 1037,35 | 344,03 | 434,92 | 23,48 |
| | 20 | 21,05 | 1078,43 | 602,08 | 979,36 | 865,95 | 1020,49 | 32,77 |
| DTLZ4 | 3 | 0,92 | 2,49 | 3,01 | 3,90 | 4,97 | 9,58 | 3,32 |
| | 5 | 2,16 | 5,20 | 6,25 | 7,85 | 10,10 | 17,70 | 8,76 |
| | 10 | 3,56 | 5,91 | 7,36 | 8,00 | 11,53 | 35,22 | 19,95 |
| | 15 | 10,73 | 16,18 | 16,89 | 21,09 | 25,25 | 124,62 | 30,22 |
| | 20 | 19,17 | 25,24 | 27,18 | 31,25 | 36,27 | 166,01 | 40,28 |

equivalente ao o SMPSO ($S_i = 0,5$), especialmente para muitas funções objetivo. Em resumo, a técnica CDAS obteve uma melhor convergência.

A Tabela 7.10 apresenta um resumo com os valores de IGD. Novamente, os melhores resultados foram obtidos pelo CDAS. Para o problema DTLZ2, o CDAS-SMPSO obteve os melhores valores de IGD para todos os objetivos, no entanto o AR-SMPSO obteve um resultado bem próximo. Para o DTLZ4, os algoritmos têm resultados semelhantes, porém o SMPSO apresentou o melhor resultado. Tanto AR-SMPSO e o CDAS-SMPSO possuem resultados equivalentes para a maioria dos números funções objetivo. Analisando os dois problemas, conclui-se que o CDAS-SMPSO gera um conjunto de aproximação mais bem distribuído que o AR-SMPSO, porém ambos enfrentam dificuldades de diversificação da busca para o DTLZ4.

Para o *spacing* (ver Tabela 7.11) os resultados são semelhantes aos demais indicadores. O CDAS-SMPSO obteve o melhor resultado, para ambos os problemas. Isto ocorreu devido ao menor número de soluções gerado por este algoritmo. Como discutido na seção anterior, quanto menor o tamanho do conjunto de aproximação, menor é o *spacing*. O AR-SMPSO gerou um número constante de soluções, do mesmo tamanho do repositório. O AR-SMPSO obteve um resultado melhor que a relação da dominância de Pareto ($S_i = 0,5$).

A Tabela 7.12 apresenta o tempo médio de execução, em segundos, para todas as configurações do CDAS-SMPSO, o SMPSO ($S_i = 0,5$) e o AR-SMPSO. Para o DTLZ2, o CDAS-SMPSO obteve um melhor tempo de execução, mas somente para a configuração $S_i = 0,25$. Esta configuração obteve um baixo tempo de execução devido ao pequeno conjunto de aproximação gerado, Tabela 7.6. O AR-SMPSO executou mais rápido que as demais configurações, incluindo ao SMPSO. Para o DTLZ4, os resultados são diferentes. O CDAS-SMPSO executou mais rápido que AR-SMPSO, especialmente para um $S_i$ é pequeno.

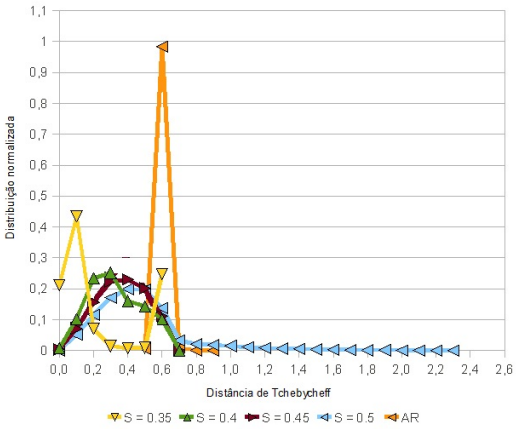
Em resumo, a técnica CDAS apresentou melhores resultados que o AR, para problemas com muitos objetivos. Ele superou os resultados do AR para todos os indicadores analisados. O CDAS-SMPSO obteve melhor convergência e diversidade do que o AR-SMPSO.

O AR-SMPSO obteve melhores resultados que o SMPSO com a relação da dominância de Pareto, especialmente quando o número de objetivos cresce. Além disso, o AR-SMPSO não necessita da configuração de parâmetros extras e executa rápido independentemente do problema e da configuração dos parâmetros.

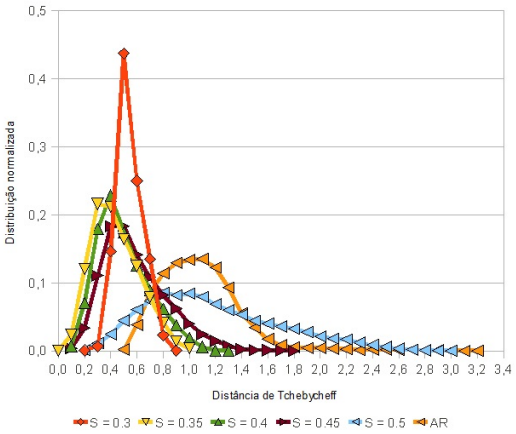
A segunda etapa deste conjunto de experimentos visa comparar as distâncias das soluções geradas por cada técnica em relação ao joelho da fronteira de Pareto [24], através da distribuição da distância de Tchebycheff. As Figuras 7.26 e 7.27 apresentam a distribuição da distância de Tchebycheff para as melhores configurações do CDAS-SMPSO (melhor GD e IGD), AR-SMPSO e o SMPSO com a relação da dominância de Pareto($S_i = 0.5$), para os problemas DTLZ2 e DTLZ4 com 3, 5, 10, 15 e 20 números de objetivos. Os gráficos com as distribuições apresentam as concentrações das soluções normalizadas (variam entre 0 e 1).

Para o problema DTLZ2, o CDAS-MOPSO produziu as melhores distribuições. Para 3 e 5 objetivos, todas as configurações do CDAS-MOPSO geraram suas distribuições concentradas em valores baixos da distância de Tchebycheff, isto é, próximas ao joelho. Quando o número de objetivos cresceu, somente as configurações $S_i = 0,25$ e $0,3$ ainda concentraram suas distribuições próximas ao joelho. As demais configurações, produziram uma distribuição mais diversificada. Como discutido na seção anterior, o SMPSO ($S_i = 0,5$) produziu uma distribuição similar para todos os objetivos. Essas soluções estão distribuídas em diferentes distâncias. Além disso, estas distribuições estão longe do joelho. Os resultados do AR-SMPSO são semelhantes aos obtidos pelo SMPSO com a relação da dominância de Pareto: soluções distribuídas em diferentes distâncias e longe do joelho. Como discutido nos Capítulos 2 e 3, o método AR privilegia a geração de soluções nos extremos da busca, longe da solução ideal. Para o DTLZ4, novamente a relação da dominância de Pareto e o AR-SMOPSO geraram distribuições diversificadas. No entanto, o AR-SMPSO concentrou suas soluções mais longe do joelho. Os melhores resultados foram obtidos pelo CDAS-MOPSO. Para esse problema, quase todas as soluções geradas se concentraram numa mesma distância, próximo ao joelho.

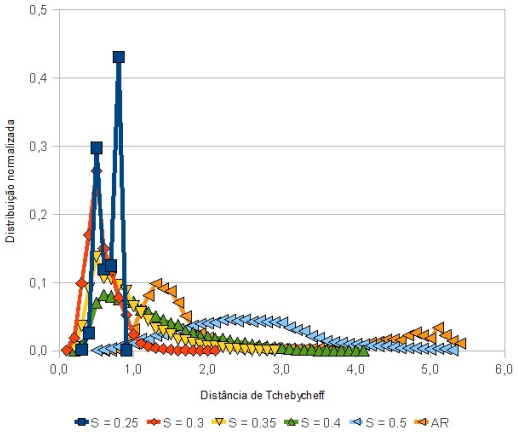
Em resumo, o CDAS-SMPSO foi a melhor técnica, agora considerando a distribuição



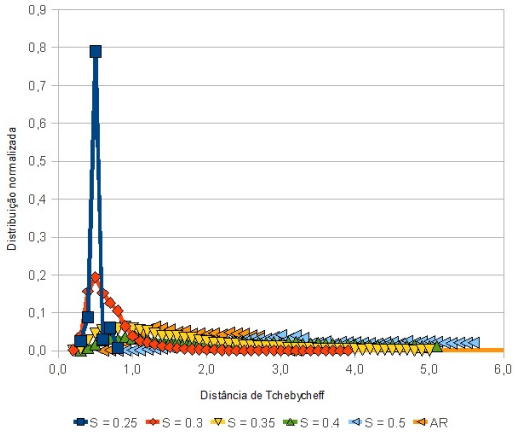

(a) 3 objetivos



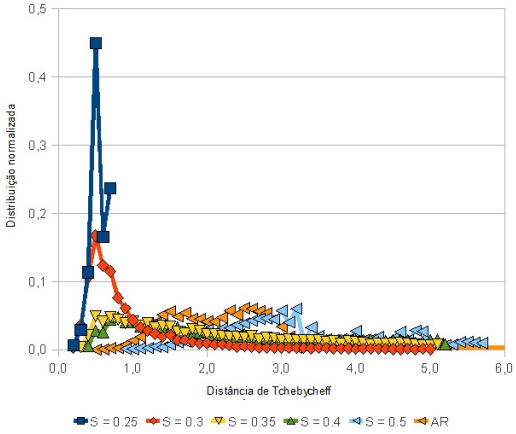

(b) 5 objetivos




(c) 10 objetivos

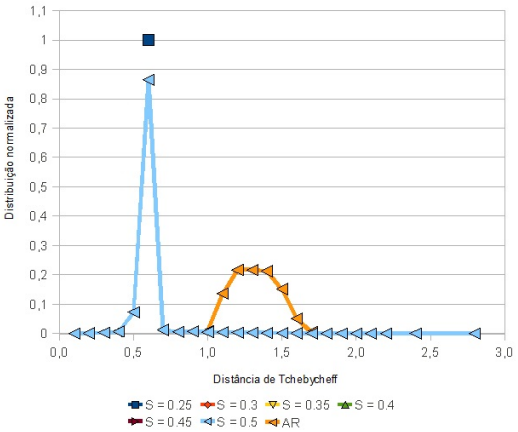



(d) 15 objetivos

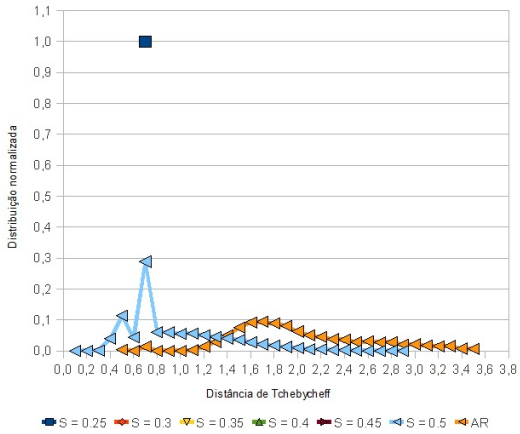



(e) 20 objetivos

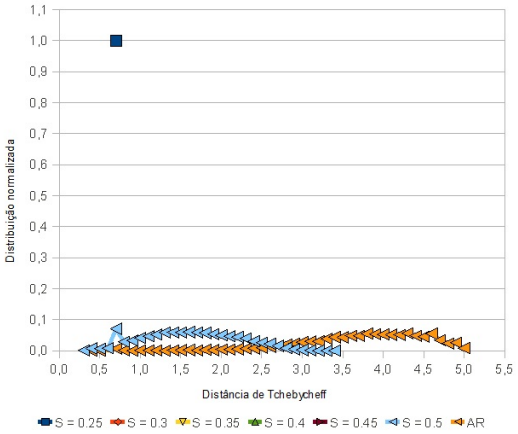
Figura 7.26: Distribuição da distância de Tchebycheff para o CDAS-SMPSO (melhor GD ou IGD), SMPSO ($S_i = 0,5$) e AR-SMPSO para o problema DTLZ2.



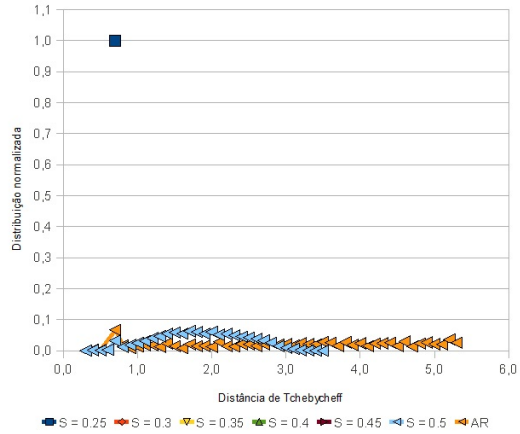

(a) 3 objetivos



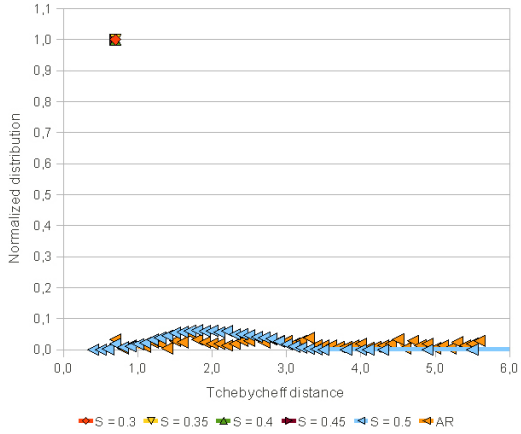

(b) 5 objetivos




(c) 10 objetivos




(d) 15 objetivos




(e) 20 objetivos

Figura 7.27: Distribuição da distância de Tchebycheff para o CDAS-SMPSO (melhor GD ou IGD), SMPSO ($S_i = 0,5$) e AR-SMPSO para o problema DTLZ4.

da distância em relação ao joelho. Ele gerou quase todas suas soluções próximas ao joelho para ambos os problemas e todos os números de objetivos. Examinando os indicadores de qualidade e distribuição da distância de Tchebycheff, o CDAS-MOPSO obteve resultados muito bons: boa convergência, diversidade e soluções próximas ao joelho. O AR-SMPSO produziu uma distribuição mais diversificada.

### 7.2.4 Avaliação dos métodos de arquivamento aplicados ao MOPSO

Esta seção avalia os resultados dos métodos de arquivamento propostos. Esse conjunto de experimentos tem como objetivo comparar os métodos de arquivamento propostos no Capítulo 5 com diferentes arquivadores da literatura, todos aplicados em um algoritmo MOPSO. Além disso, esse estudo busca identificar quais métodos apresentam os melhores resultados em diferentes problemas com muitos objetivos.

Os arquivadores utilizados são descritos nos Capítulos 3 e 5. Um resumo dos métodos utilizados é apresentado nas Tabelas 7.13 e 7.14. A primeira tabela apresenta somente os métodos propostos nesta tese, enquanto a segunda apresenta os métodos da literatura. O Arquivador Hiperplano é utilizado construindo o hiperplano no início da busca e selecionando o ponto de referência através da escolha de regiões. Todos os métodos de arquivamento são aplicados ao algoritmo base SMPSO, com os parâmetros descritos na Tabela 7.1.

Tabela 7.13: Métodos de arquivamento propostos nesta tese.

| **Método** | **Descrição** |
|---|---|
| Ideal | Arquivador Ideal |
| Dist | Arquivador Distribuído |
| Eucli | Arquivador Distribuído pela Busca usando distância Euclidiana |
| Tcheb | Arquivador Distribuído pela Busca usando distância de Tchebycheff |
| Hyp_Ex | Arquivador Hiperplano usando a escolha do ponto de referência numa região nos extremos do hiperplano |
| Hyp_M | Arquivador Hiperplano usando a escolha do ponto de referência numa região no centro do hiperplano |

Tabela 7.14: Métodos de arquivamento da literatura.

| **Método** | **Descrição** |
|---|---|
| AG | Arquivador *Adaptive Grid* |
| AR | Arquivador *Average Rank* |
| Crowd | Arquivador *Crowding Distance* |
| Dom | Arquivador Dominante |
| MGA | *Multi-level Grid Archiving* |
| Rand | Arquivador aleatório |

Nesta análise empírica foram utilizados os indicadores de qualidade GD, IGD e LD para medir a convergência e a diversidade da busca dos algoritmos. O LD foi utilizado para auxiliar a interpretação da diversidade da busca dos métodos. Os resultados dos arquivadores foram comparados entre si, avaliando como cada algoritmo se comporta quando é submetido a diferentes problemas com diferentes números de funções objetivo. São utilizados os problemas DTLZ2, DTLZ4, DTLZ6 e DTLZ7. Os números de funções objetivo utilizados foram: 3, 5, 10, 15 e 20. Para os problemas DTLZ2, DTLZ6 e DTLZ7 foi definido um total 200 partículas na população, 200 soluções no arquivo externo e um total de 50000 avaliações de *fitness*. Para o DTLZ4 é gerado um número menor de soluções, porém para a avaliação dos métodos de arquivamento o arquivo externo precisa ser cheio para que seja executada a função $filtro$ de cada método. Assim, os parâmetros para execução foram aumentados para que fossem geradas soluções suficientes para preencher o arquivo, além disso o tamanho do arquivo externo foi diminuído. O DTLZ4 foi executado com 200 partículas na população, 100 soluções no arquivo externo e um total de 100000 avaliações de fitness. Os algoritmos foram executados um total de 20 execuções

Os resultados das melhores configurações obtidas pelo pós-teste do teste de Friedman para todos os problemas são apresentados na Tabela 7.15. Os gráficos contendo a variação da média dos indicadores de acordo com o número de objetivos são apresentados nas Figuras de 7.28 a 7.32.

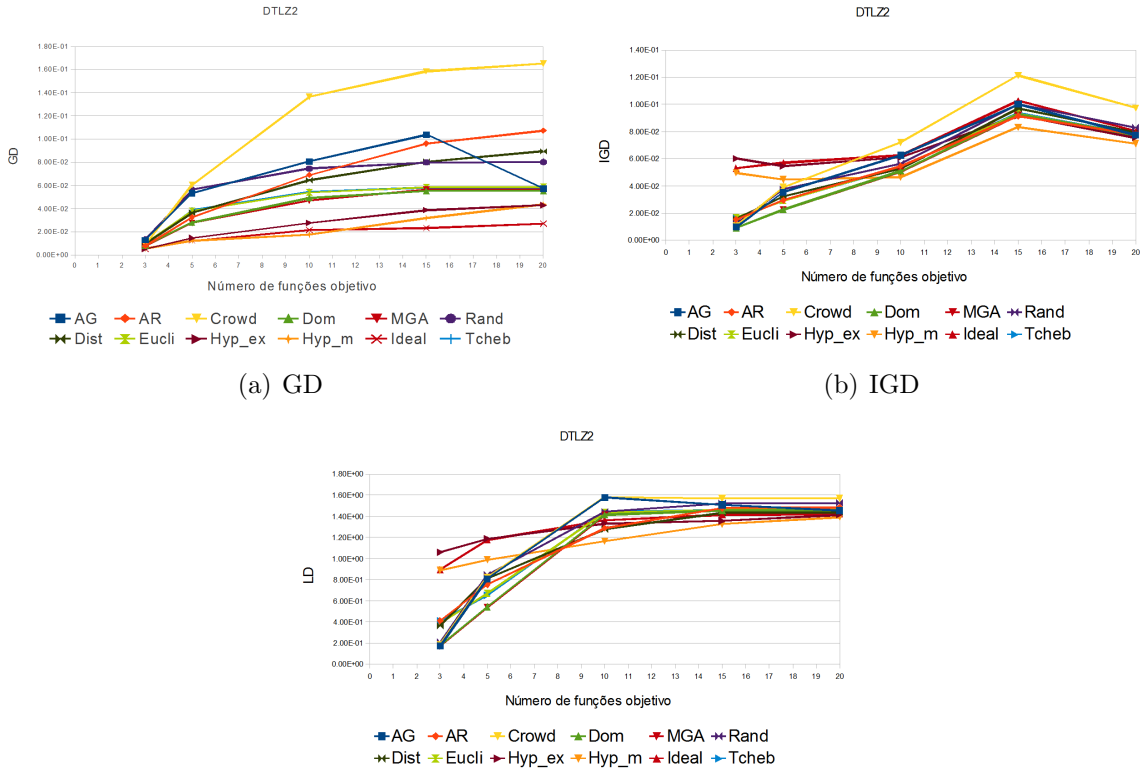
Tabela 7.15: Melhores métodos de arquivamento para todos os problemas e funções objetivos, de acordo com o pós-teste do teste de Friedman.

| **Prob** | **Obj** | **Melhores métodos de arquivamento** | | |
|---|---|---|---|---|
| | | **GD** | **IGD** | **LD** |
| **DTLZ2** | **3** | Hyp_Ex, Hyp_M e Ideal | AG, Crowd, Dom, MGA e Rand | AG, Crowd, Dom, MGA e Rand |
| | **5** | Dom, MGA, Hyp_Ex, Hyp_M e Ideal | AR, Dom, MGA, Eucli e Tcheb | Dom, MGA, Eucli e Tcheb |
| | **10** | MGA, Hyp_Ex, Hyp_M e Ideal | Dom, MGA, Dist e Hyp_M | Ar, Dist, Hyp_Ex e Hyp_M |
| | **15** | Dom, Hyp_Ex, Hyp_M e Ideal | AR, MGA, Hyp_Ex e Hyp_M | Dist, Hyp_Ex, Hyp_M e Ideal |
| | **20** | Hyp_Ex, Hyp_M e Ideal | Hyp_Ex, Hyp_M | Dist, Hyp_Ex, Hyp_M e Ideal |
| **DTLZ4** | **3** | Todos | Todos | Todos |
| | **5** | Dist, Hyp_Ex, Hyp_M e Ideal | AG, AR, Crowd, Dom, MGA, Rand, Dist e Eucli | Todos (exceto Hyp_Ex e Hyp_M) |
| | **10** | Hyp_Ex, Hyp_M e Ideal | AR, Crowd, Dom, MGA, Rand, Eucli, Hyp_M e Tcheb | AG, Crowd, Dom, MGA, Rand, Eucli e Tcheb |
| | **15** | Dist, Hyp_Ex, Hyp_M e Ideal | AR, MGA, Hyp_Ex, Hyp_M, Ideal e Tcheb | Eucli, Hyp_Ex, Hyp_M, Ideal e Tcheb |
| | **20** | Hyp_Ex, Hyp_M e Ideal | Hyp_Ex, Hyp_M e Ideal | Hyp_Ex, Hyp_M e Ideal |
| **DTLZ6** | **3** | AR, Crowd, Dom, MGA, Dist, Hyp_Ex, Hyp_M e Ideal | AG, Crowd, MGA e Rand | AG, Crowd, Dom, MGA e Rand |
| | **5** | Hyp_Ex, Hyp_M e Ideal | AR, Crowd, Dom, MGA, Rand, Hyp_M e Ideal | Crowd, Dom, MGA, Rand, Hyp_M e Ideal |
| | **10** | Hyp_Ex, Hyp_M e Ideal | AG, Dom, MGA, Dist, Hyp_Ex e Ideal | AG, Dom, MGA, Dist e Ideal |
| | **15** | Hyp_Ex, Hyp_M e Ideal | Dom, MGA e Dist | Dom, MGA e Dist |
| | **20** | Hyp_Ex, Hyp_M e Ideal | Dist e Hyp_Ex | Dist e Hyp_Ex |
| **DTLZ7** | **3** | AR, Eucli e Tcheb | AG, Crowd, Dom, MGA e Rand | AG, Crowd, Dom, MGA e Rand |
| | **5** | Hyp_Ex, Hyp_M e Ideal | AG, Crowd, Dom e Rand | AG, Crowd, Rand e Dist |
| | **10** | Crowd, Rand, Hyp_Ex, Hyp_M e Ideal | Crowd, Rand e Dist | Crowd, Rand e Dist |
| | **15** | Crowd, Hyp_Ex, Hyp_M e Ideal | Crowd, Rand e Dist | Crowd, Rand e Dist |
| | **20** | Crowd, Hyp_Ex, Hyp_M e Ideal | Crowd, Rand e Dist | Crowd, Rand e Dist |

O primeiro problema analisado é o DTLZ2. Para a medida de convergência da busca, GD, os métodos Hyp_Ex, Hyp_M e Ideal obtiveram os melhores resultados para todos os números de objetivo. Esses métodos têm a característica de definir um ponto e guiar a seleção das soluções do arquivo em direção desse ponto obtendo maior convergência. Para o DTLZ2 essa estratégia se mostrou eficiente tanto para poucos e muitos objetivos. Além desses métodos os arquivadores Dom e MGA também apresentaram bons resultados para alguns números de funções objetivo. Os indicadores IGD e LD apresentaram valores semelhantes, porém alguns métodos se destacaram num indicador e não em outro. Vale ressaltar, que o LD não sofre influência do fato da $PF_{aprox}$ estar muito próxima da fronteira de Pareto. Para essas medidas os métodos MGA e Dom apresentam bons resultados, porém perdem desempenho quando o número de objetivos cresce. Esses dois métodos apresentaram bons valores de IGD e LD para 3, 5 e 10 funções objetivo. Para os maiores conjunto de funções, os arquivadores Dist, Hyp_Ex, Hyp_M apresentaram os melhores resultados.

Nos gráficos apresentados na Figura 7.28 é possível observar o comportamento derivado dos resultados do teste de Friedman. O Arquivador Hiperplano apresenta a menor deterioração, tanto em termos de convergência quanto em diversidade. Para as medidas de diversidade, esse arquivador não apresenta um valor tão bom quanto os demais para poucos objetivos, porém consegue manter uma mesma faixa desse valor para todos os números de objetivos, enquanto os demais apresentam uma deterioração.

Esse baixo valor inicial de IGD e LD do Arquivador Hiperplano para poucos objetivos pode ser observado na Figura 7.29. Essa figura mostra a $PF_{aprox}$ gerada pelo Hyp_M (pontos em amarelo) e pelo MGA (pontos em verde). Também é traçada a fronteira de Pareto do problema. Percebe-se que o conjunto de pontos do Hyp_M ficou concentrado no centro da fronteira, quanto o MGA apresenta uma $PF_{aprox}$ distribuída em diferentes regiões da fronteira. Isso mostra que o Hyp_M consegue manter sua busca perto da região central. Além disso, os bons valores de GD e IGD desse método mostram que limitar a busca para um região menor do espaço de objetivos é eficiente para controlar a deterioração quando o número de objetivos cresce.




(a) GD

(b) IGD

(c) LD

Figura 7.28: Valores médios de GD, IGD e LD dos métodos de arquivamento, DTLZ2.

Para o problema DTLZ4, um fato a se destacar é que para 3 funções objetivos, os algoritmos não geraram um número suficiente de soluções para preencher o arquivo. Assim, não foi executada a função *filtro* e todos os algoritmos tiveram um resultado equivalente. Logo, para esse problema esse número de objetivos será descartado. Novamente, os métodos Hyp_Ex, Hyp_M e Ideal se destacaram. Esses métodos obtiveram os melhores valores de GD. Porém, somente a análise do GD pode ser enganosa para o DTLZ4, pois isso pode indicar que o algoritmo ficou preso na região mais densa do espaço de objetivos. Porém, observando as medidas IGD e LD, esses algoritmos se destacaram para um número grande de funções objetivo e apresentaram os melhores resultados para o DTLZ4. Para um menor número de objetivos, vários métodos apresentam um resultado equivalente, porém o MGA e o **Eucli** se destacaram. Novamente o MGA apresentou os melhores valores de IGD, porém não apresentou os melhores valores de LD. Isso indica que esse método atingiu menos regiões diferentes da fronteira que os métodos que conseguiram melhor LD, porém sua $PF_{aprox}$ é gerada mais próxima da fronteira. Nos gráficos apresen-

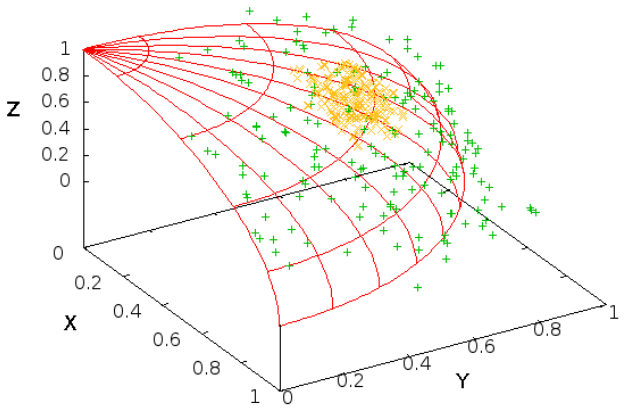
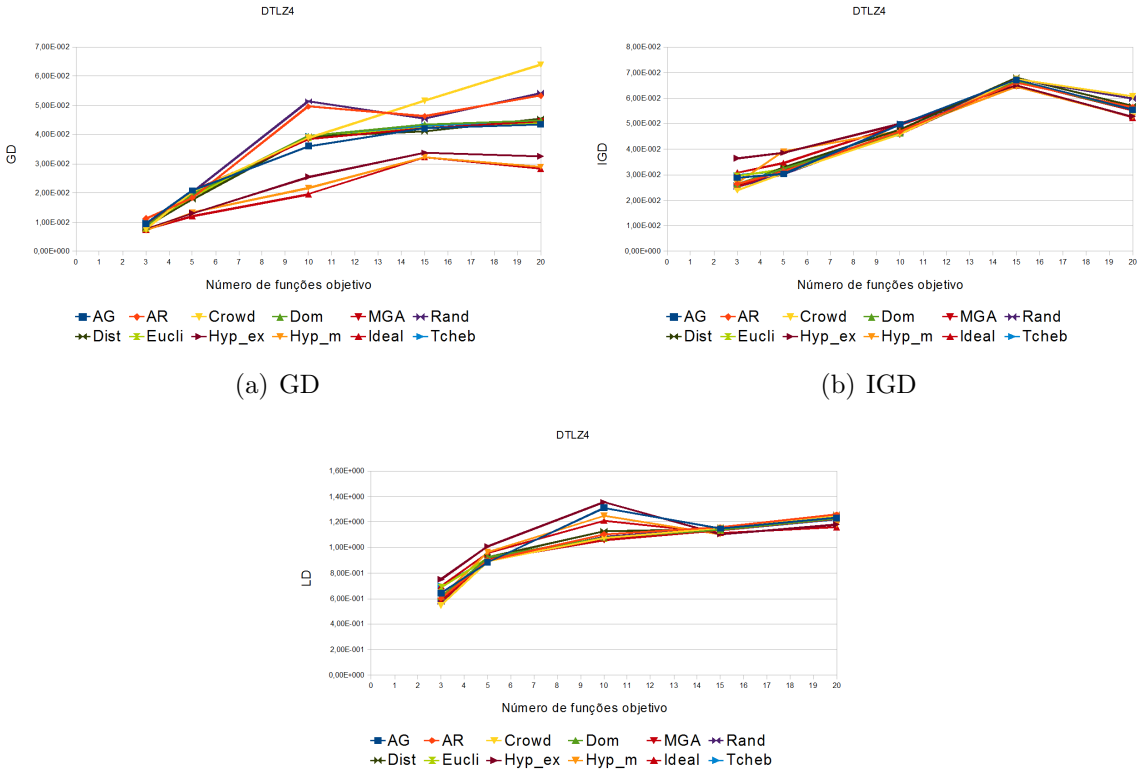



Figura 7.29: $PF_{aprox}$ do MGA e do Hyp_M para o problema DTLZ2 com 3 funções objetivo.

tados na Figura 7.30 é possível perceber os bons valores de GD para os métodos Hyp_Ex, Hyp_M e Ideal, em especial para muitos objetivos. Para as medidas de diversidade, todos os algoritmos apresentaram resultados semelhantes. Isso mostra que todos enfrentaram dificuldades com as regiões mais densas do espaço de objetivos e tiveram dificuldade em diversificar a busca.

O problema DTLZ6 possui vários múltiplos locais, o que dificulta a convergência dos MOEAs. Para o GD, novamente os métodos Hyp_Ex, Hyp_M e Ideal obtiveram os melhores resultados, especialmente para muitos objetivos. A estratégia utilizada por esses métodos se mostrou eficiente para evitar os ótimos locais. Porém, em geral, esses métodos não obtiveram bons valores para as medidas de diversidade. Para o IGD e LD, se destacaram os métodos MGA, Dom e Dist. O Hyp_Ex conseguiu o melhor valor de IGD e LD para 20 funções objetivo. Esse resultado é semelhante aos resultados dos demais problemas, onde o Arquivador Hiperplano também sofre a menor deterioração dos indicadores de qualidade. O método Dist consegue os melhores resultados em termos de IGD e LD. Logo, a abordagem de dividir a seleção das soluções do arquivo através dos pontos extremos se mostrou eficiente no problema DTLZ6. Os gráficos da Figura 7.31 mostram o comportamento dos métodos de arquivamento para o DTLZ6. Através da variação do GD, é possível perceber como os métodos Hyp_Ex, Hyp_M e Ideal controlaram a deteri-




(a) GD

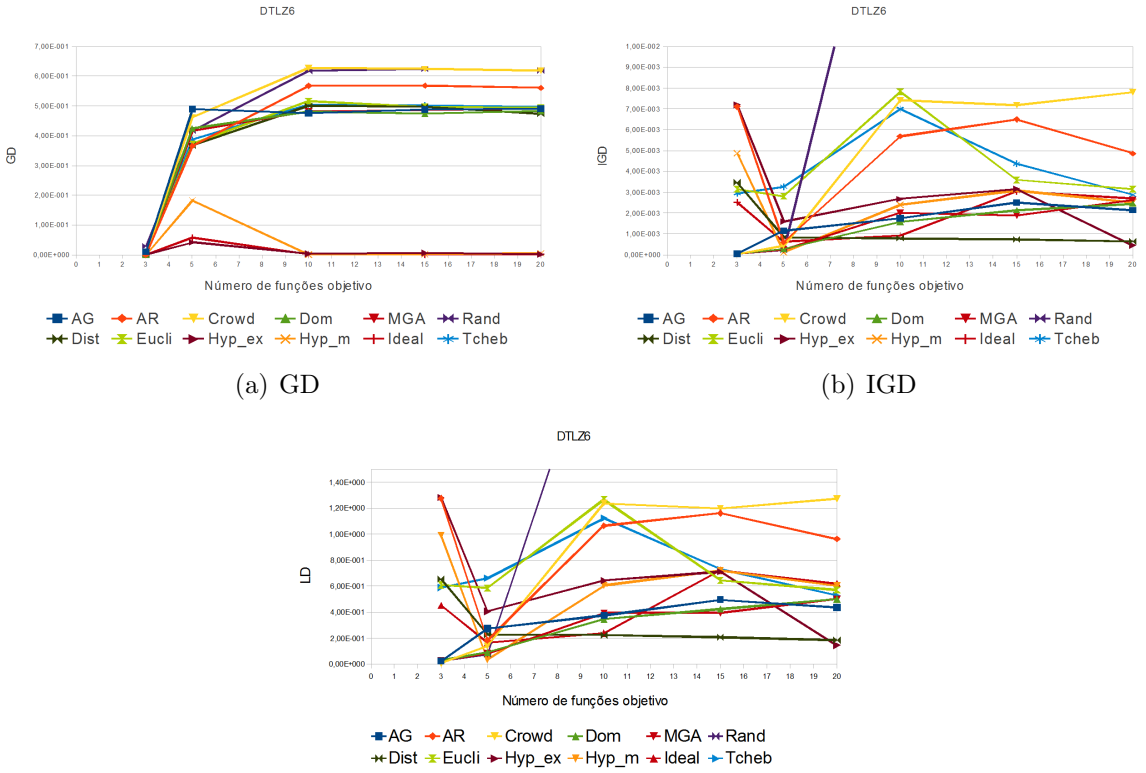
(b) IGD

(c) LD

Figura 7.30: Valores médios de GD, IGD e LD dos métodos de arquivamento, DTLZ4.

oração da busca em termos de convergência em relação aos demais. Para o IGD e LD, o Arquivador Distribuído consegue bons resultados para todos os números de objetivos.

Por fim, o último problema é o DTLZ7. Nesse problema o arquivador Crowd, que é originalmente utilizado pelo SMPSO, se destacou. Esse método apresentou os melhores valores para o GD, IGD e para o LD. Para o GD, além do Crowd os algoritmos Hyp_Ex, Hyp_M e Ideal também apresentaram os melhores valores para a maioria dos números de funções objetivo. Os arquivadores AR, Eucli e Tcheb obtiveram os melhores resultados em termos de convergência para 3 funções objetivo, porém perderam desempenho quando o número de objetivos cresceu. Essas três abordagens têm em comum o fato de privilegiarem os pontos no extremo na busca e essa estratégia se mostrou eficiente para poucos objetivos. Porém, quando o número de objetivos cresce o número de pontos extremos também cresce, o que prejudica o desempenho desses algoritmos. Para as medidas de diversidade, IGD e LD, os arquivadores Rand, Dist e Crowd obtiveram os melhores resultados. Esses métodos conseguiram cobrir um maior número de fronteiras desconexas que os demais. Na



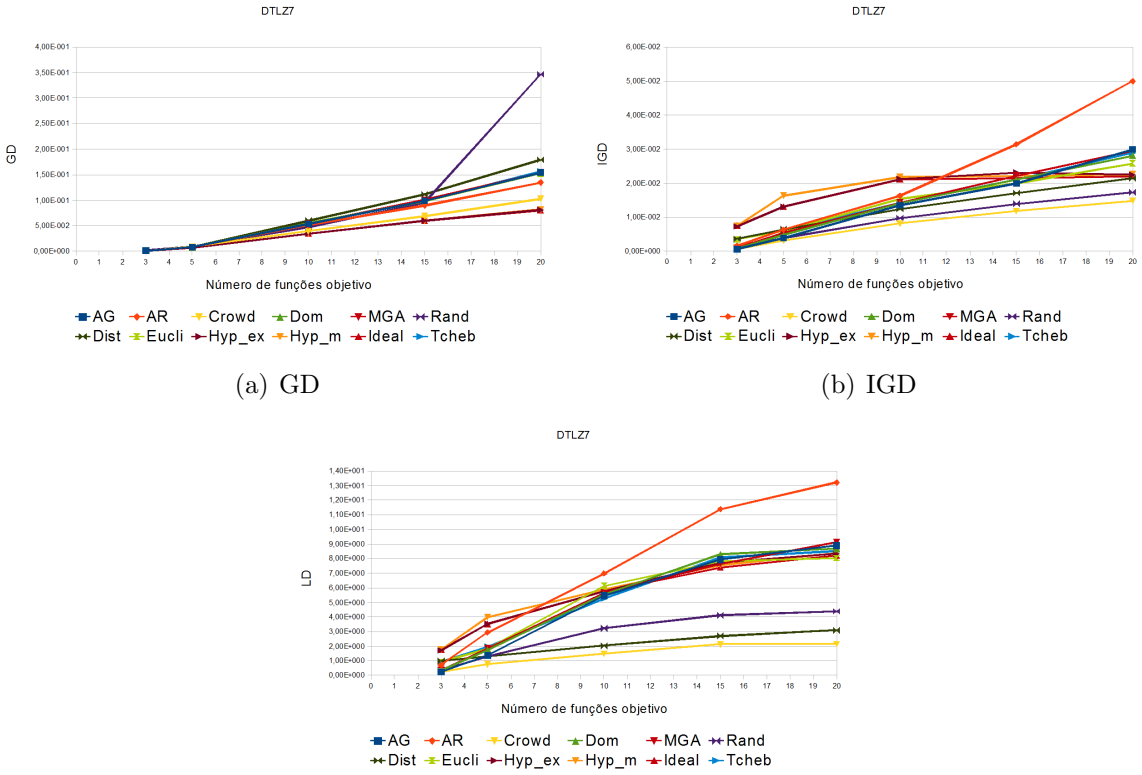

(a) GD (b) IGD

(c) LD

Figura 7.31: Valores médios de GD, IGD e LD dos métodos de arquivamento, DTLZ6.

Figura 7.32 é possível observar o comportamento da busca dos métodos de arquivamento para o DTLZ7. Para o GD, todos os arquivadores perdem desempenho quando o número de objetivos cresce de forma semelhante. Porém o Hyp_Ex, Hyp_M, Ideal e o Crowd mantêm o melhor resultado em termos de convergência em todos os cenários. O resultado é semelhante para o IGD, porém os métodos Crowd, Dist e Rand apresentam uma menor deterioração que os demais. O arquivador Hiperplano apresentou um comportamento semelhante aos demais problemas. Ele não apresentou um bom valor de IGD para poucos objetivos, mas conseguiu manter um valor semelhante para todos os números de funções. Por fim, observando o LD, os métodos Crowd, Rand e Dist se destacaram em relação aos demais. Enquanto os outros métodos apresentaram uma grande deterioração, esses métodos obtiveram bons valores de LD para todas as funções. Isso mostra, que os demais métodos se concentram em poucas fronteiras desconexas, enquanto o Crowd, o Rand e o Dist efetuam uma busca que cobre um maior número dessas fronteiras.

Fazendo uma análise de todos os problemas é possível derivar algumas conclusões.




(a) GD

(b) IGD

(c) LD

Figura 7.32: Valores médios de GD, IGD e LD dos métodos de arquivamento, DTLZ7.

Primeiro, os métodos Hyp_Ex, Hyp_M e Ideal se destacaram em termos de convergência da busca, especialmente para muitos objetivos. A estratégia de escolher um ponto como referência para a seleção das soluções do arquivo se mostrou eficiente e esses algoritmos se aproximaram da fronteira de Pareto de melhor forma que os demais métodos em todos os problemas. Além disso, o Arquivador Hiperplano foi o método que melhor controlou a deterioração da busca. Logo a ideia de se escolher uma região para limitar a busca é eficiente em termos de convergência e há uma cobertura dessa região mesmo para espaço de busca com alto número de funções objetivo. Os métodos MGA e Dom apresentaram os melhores valores para as medidas de diversidade para a maioria dos cenários, porém perdem desempenho quando o número de funções é grande. Dos métodos propostos, o Arquivador Distribuído se destacou nos problemas DTLZ6 e DTLZ7 em termos de diversidade. Para um problema com a fronteira de Pareto desconexa, os métodos Crowd, Rand e Dist são os mais indicados, pois conseguiram controlar melhor a deterioração, enquanto os demais métodos sofreram com o alto número de fronteiras desconexas.

### 7.2.5 Ideal versus CDAS

Este conjunto de experimentos tem como objetivo comparar os resultados de um algoritmo MOPSO com o Arquivador Ideal e algoritmo CDAS-MOPSO. O Arquivador Ideal é proposto nesta tese e juntamente com o Arquivador Hiperplano, apresentou bons resultados quando comparado aos demais métodos de arquivamento da literatura. Assim, esse conjunto de experimentos visa comparar os resultados obtidos por esse método com o algoritmo CDAS-MOPSO que apresenta os melhores resultados dentre as técnicas que utilizam novas relações de preferência. Além disso, também são utilizados os resultados do algoritmo SMPSO.

O algoritmo MOPSO utilizado como base para o uso do Arquivador Ideal segue o mesmo esquema dos experimentos anteriores. É utilizado o algoritmo SMPSO, configurado com os parâmetros da Tabela 7.1. Entretanto, uma nova extensão é adicionada ao algoritmo. O trabalho apresentado em [17] mostra que os métodos de escolha de líder NWSum [72] e Sigma [67] são mais adequados para problemas com muitos objetivos. Nesse contexto dois algoritmos são utilizados: I-MOPSO que utiliza o SMPSO como base, porém utiliza o Arquivador Ideal e o método NWSum para escolha dos líderes e I-Sigma que utiliza o SMPSO como base, porém utiliza o Arquivador Ideal e o método Sigma para escolha dos líderes.

O CDAS-MOPSO foi utilizado com as configurações apresentadas na Tabela 7.4. Todos os algoritmos foram executados um total de 50000 avaliações de fitness e 20 execuções. A população foi definida com 200 partículas e o arquivo externo foi limitado em 200 soluções. Foram utilizados os problemas DTLZ2, DTLZ4 e DTLZ6 com 3, 5, 10, 15 e 20 números de funções objetivo. Nesse conjunto de experimentos são utilizados os indicadores GD e IGD.

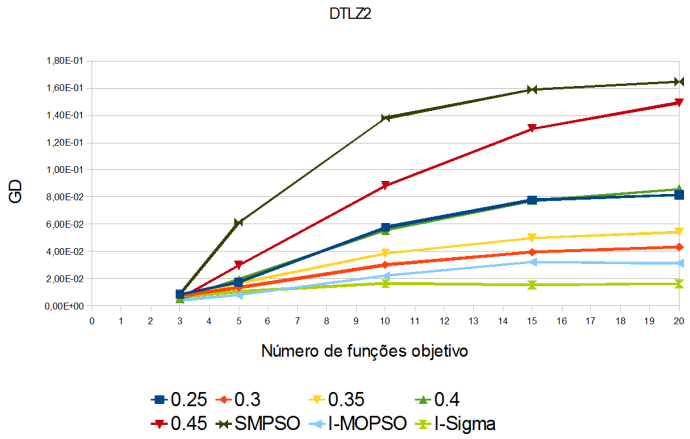
O primeiro problema é o DTLZ2. A Tabela 7.16 apresenta os resultados do pós-teste do teste de Friedman e a Figura 7.33 mostra a média dos indicadores para todos os números de objetivos. Observando a convergência, os algoritmos I-MOPSO e I-Sigma se destacaram. Esses algoritmos obtiveram o melhor valor de GD para todos os números de funções objetivo. O CDAS-MOPSO também conseguiu bons valores de GD para a

configuração com $S_i = 0,3$. Observando a Figura 7.33(a), percebe-se o bom resultado das abordagens baseadas no arquivador Ideal. Elas apresentaram as melhores médias e controlaram a deterioração do GD. O CDAS-MOPSO com $S_i = 0,3$ também consegue controlar a deterioração, mas apresenta um resultado em média pior que os outros dois algoritmos.

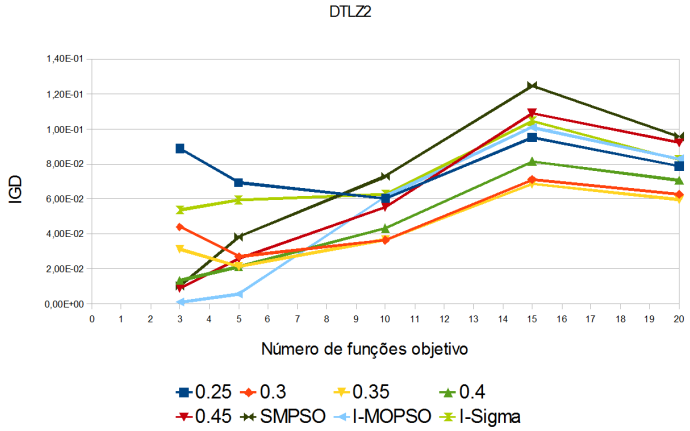
Para o IGD, o CDAS-MOPSO consegue melhores resultados, especialmente para as configurações $S_i = 0,35$ e $0,4$. Essas configurações obtiveram uma melhor diversidade para a maioria dos números de funções objetivo. O I-MOPSO consegue bons resultados apenas para 3 e 5 funções objetivo, porém apresenta uma deterioração em termos de diversidade maior do que o CDAS-MOPSO. Como nos demais experimentos, o SMPSO apresenta uma alta deterioração da busca tanto em termos de convergência quanto em diversidade.

O segundo problema é o DTLZ4. Esse problema possui uma região do espaço de objetivos mais povoada e atrai a busca para essa região. Os algoritmos I-MOPSO e I-Sigma obtiveram os melhores valores de GD para a maioria dos números de objetivos, especialmente para os espaços com maiores dimensões. O SMPSO consegue bons resultados em termos de convergência para 3 funções objetivo, mas seu resultado deteriorou quando o número de objetivos cresceu. Para o CDAS-MOPSO diferentes configurações obtiveram melhores resultados, porém somente para 3 e 5 funções objetivo. Embora uma boa convergência seja desejável para o DTLZ4, a habilidade de evitar as regiões mais povoadas da fronteira de Pareto e assim obter uma boa diversidade é mais importante. Nesse contexto, o CDAS-MOPSO apresentou os melhores resultados, especialmente para problemas com muitos objetivos. As configurações com $S_i = 0,4$ e $0,45$ obtiveram os melhores valores de IGD para todos os números de funções objetivo. O algoritmo I-MOPSO obteve os melhores valores de IGD para 3, 5 e 10 funções objetivo, no entanto quando o número de objetivos cresceu o algoritmo privilegiou convergência sobre diversidade. O SMPSO, novamente, apresentou os piores resultados em termos de diversidade.

Observando as Figuras 7.34(a) e 7.34(b) nota-se os bons valores de GD para o I-MOPSO e I-Sigma. Pode ser observado que a configuração do CDAS-MOPSO com




(a) GD

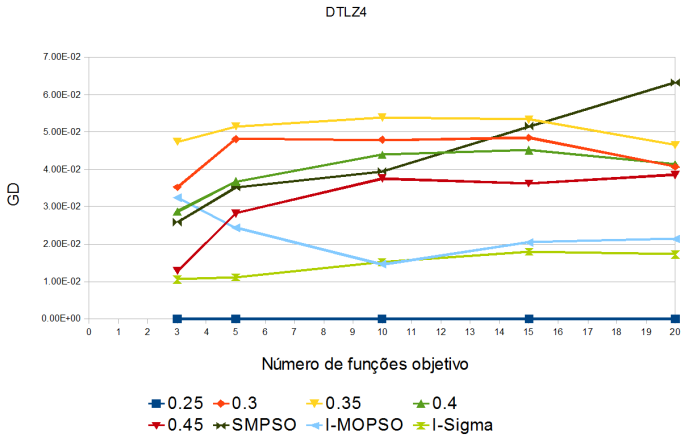



(b) IGD

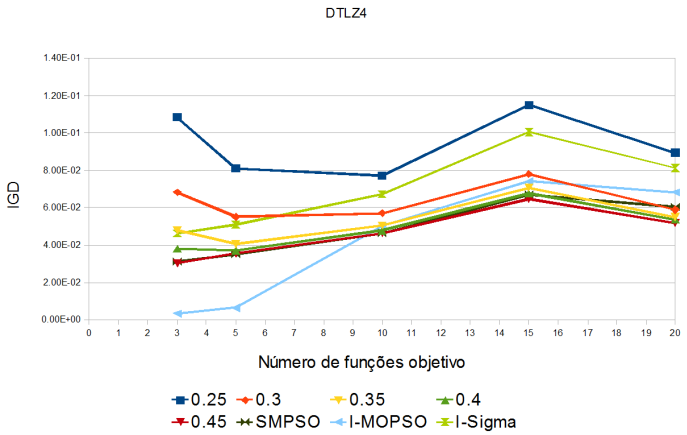
Figura 7.33: I-MOPSO x I-Sigma x CDAS-SMPSO x SMPSO - Valores médios dos indicadores de qualidade para o problema DTLZ2.

$S_i = 0,25$ obteve um valor de GD próximo ao valor zero. Isso ocorreu, pois nesse caso o algoritmo gerou somente uma solução em sua $PF_{aprox}$. Assim, como o algoritmo não efetuou uma busca multiobjetivo, seus resultados foram descartados da análise do teste de Friedman. Por fim, embora o CDAS-MOPSO tenha obtido os resultados significativamente melhores pelo teste de Friedman, observa-se pela Figura 7.34(b) que todos os algoritmos apresentam valores semelhantes de IGD.

O problema DTLZ6 introduz vários ótimos locais com o intuito de dificultar a convergência dos algoritmos. Esses ótimos locais prejudicaram a busca do algoritmo I-MOPSO. Esse algoritmo não obteve o melhor valor de GD para nenhum número de objetivos. No entanto, o I-Sigma obteve os melhores resultados para a maioria dos cenários.




(a) GD



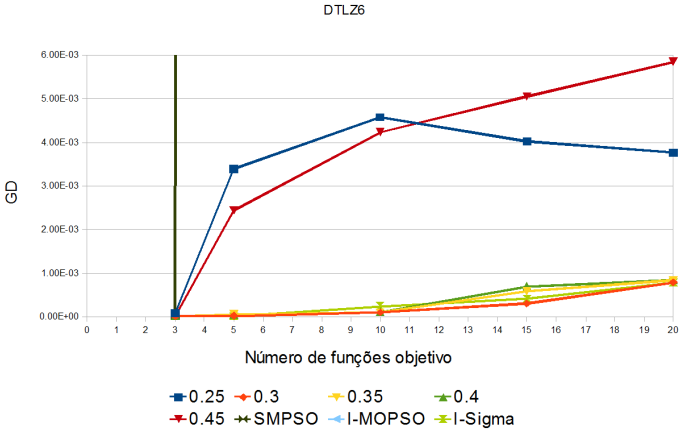

(b) IGD

Figura 7.34: I-MOPSO x I-Sigma x CDAS-SMPSO x SMPSO - Valores médios dos indicadores de qualidade para o problema DTLZ4.

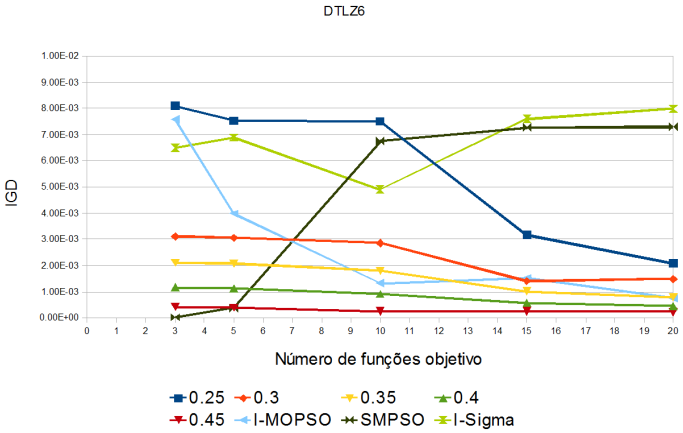
A Figura 7.35(a) mostra os resultados negativos do I-MOPSO e o resultados destacáveis do I-Sigma. O último apresenta uma baixa deterioração, obtendo valores similares de GD para todas as funções objetivo. A diferença entre esses dois algoritmos ressalta a importância da escolha do líder num algoritmo MOPSO. Embora o arquivador Ideal consiga introduzir uma boa convergência na busca, a escolha dos líderes pode introduzir diferentes comportamentos ao algoritmo.

O algoritmo CDAS-MOPSO consegue obter bons valores de GD para diferentes números de objetivos. Observado o IGD (ver Figura 7.35(b)), I-MOPSO apresenta uma diversidade equivalente ao CDAS-MOPSO, especialmente para muitos objetivos, enquanto o I-Sigma não obteve o melhor resultado em nenhum cenário. SMPSO apresentou o comporta-

mento esperado, bons valores para poucos objetivos, porém sofreu deterioração quando esse número cresceu.




(a) GD




(b) IGD

Figura 7.35: I-MOPSO x I-Sigma x CDAS-SMPSO x SMPSO - Valores médios dos indicadores de qualidade para o problema DTLZ6.

Em resumo, os algoritmos baseados em arquivadores produziram notavelmente os melhores resultados em termos de convergência. Além disso, o I-Sigma obteve um resultado muito bom em termos de convergência para um problema com múltiplos ótimos locais. Esse resultado é esperado, já que uma das características do Arquivador Ideal é introduzir maior diversidade na busca. Um fato que influenciou a busca dos algoritmos foi o método de escolha de líder. O algoritmo CDAS-MOPSO também apresentou bons resultados em termos de convergência e superou o I-MOPSO e I-Sigma em termos de diversidade. Porém, o CDAS-MOPSO apresenta a limitação referente à escolha do parâmetro $S_i$. Na maioria

Tabela 7.16: Melhores algoritmos de acordo com o pós-teste do teste de Friedman: I-MOPSO x I-Sigma x CDAS-SMPSO x SMPSO.

| Prob | Obj | Melhores algoritmos | |
|---|---|---|---|
| | | GD | IGD |
| DTLZ2 | 3 | 0,4, 0,45, I-MOPSO e I-Sigma | 0,45, I-MOPSO e SMPSO |
| | 5 | 0,25, 0,3, I-MOPSO e I-Sigma | 0,35, 0,4 e I-MOPSO |
| | 10 | 0,3, I-MOPSO e I-Sigma | 0,35, 0,4 e 0,45 |
| | 15 | 0,3, I-MOPSO e I-Sigma | 0,35, 0,4 e 0,45 |
| | 20 | 0,3, I-MOPSO e I-Sigma | 0,35, 0,4 e 0,45 |
| DTLZ4 | 3 | 0,3, 0,4, 0,45, I-Sigma e SMPSO | 0,4, 0,45, I-MOPSO e SMPSO |
| | 5 | 0,45, I-MOPSO e I-Sigma | 0,4, 0,45, I-MOPSO e SMPSO |
| | 10 | I-MOPSO e I-Sigma | 0,35, 0,4, 0,45, I-MOPSO e SMPSO |
| | 15 | I-MOPSO e I-Sigma | 0,4 e 0,45 |
| | 20 | I-MOPSO e I-Sigma | 0,35, 0,4 e 0,45 |
| DTLZ6 | 2 | 0,3, 0,35, 0,4, 0,45 e SMPSO | 0,3, 0,35 and 0,4 |
| | 5 | I-Sigma | 0,3, 0,35 e SMPSO |
| | 10 | 0,3, e I-Sigma | 0,3, 0,35 e I-MOPSO |
| | 15 | 0,3, 0,45 e I-Sigma | 0,25, 0,3, 0,35 e I-MOPSO |
| | 20 | 0,35, 0,45 e I-Sigma | 0,25, 0,3, 0,35 e I-MOPSO |

dos cenários, as configurações que obtiveram a melhor diversidade não apresentaram os melhores valores em termos de convergência. Assim, para utilizar o CDAS-MOPSO é necessário definir o valor de $S_i$ de acordo com um objetivo específico, conseguir convergência ou diversidade.

### 7.2.6 Avaliação do arquivador hiperplano

O Arquivador Hiperplano proposto nesta tese utiliza um ponto de referência (escolhido através de um hiperplano) como base para guiar a busca para uma região previamente escolhida pelo usuário. Nesse conjunto de experimentos, a abordagem utilizada constrói o hiperplano no começo da busca. Além disso, duas estratégias são utilizadas, escolher um ponto na região central do hiperplano ou escolher um ponto próximo de um dos extremos do hiperplano. Em cada execução, o ponto é escolhido de forma aleatória dentro da área selecionada (meio ou extremos).

O objetivo desta análise empírica é mostrar que a busca de um algoritmo MOPSO que utiliza o Arquivador Hiperplano gera uma $PF_{aprox}$ em torno do ponto de referência escolhido. A ideia é que a busca é guiada pelo ponto de referência escolhido e é possível gerar um conjunto de pontos próximos a um ponto de referência através do uso de um método de arquivamento. Além disso, deseja-se verificar se esse comportamento se mantém mesmo em cenários com muitos objetivos.

Para medir se uma $PF_{aprox}$ está localizada em torno de um ponto de referência é

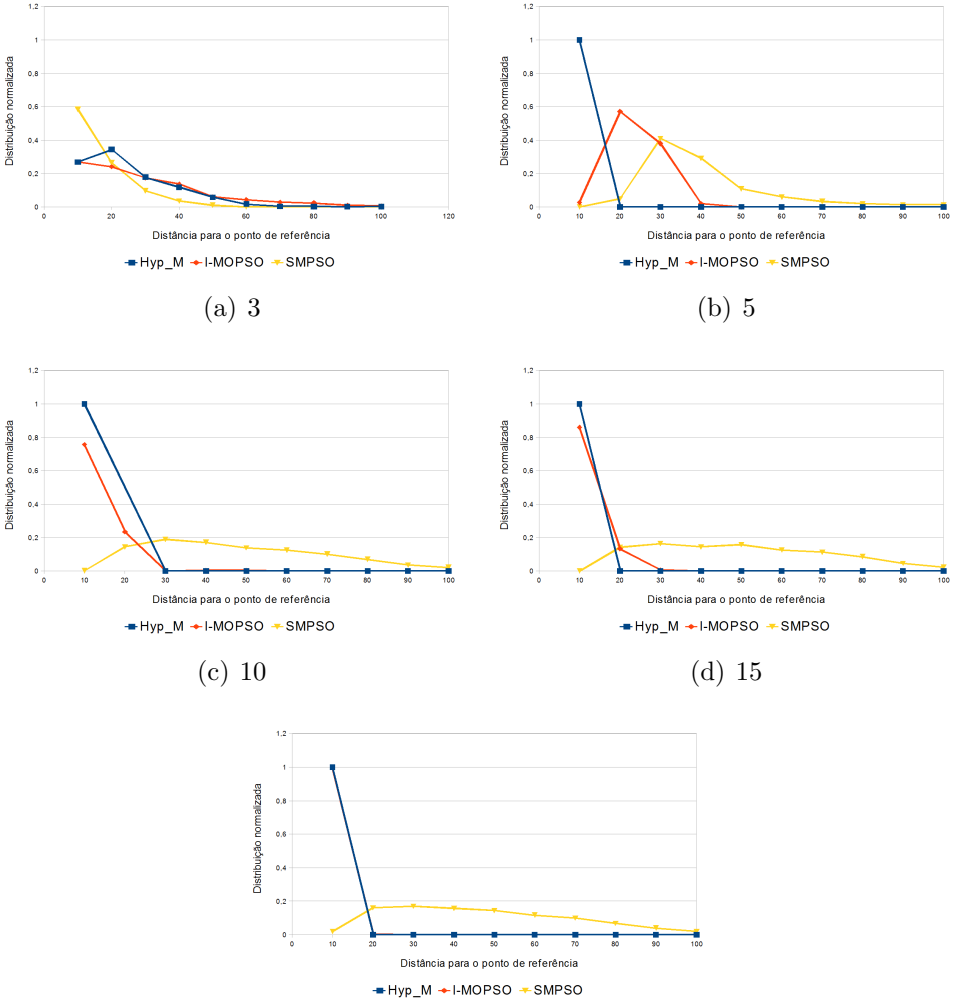
utilizada análise distribuição dos pontos sobre um ponto de referência (ver Seção 7.1.3). De forma semelhante que a análise de Tchebycheff, são utilizados histogramas para observar como os pontos de cada $PF_{aprox}$ estão localizados em relação ao ponto de referência. Esta análise compara diferentes $PF_{aprox}$ geradas por diferentes algoritmos e define se existe um algoritmo que está mais próximo do ponto de referência escolhido.

Nesta análise é utilizado o problema DTLZ2. São utilizados 4 algoritmos. Dois utilizam configurações diferentes do Arquivador Hiperplano: Hyp_M, algoritmo SMPSO com arquivador Hiperplano escolhendo pontos no centro do hiperplano, e Hyp_Ex algoritmo SMPSO com arquivador Hiperplano escolhendo pontos nos extremos do hiperplano. Além desses algoritmos são utilizados os algoritmos I-MOPSO e SMPSO, discutidos na seção anterior. Todos os algoritmos foram executados um total de 50000 avaliações de fitness e 20 execuções. A população foi definida com 200 partículas e o arquivo externo foi limitado em 200 soluções. Foi utilizado o problema DTLZ2 com 3, 5, 10, 15 e 20 números de funções objetivo.

A análise é feita em duas etapas. Primeiro comparando o Hyp_M com o I-MOPSO e SMPSO, depois comparando o Hyp_Ex com o I-MOPSO e SMPSO. Os pontos selecionados para o cálculo das distribuições são os pontos de referência escolhidos no início da execução do Arquivador Hiperplano. Os resultados são apresentados nas Figuras 7.36 e 7.37, para o Hyp_M e Hyp_Ex respectivamente. Essas figuras apresentam os gráficos onde o eixo $x$ representa a distância para o ponto de referência (por exemplo, 10% da distância, 20% da distância). O eixo $y$ representa o número de soluções geradas por cada algoritmo (variando entre 0 e 1 (0% a 100% do total de soluções)). Um ponto representa quantas soluções geradas por cada algoritmo estão localizadas naquela distância do ponto de referência (no espaço de objetivos). O intervalo das distâncias é obtido entre a solução mais próxima e a solução mais distante do ponto de referência, dentre todas as soluções de todos os algoritmos analisados.

Observando as distribuições do Hyp_M (ver Figura 7.36), nota-se que este algoritmo gera suas soluções mais próximas ao ponto de referência do que os demais. Para poucas funções objetivo os algoritmos geram distribuições semelhantes. Essas distribuições são

mais diversificadas, cobrindo diferentes distâncias aos pontos. Isto ocorre, pois em geral, para poucos objetivos os algoritmos cobrem a maioria das partes da fronteira de Pareto. Porém, quando o número de objetivos cresce o algoritmo Hyp_M concentra mais suas soluções próximas ao ponto de referência que os demais algoritmos.



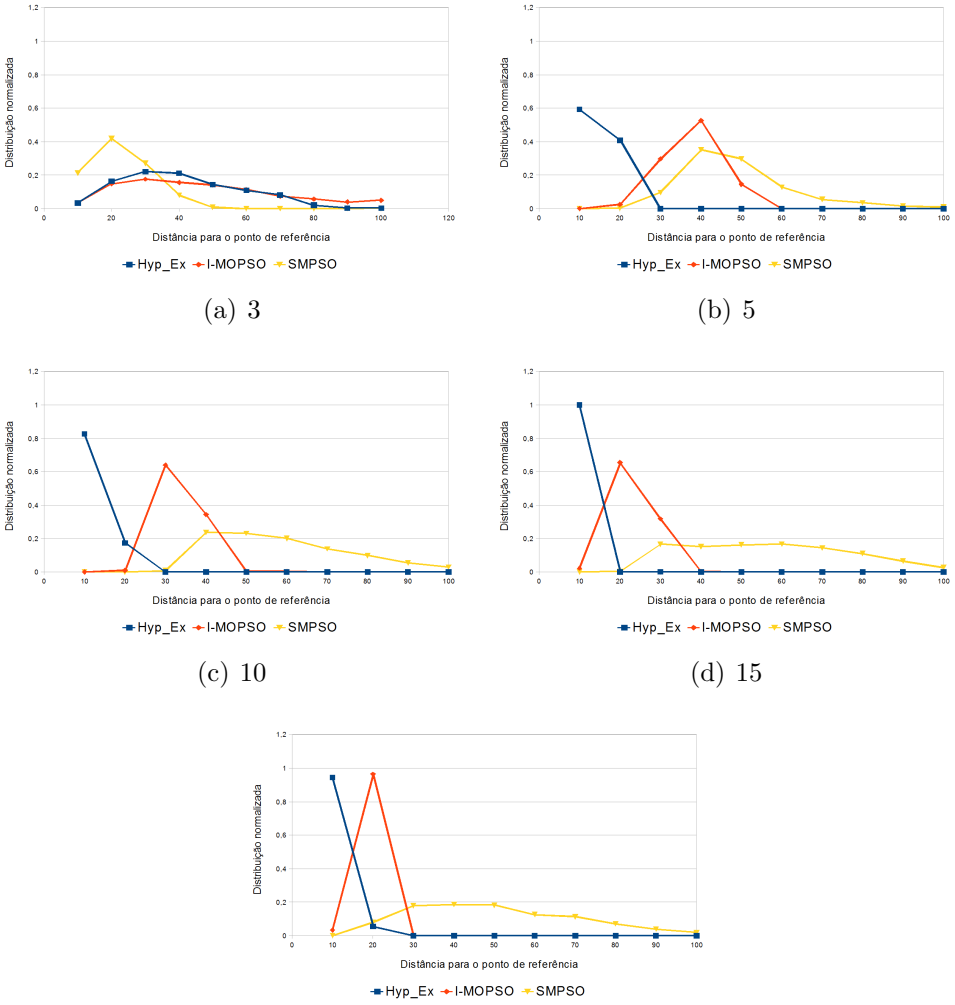

(a) 3 (b) 5

(c) 10 (d) 15

(e) 20

Figura 7.36: Distribuição sobre o ponto de referência para o Hyp_M, I-MOPSO e SMPSO.

O mesmo comportamento foi obtido pelo Hyp_Ex, como mostrado na Figura 7.36. Para 5 e 10 funções objetivo, esse algoritmo gerou todas as soluções distribuídas ao menos 30% distante do ponto de referência. Para 15 e 20 funções objetivo, esse algoritmo concentrou todas as suas soluções a uma distância de 10%. O I-MOPSO também consegue bons resultados, especialmente quando o ponto está localizado no centro, porém localiza suas soluções mais distantes do que os algoritmos baseados no Arquivador Hiperplano.




(a) 3 (b) 5

(c) 10 (d) 15

(e) 20

Figura 7.37: Distribuição sobre o ponto de referência para o Hyp_Ex, I-MOPSO e SMPSO.

Concluindo, é possível guiar a busca de um algoritmo MOPSO para uma região específica da fronteira de Pareto utilizando um método de arquivamento. Os resultados das abordagens que utilizam o Arquivador Hiperplano geraram uma $PF_{aprox}$ próxima ao ponto de referência selecionado.

### 7.2.7 Avaliação do algoritmo I-Multi

O último algoritmo desenvolvido, I-Multi, executa diferentes *sub-swarms* utilizando o Arquivador Ideal e limita a região de busca. O algoritmo possui duas fases distintas: primeiro a execução de uma busca inicial diversificada e em seguida a busca *multi-swarm*. Durante a segunda etapa do algoritmo três parâmetros são importantes e serão explorados nesses

experimentos: o tamanho da região de busca, o número de iterações de particionamento e o número de enxames.

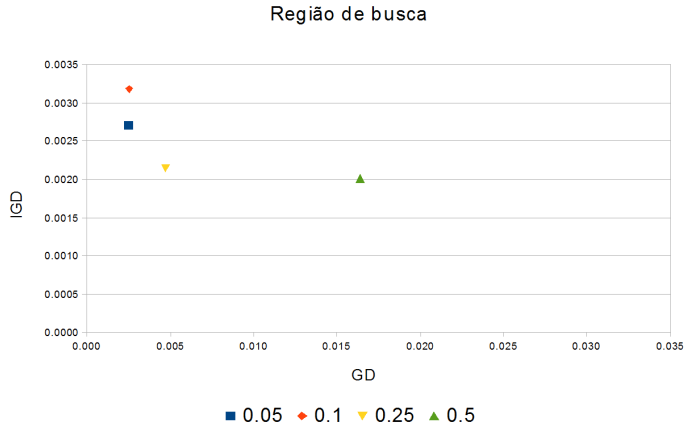
Todos os experimentos discutidos nas próximas seções possuem um conjunto de parâmetros em comum. Primeiro, o I-Multi tem como base o algoritmo SMPSO. Os parâmetros base do SMPSO apresentados na Tabela 7.1. Outra questão refere-se à fase de diversidade. Em todos os experimentos, essa fase é executada com a mesma configuração. Para introduzir maior diversidade é escolhido o Arquivador MGA [61]. Esse algoritmo MOPSO base é executado durante 100 gerações, sua população possui 200 partículas e o arquivo externo é limitado também com 200 soluções. Por fim, em todas as execuções todos os algoritmos foram executados um total de 20 vezes.

A próxima seção discute a configuração do parâmetro que define o tamanho da região de busca de cada *sub-swarm*. Em seguida, a Seção 7.2.7.2 executa o I-Multi com diferentes números de iterações de particionamento. A Seção 7.2.7.3 apresenta a configuração do número de enxames do algoritmo. Esse conjunto de experimentos é maior que o executado para os parâmetros anteriores. É dada mais ênfase à avaliação do número de enxames, pois a divisão da busca em diversos *sub-swarms* é o principal ponto do algoritmo. Por fim, a Seção 7.2.8 apresenta a comparação das melhores configurações do algoritmo I-Multi com o algoritmos SMPSO utilizando os arquivadores Ideal e MGA. O objetivo dessa comparação é mostrar que combinando os diferentes métodos de arquivamento é possível obter melhores resultados que executar cada método de arquivamento separadamente em um algoritmo MOPSO.

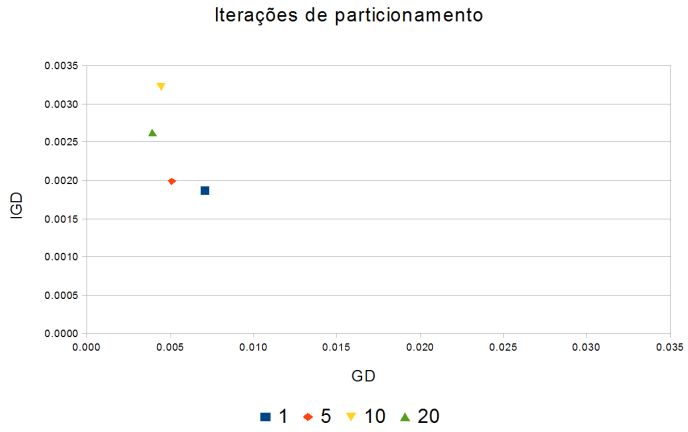
#### 7.2.7.1 Região de busca

No I-Multi, cada *sub-swarm* é definido por uma partícula semente. Este ponto base define o centro da região de busca no espaço das variáveis de decisão de cada enxame. Aqui é apresentada uma breve análise do parâmetro, trabalhos futuros vão estender esta análise e irão executar uma maior exploração deste parâmetro. A variante I-Multi Centróide foi escolhida pois o método para escolha das sementes já foi validado em [68]. O algoritmo é aplicado ao problema com 5 funções objetivo. O espaço de busca para este problema

está limitado num intervalo entre 0 e 1. O parâmetro da região de busca foi definido com quatro valores diferentes: $0,5$, $0,25$, $0,1$ e $0,05$. Estes valores variam de 50% a 5% do tamanho do espaço de busca. Os resultados são apresentados na Figura 7.38(a). Essa figura apresenta os valores de GD versus os valores de IGD. Assim, quanto mais próximo da origem melhor é o resultado do algoritmo tanto em convergência quanto em diversidade.




(a) Região de busca




(b) Número de iterações

Figura 7.38: Valores médios de GD X IGD para a variação do tamanho da região de busca e número de iterações de particionamento para o I-Multi Centroid , problema DTLZ2 com 5 funções objetivo.

Pode ser observado que, com uma maior região de busca, $0,5$, o melhor IGD e o pior GD foram obtidos, uma vez que com uma grande região de busca o algoritmo privilegiou diversidade ao invés de convergência. O inverso também é verdadeiro, quanto menor é o valor da região, $0,05$ ou $0,1$, melhor é o valor do GD e pior é o IGD. Para um tamanho

de $0,25$ o algoritmo obteve valores semelhantes de GD e IGD. Assim, quanto maior é a região mais diversificada é a busca realizada pelo algoritmo e quanto menor é o tamanho o algoritmo limita a sua pesquisa em uma região menor e obtém mais convergência. Assim, disposto a obter tanto a convergência e a diversidade, uma estratégia dinâmica é utilizada. Nessa estratégia o algoritmo varia o tamanho da região de exploração durante a execução. No início da pesquisa, a diversidade será favorecida e o tamanho da região de busca será ajustado para $0,5$. Em cada iteração de particionamento, o tamanho da região de busca será diminuído uniformemente favorecendo a convergência até atingir o valor mais baixo na última iteração. O valor final escolhido para os demais experimentos é 0.1.

#### 7.2.7.2 Número de iterações de particionamento

O número de iterações de particionamento representa o número de vezes em que os enxames irão efetuar troca de informação. Além disso, o número de iterações define o número de novas sementes escolhidas e o número de vezes que cada enxame será reiniciado. Para avaliação desse parâmetro é utilizada a mesma configuração da análise do parâmetro anterior: I-Multi Centroid aplicado ao DTLZ2 com 5 funções objetivo. Foram selecionados quatro diferentes números de iterações: 1, 5, 10 e 20. É importante salientar que, quanto maior o número de iterações, maior é o número das avaliações da função de fitness. Figura 7.38(b) apresenta os resultados do GD contra IGD.

Observando a Figura 7.38(b), quanto maior é o número de iterações o I-Multi perde diversidade e ganha convergência. No entanto, para 20 iterações o algoritmo obteve piores valores de GD do que a execução com 10 iterações. Assim, para o resto dos experimentos, é escolhido um número de 5 iterações de particionamento pois essa configuração apresentou um bom compromisso entre convergência e diversidade.

#### 7.2.7.3 Número de enxames

Este conjunto de experimentos varia o número de enxames do I-Multi. O objetivo é verificar se o número de enxames utilizado influencia a busca do algoritmo e se há influência qual é o número enxames que apresenta os melhores resultados. Diferentemente dos outros

Tabela 7.17: Número de enxames e partículas em cada enxame paro algoritmo I-Multi.

| **Número de enxames** | **Tamanho da população** |
|---|---|
| 3 | 250 |
| 5 | 150 |
| 10 | 75 |
| 30 | 25 |

parâmetros, esse parâmetro é testado com todos os métodos de escolha das partículas semente. O número de enxames influencia a escolha das partículas sementes, especialmente para o método I-Multi Extremos, como discutido na Seção 6.3.1. Assim, são usados três diferentes algoritmos: I-Multi Centroid, I-Multi Extremos e I-Multi Aleatório, cada um executando um método de escolha de sementes diferente.

Além disso a análise é feita em diferentes problemas. São utilizados os problemas DTLZ2, DTLZ4, DTLZ6 e DTLZ7 com 3, 5, 10 e 20 funções objetivo. O conjunto de indicadores utilizado também é maior. Nesta análise, para se ter uma visão maior dos aspectos da diversidade e da convergência da busca são utilizados os indicadores GD, IGD, *Convergence* e LD.

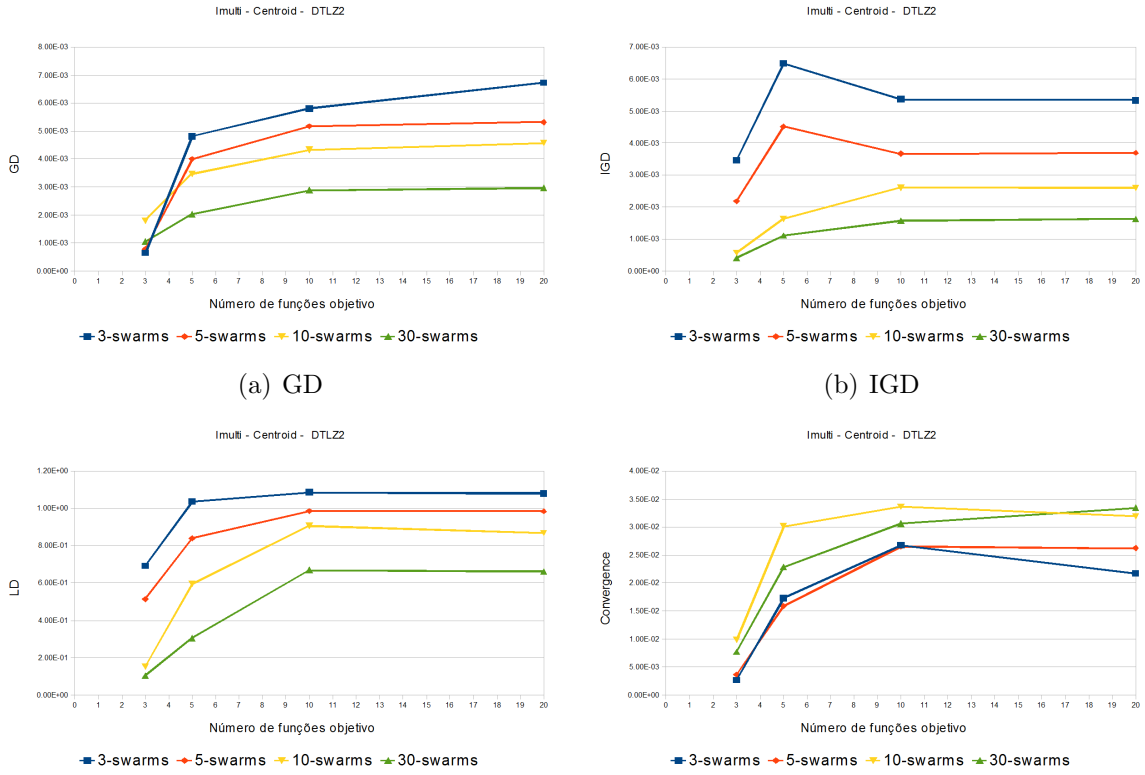
Os parâmetros do I-Multi foram definidos com base nos experimentos anteriores. Os parâmetros selecionados foram: região de exploração é definida inicialmente com $0,5$ (50% do espaço de busca dos problemas) e reduzida para $0,1$ (10% do espaço de busca) proporcionalmente em cada iteração de partição. O número de iterações de particionamento foi definido como 5.

Nestes experimentos, cada algoritmo é executado com os seguintes números de enxames: 3, 5, 10 e 30 enxames. Para evitar que haja um desbalanceamento na execução das diferentes configurações, todos os algoritmos deve executar um mesmo número de avaliações da função de fitness. Para que isso seja possível, foi definido que a soma do número de partículas de todos os *sub-swarms* deve ser igual. Assim, as configurações com menores números de enxames possuem um número maior de partículas. O número de partículas em cada *sub-swarms* é apresentado na Tabela 7.17. Para essas configurações o número de avaliações de fitness foi 385850. Cada algoritmo com as diferentes configuração dos parâmetros foi executado 30 vezes.

Os resultados desses experimentos seguem o padrão utilizado nesta tese. Os resultados do teste do pós-teste do teste de Friedman são apresentados em tabelas e as figuras mostram a média das execuções para os diferentes números de funções objetivos. As Tabelas 7.18, 7.19 e 7.20 apresentam os resultados do teste de Friedman, para o I-Multi Centroid, I-Multi Extreme e I-Multi Aleatório, respectivamente. Os valores contidos nas células mostram o valor da média para cada indicador e as células marcadas indicam as melhores configurações de acordo com o teste de Friedman. As Figuras 7.39 a 7.50, mostram a evolução dos indicadores.

O primeiro algoritmo analisado o I-Multi Centroid. Nesse método, as sementes de cada *sub-swarm* são definidas através do agrupamento das soluções do *font* base. Os resultados são apresentados na Tabela 7.18 e nas Figuras de número 7.39 até 7.42. O primeiro problema analisado é o DTLZ2. Observando os valores dos indicadores, a configuração com 30 enxames obteve o melhor resultado tanto em termos de convergência quanto em diversidade. Essa configuração obteve os melhores valores de GD para a maioria dos números de objetivos, em especial para muitos objetivos. Ela também conseguiu os melhores valores para as duas medidas utilizadas para medir diversidade, IGD e LD. Isso representa que a execução do algoritmo com 30 enxames gerou uma $PF_{aprox}$ que cobre uma área maior da fronteira de Pareto. Esse resultado é esperado, pois com mais *sub-swarm* é possível definir sementes em diferentes áreas do espaço de busca e assim obter maior diversidade. Para a medida $Convergence$, a configuração com 3 enxames obteve o melhor resultado. Esse indicador não considera a distância de toda a $PF_{aprox}$ em relação à $PFreal$, mas somente a melhor distância obtida. Assim, também é esperado um bom resultado dessa medida para o menor número de enxames, pois essa configuração possui o maior número de partículas por enxame. Esses resultados podem ser vistos na Figura 7.39. Para GD, IGD e LD a configuração com 30 enxames apresenta a menor deterioração das medidas e consegue manter o melhor valor dos indicadores para a maioria dos números de funções objetivo.

Em resumo, quanto maior o número de enxames, melhor é o resultado da busca para o problema DTLZ2. Um número maior de centróides produziu uma $PF_{aprox}$ que cobriu




(a) GD (b) IGD

(c) LD (d) *Convergence*

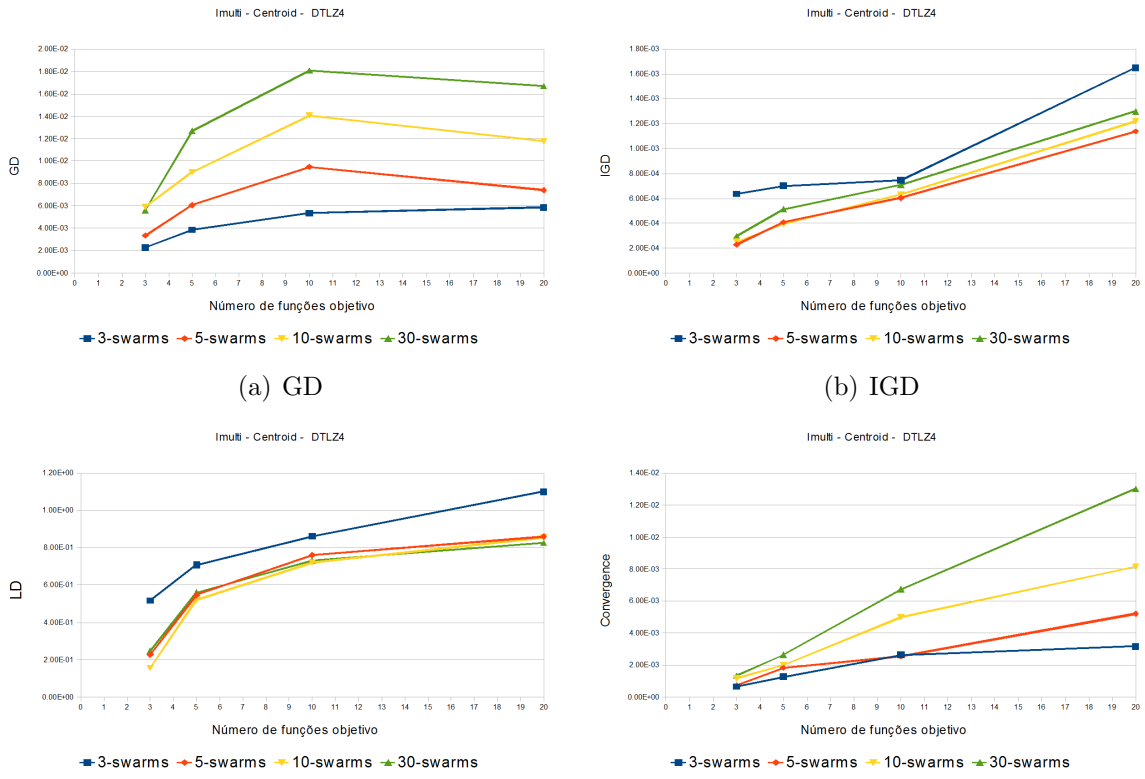
Figura 7.39: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Centroid, DTLZ2.

uma maior área e chegou mais perto da fronteira de Pareto. Além disso, a variante com 30 enxames teve a menor deterioração da busca, conseguindo bons valores para poucas e muitas funções objetivo.

O segundo problema é o DTLZ4. A característica principal desse problema é possuir uma região da fronteira de Pareto mais povoada. Aqui, as medidas de diversidade são mais importantes, já que um bom valor de convergência pode implicar que a busca ficou presa nessa região densa. Os resultados são apresentados na Tabela 7.18 e nas Figura 7.39. Pode ser observado que a variante com 3 enxames teve o melhor valor para os indicadores GD e *Convergence*. No entanto, essa configuração obteve um valor ruim de IGD e LD quando comparada às demais configurações. Conclui-se, que esse conjunto de enxames, embora tenha obtido um bom valor de GD, não obteve um bom resultado para o DTLZ4. A melhor configuração foi a com 5 enxames, já que obteve bons valores de IGD e LD e teve melhores valores de GD e *Convergence*. As variantes com 10 e 30 enxames obtiveram

valores similares de IGD e LD, em relação à configuração 5 enxames, mas apresentaram piores valores das medidas de convergência.

Em resumo, para DTLZ4, a configuração com o menor número de centróides apresentou um resultado ruim, já que o algoritmo não pode escapar da região mais densa. As outras variantes apresentaram os melhores resultados para as medidas de diversidade, mesmo para o número elevado de objetivos. Da mesma forma do que o DTLZ2, todos as variantes apresentaram alguma deterioração quando o número de objetivos aumentou, como pode ser observado nas Figura 7.40. No entanto, os algoritmos controlaram essa degradação quando o número de objetivos aumentou.




(a) GD (b) IGD

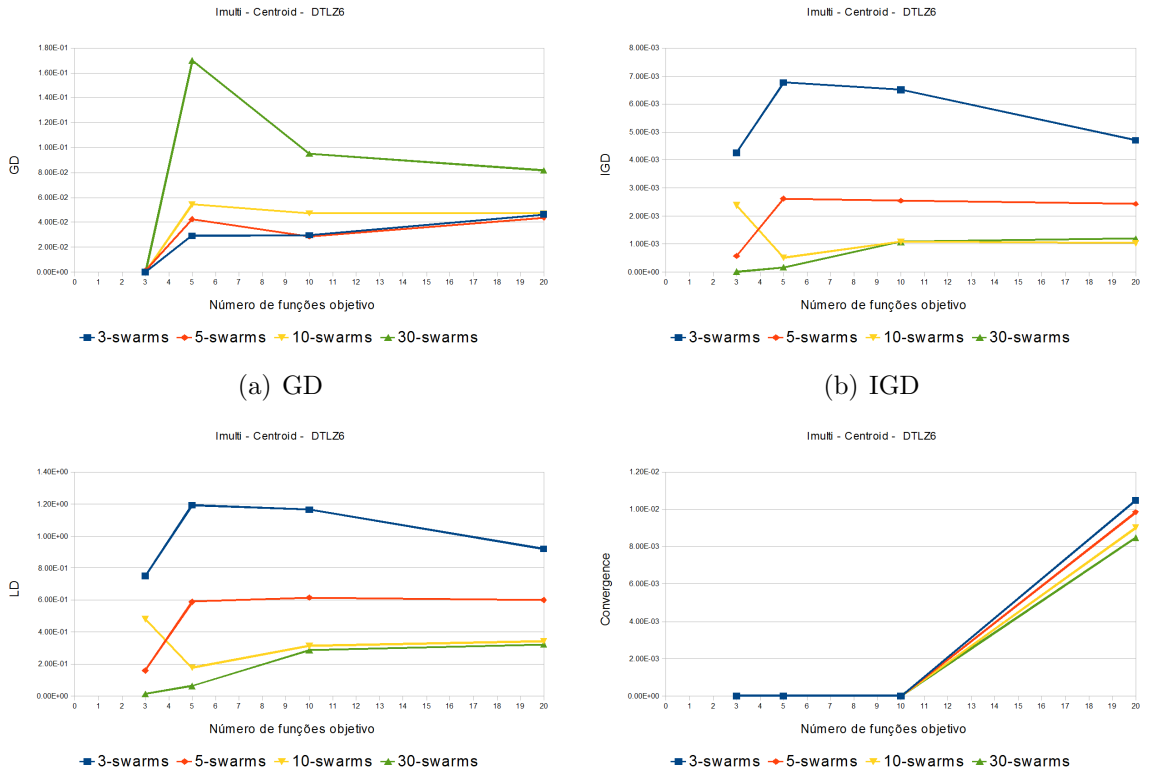
(c) LD (d) *Convergence*

Figura 7.40: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Centroid, DTLZ4.

Agora DTLZ6 é analisado. Aqui, uma vez que a fronteira de Pareto é representada por somente uma curva, a região a ser coberta é menor do que os problemas anteriores. Porém é mais difícil a convergência, já que há diversos ótimos locais. Em geral, todas as variantes obtiveram resultados semelhantes. As variantes com 30 e 10 enxames tiveram

melhor diversidade do que as outras (melhores valores de IGD e LD). Para o GD, as configurações com menor número de enxames apresentaram os melhores resultados. As configurações com 3, 5 e 10 enxames obtiveram um resultado equivalente. Para a medida *Convergence*, a variante com 30 enxames obteve o melhor resultado, para quase todos os números de objetivos. Essa medida mostra que essa configuração chegou perto $PF_{real}$ em regiões específicas. Porém o baixo valor de GD indica que essa configuração não conseguiu evitar ótimos locais em algumas regiões do espaço de objetivos.

Em resumo, as configurações com 10 e 30 enxames se destacaram. Essas configurações obtiveram uma busca com maior diversidade, porém não houve uma perda muito grande de convergência, uma vez que a variante com 10 enxames obteve um bom valor de GD e a com 30 enxames obteve o melhor *Convergence*. A Figura 7.41 mostra a variação dos indicadores e é possível perceber os bons resultados das medidas de diversidade para as configurações com maior número de enxames.




(a) GD

(b) IGD

(c) LD

(d) *Convergence*

Figura 7.41: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Centroid, DTLZ6.

Por fim, o último problema é o DTLZ7. Esse problema tem a característica de ter uma fronteira de Pareto desconexa. Assim, espera-se que a variante Centroid obtenha um bom resultado, uma vez que usa um algoritmo de agrupamento para definir a semente de cada enxame. Para esse problema, a variante com 30 enxames se destacou. Ela obteve os melhores valores para todas as medidas, em quase todos os números objetivos. Como o DTLZ7 tem várias subfronteiras, quando o número de objetivos cresce, um maior número de enxames cobre um maior número dessas fronteiras, implicando melhores valores de GD e IGD. A análise do IGD e LD para DTLZ7 mostra uma característica interessante do problema. O número de fronteiras cresce de forma exponencial, assim, para um número muito grande de funções objetivo sempre vai haver um conjunto de subfronteiras não cobertas pelos algoritmos. Nesse sentido, sempre há uma conjunto de pontos no espaço de objetivos longe da $PF_{aprox}$, assim, para muitos objetivos o valor do LD é similar para todos as configurações. Porém, fazendo uma análise conjunta do IGD e LD é possível perceber uma diferença a favor da configuração com 30 enxames. Para o indicador *Convergence*, os resultados são semelhantes para todas as variantes, para quase todos os números objetivos. Esses resultados mostram que cada variante aproximou-se de pelo menos uma subfronteira.

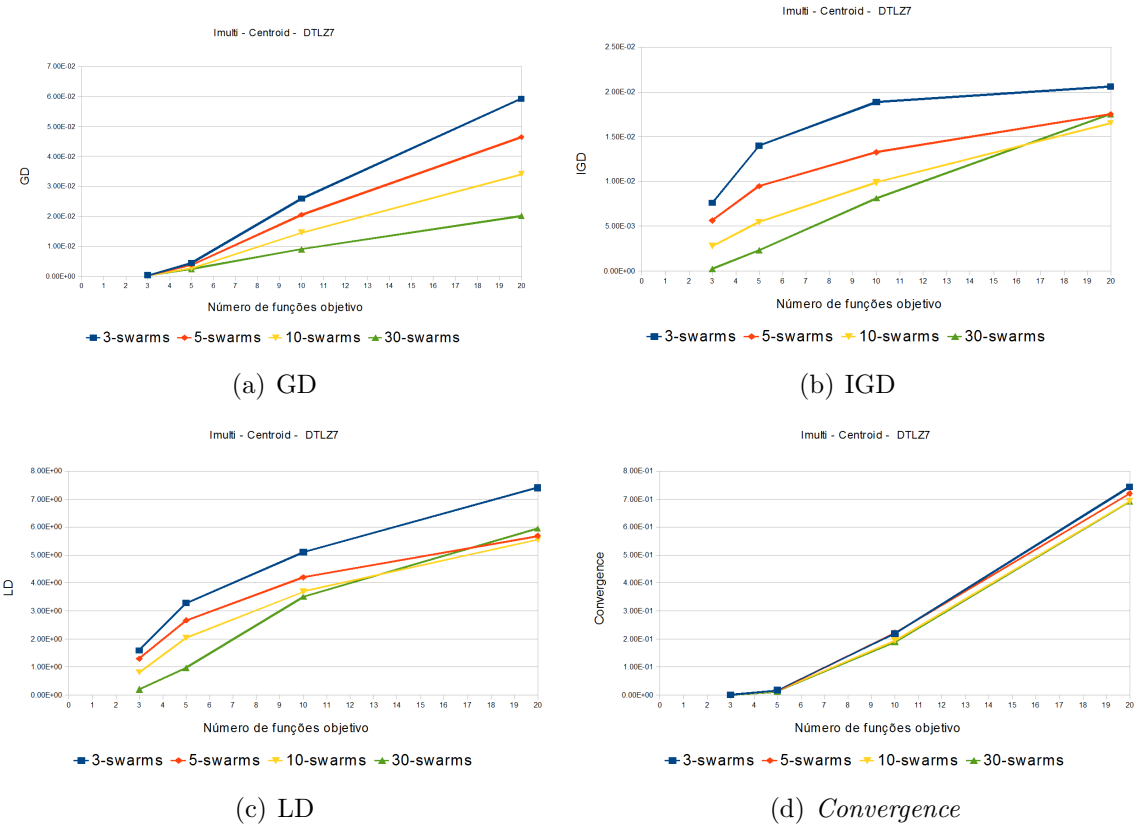
Em resumo, as configurações com o maior número de enxames obtiveram os melhores resultados, uma vez que cobriram um maior número de subfronteiras. No entanto, quando o número de subfronteiras cresceu, a diferença entre as configurações for menor. Na Figura 7.42, podemos ver o comportamento de cada medida quando o número de objetivos cresce. Pode ser observado que a variante com 30 enxames obteve a menor deterioração para GD. Para as outras medidas, a deterioração foi semelhante. Todas as variantes deterioraram quando o número de objetivos cresceu.

Fazendo uma análise dos resultados de todos os problemas, conclui-se que quanto maior o número de centróides melhor é o comportamento para o algoritmo I-Multi Centroid. Em geral, para os diferentes problemas, a variante com maior número de enxames obteve a melhor convergência e diversidade, especialmente para espaços de objetivos grandes.

A segunda variante utilizada é a I-Multi Extremos. Essa abordagem seleciona alguns *sub-swarms* para otimizar regiões próximas a cada um dos eixos do espaço de busca. A

Tabela 7.18: I-Multi Centroid

| | | DTLZ2 | | | | DTLZ4 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Swarms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **3** | 6,51E-04 | 3,45E-03 | 6,92E-01 | 2,66E-03 | 2,28E-03 | 6,36E-04 | 5,18E-01 | 6,46E-04 |
| | **5** | 7,87E-04 | 2,18E-03 | 5,14E-01 | 3,61E-03 | 3,33E-03 | 2,29E-04 | 2,26E-01 | 7,25E-04 |
| | **10** | 1,80E-03 | 5,71E-04 | 1,53E-01 | 9,87E-03 | 5,89E-03 | 2,52E-04 | 1,56E-01 | 1,16E-03 |
| | **30** | 1,05E-03 | 4,12E-04 | 1,04E-01 | 7,72E-03 | 5,62E-03 | 2,97E-04 | 2,47E-01 | 1,35E-03 |
| **5** | **3** | 4,81E-03 | 6,49E-03 | 1,04E+00 | 1,73E-02 | 3,85E-03 | 7,00E-04 | 7,08E-01 | 1,27E-03 |
| | **5** | 4,00E-03 | 4,52E-03 | 8,40E-01 | 1,59E-02 | 6,07E-03 | 4,08E-04 | 5,47E-01 | 1,83E-03 |
| | **10** | 3,47E-03 | 1,63E-03 | 5,97E-01 | 3,01E-02 | 9,01E-03 | 3,93E-04 | 5,21E-01 | 2,00E-03 |
| | **30** | 2,03E-03 | 1,11E-03 | 3,05E-01 | 2,28E-02 | 1,27E-02 | 5,12E-04 | 5,60E-01 | 2,64E-03 |
| **10** | **3** | 5,80E-03 | 5,37E-03 | 1,09E+00 | 2,67E-02 | 5,36E-03 | 7,46E-04 | 8,61E-01 | 2,63E-03 |
| | **5** | 5,17E-03 | 3,67E-03 | 9,85E-01 | 2,65E-02 | 9,47E-03 | 6,04E-04 | 7,60E-01 | 2,54E-03 |
| | **10** | 4,33E-03 | 2,61E-03 | 9,07E-01 | 3,36E-02 | 1,41E-02 | 6,32E-04 | 7,20E-01 | 4,99E-03 |
| | **30** | 2,88E-03 | 1,57E-03 | 6,70E-01 | 3,06E-02 | 1,81E-02 | 7,09E-04 | 7,31E-01 | 6,73E-03 |
| **20** | **3** | 6,72E-03 | 5,33E-03 | 1,08E+00 | 2,17E-02 | 5,84E-03 | 1,65E-03 | 1,10E+00 | 3,17E-03 |
| | **5** | 5,32E-03 | 3,70E-03 | 9,84E-01 | 2,62E-02 | 7,39E-03 | 1,14E-03 | 8,59E-01 | 5,21E-03 |
| | **10** | 4,56E-03 | 2,59E-03 | 8,68E-01 | 3,19E-02 | 1,18E-02 | 1,22E-03 | 8,55E-01 | 8,14E-03 |
| | **30** | 2,97E-03 | 1,64E-03 | 6,61E-01 | 3,34E-02 | 1,67E-02 | 1,30E-03 | 8,26E-01 | 1,30E-02 |
| | | **DTLZ6** | | | | **DTLZ7** | | | |
| **Obj** | **Swarms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **3** | 2,28E-03 | 2,24E-03 | 4,36E-01 | 2,33E-07 | 3,45E-04 | 7,60E-03 | 1,59E+00 | 2,45E-04 |
| | **5** | 5,82E-06 | 2,00E-03 | 4,17E-01 | 2,13E-07 | 3,04E-04 | 5,65E-03 | 1,29E+00 | 2,29E-04 |
| | **10** | 4,03E-06 | 5,06E-04 | 1,10E-01 | 1,12E-07 | 2,80E-04 | 2,78E-03 | 8,04E-01 | 1,83E-04 |
| | **30** | 2,79E-06 | 1,47E-05 | 9,34E-03 | 3,11E-08 | 3,51E-04 | 2,39E-04 | 2,03E-01 | 1,33E-04 |
| **5** | **3** | 2,78E-01 | 3,01E-03 | 5,42E-01 | 1,58E-05 | 4,47E-03 | 1,40E-02 | 3,28E+00 | 1,77E-02 |
| | **5** | 6,78E-02 | 1,76E-03 | 4,63E-01 | 2,63E-07 | 3,84E-03 | 9,48E-03 | 2,66E+00 | 1,74E-02 |
| | **10** | 1,01E-01 | 1,57E-03 | 4,14E-01 | 9,17E-07 | 2,74E-03 | 5,47E-03 | 2,04E+00 | 1,57E-02 |
| | **30** | 1,85E-01 | 1,14E-03 | 2,92E-01 | 1,50E-06 | 2,38E-03 | 2,31E-03 | 9,64E-01 | 1,33E-02 |
| **10** | **3** | 1,42E-01 | 6,61E-03 | 1,09E+00 | 3,85E-02 | 2,59E-02 | 1,89E-02 | 5,10E+00 | 2,20E-01 |
| | **5** | 1,14E-01 | 6,37E-03 | 1,01E+00 | 6,68E-02 | 2,05E-02 | 1,33E-02 | 4,20E+00 | 2,22E-01 |
| | **10** | 8,20E-02 | 3,03E-03 | 7,21E-01 | 1,34E-06 | 1,46E-02 | 9,88E-03 | 3,69E+00 | 1,95E-01 |
| | **30** | 1,08E-01 | 3,48E-03 | 6,92E-01 | 2,95E-05 | 9,00E-03 | 8,10E-03 | 3,52E+00 | 1,90E-01 |
| **20** | **3** | 1,24E-02 | 8,70E-03 | 1,41E+00 | 1,21E-01 | 6,38E-02 | 2,06E-02 | 7,41E+00 | 7,44E-01 |
| | **5** | 1,44E-02 | 8,47E-03 | 1,39E+00 | 1,28E-01 | 4,64E-02 | 1,75E-02 | 5,68E+00 | 7,20E-01 |
| | **10** | 4,29E-02 | 4,53E-03 | 8,52E-01 | 1,61E-01 | 3,40E-02 | 1,65E-02 | 5,54E+00 | 6,92E-01 |
| | **30** | 8,03E-02 | 3,42E-03 | 7,40E-01 | 3,74E-02 | 2,02E-02 | 1,75E-02 | 5,95E+00 | 6,91E-01 |




(a) GD (b) IGD

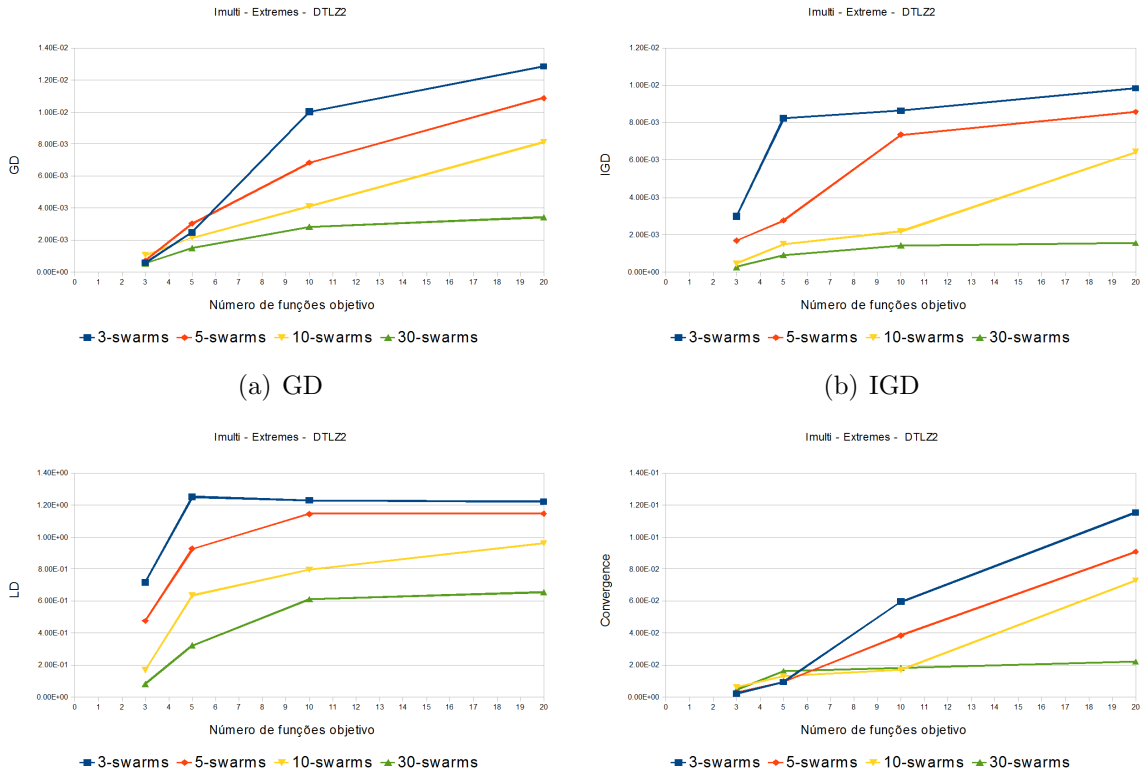
(c) LD (d) *Convergence*

Figura 7.42: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Centroid, DTLZ7.

ideia é, em cada iteração, selecionar $m$ soluções do conjunto base nos extremos de cada um dos eixos de coordenadas para ser a semente de $m$ enxames. Se o número de enxames é maior do que o número de objetivos, as sementes dos enxames restantes são selecionadas aleatoriamente. Se esse número for menor que o número de objetivos, algumas dimensões não são escolhidas.

Observando os resultados para o problema DTLZ2 na Tabela 7.19, os resultados da configuração com 30 enxames tiveram os melhores resultados. Essa configuração obteve o melhor valor de GD, IGD, LD e *Convergence* para quase todas as funções objetivo. Para um pequeno número de objetivos, 3 ou 5, as configurações com poucos enxames obtiveram alguns bons valores de GD ou *Convergence*. No entanto, quando o número de objetivos cresceu o número enxames tornou-se menor do que o número de dimensões e os algoritmos com poucos enxames perderam desempenho. Em resumo, para o DTLZ2, o I-Multi Extremo com um maior número de enxames obteve o melhores resultado. Além

disso, se o número de enxames é menor do que os números dos objetivos o algoritmo não funcionam bem. Observando a Figura 7.43, pode-se notar a deterioração da busca de cada configuração quando o número de objetivos cresce. Se o número de objetivos é menor que o número de enxames, todas as variantes apresentam um resultado semelhante. Porém, quando o número de objetivos é maior que o número de enxames da configuração o algoritmo perde desempenho em todos os indicadores.



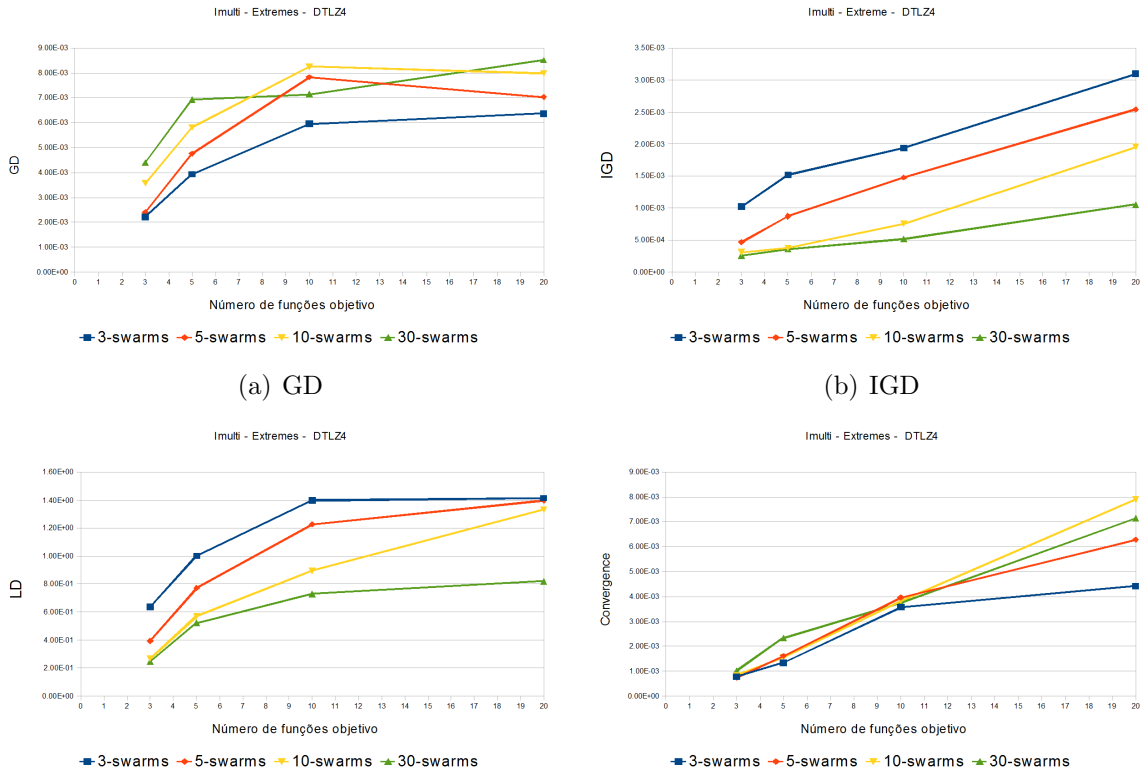

(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.43: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Extremos, DTLZ2.

Os resultados para o problema DTLZ4 são semelhantes aos do DTLZ2. Mais uma vez, quando o número de enxames é menor que o número de dimensões o algoritmo obteve um resultado ruim. As variantes com 3 e 5 enxames, apresentaram o melhor GD e *Convergence*, para quase todos os números de objetivos. No entanto, tiveram piores valores das medidas de diversidade, IGD e LD, em comparação com as outras configurações. Isso indica que a busca dessas configurações foi direcionada para a região mais densa da fronteira do Pareto e não houve uma boa diversidade. Ao analisar DTLZ4 observa-se se

o algoritmo consegue evitar as áreas mais densas do espaço objetivo e gera uma $PF_{aprox}$ bem diversificada. Assim, a configuração com 30 enxames obteve os melhores valores de IGD e LD, obtendo melhor diversidade. Porém, essa configuração teve dificuldade para convergir para a fronteira de Pareto. Essa dificuldade em convergir ocorre principalmente devido ao pequeno número de soluções na população de cada *sub-swarm* dessa configuração. Na Figura 7.44, pode-se observar que a variante com 30 enxames tem uma pequena deterioração para as medidas de diversidade. Porém, mais uma vez quando o número de objetivos torna-se maior do que o número de enxames e o algoritmo sofreu uma deterioração grande. Por exemplo, para IGD (Figura 7.44(b)), os resultados da configuração com 10 enxames é similar para 5 e 10 funções objetivo, porém sofre uma grande deterioração para o problema com 20 funções.




(a) GD (b) IGD

(c) LD (d) *Convergence*

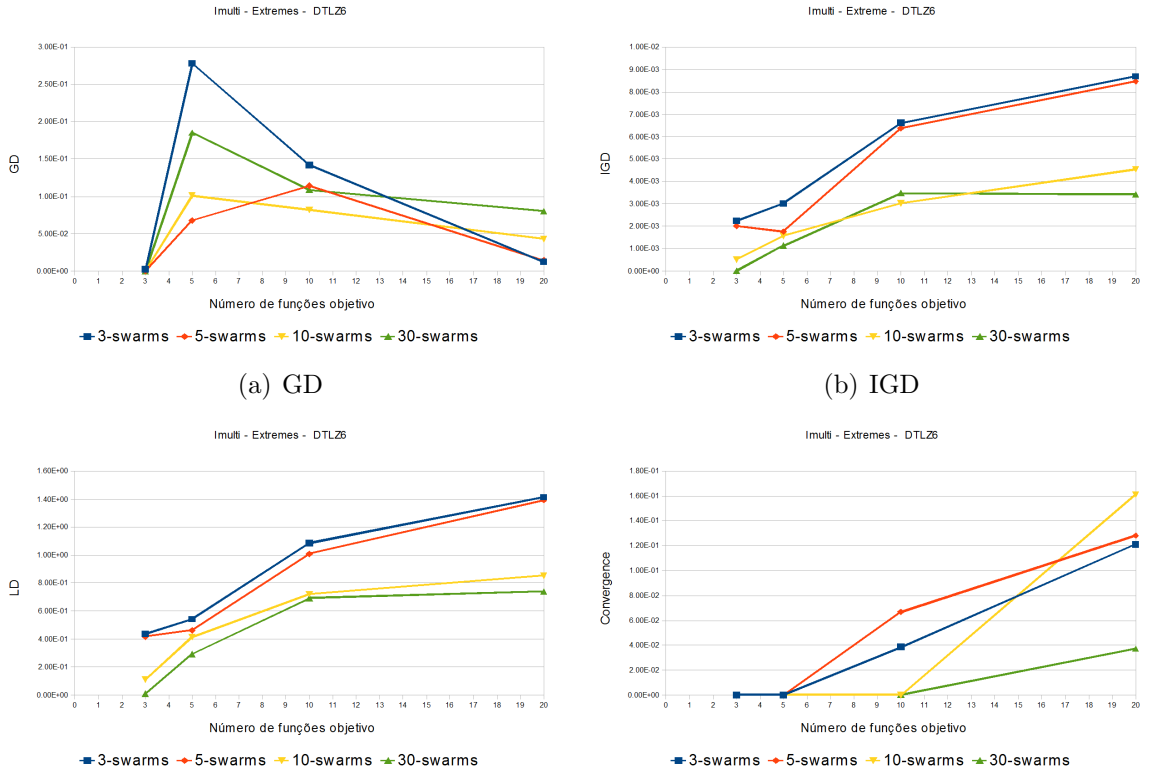
Figura 7.44: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Extremos, DTLZ4.

Para o DTLZ6, não foi possível identificar qual a configuração se destacou. Observando os resultados na tabela 7.19, as variantes com 3 , 5 e 10 enxames obtiveram o melhor

GD, para as maiores das dimensões. À medida que essas variantes têm mais soluções em suas populações, houve uma maior convergência na busca e foi produzido uma$PF_{aprox}$ mais perto da $PF_{real}$. Observando a Figura 7.45 nota-se o comportamento do I-Multi Extremos para o DTLZ6. Em especial, os valores de GD melhoram quando o número de objetivos cresceu. Isso ocorre, pois quando o número de objetivos cresce o algoritmo somente escolhe sementes próximas aos extremos. Assim, houve um ganho de convergência quando o algoritmo escolheu somente essas sementes. Porém, como o algoritmo somente privilegiou convergência, as medidas de diversidade deterioraram, pois a busca focou em pequenas regiões do espaço de objetivos. A configuração com 30 enxames apresentou um resultado semelhante ao I-Multi Centroid. Essa configuração obteve a melhor diversidade, porém obteve o melhor valor de *Convergence*. Isso mostra que com 30 enxames o I-Multi Extremos chegou perto da fronteira em alguns pontos, porém como privilegia a diversidade não consegue chegar perto fronteira em todas as regiões, devido aos ótimos locais.

Em resumo, os algoritmos com menor número de enxames obtiveram melhor convergência e evitaram mais ótimos locais. No entanto, a configuração com 30 enxames obteve bons resultados para o menor número de objetivos e obteve maior diversidade. Para obter o melhor desempenho para este problema, é sugerida a utilização de um alto número de enxames, porém utilizando mais partículas em cada enxame.

Para o DTLZ7, I-Multi Extremos obteve um comportamento semelhante ao obtido quando aplicado ao DTLZ2. À medida que o número de objetivos aproximou-se do número de enxames, o algoritmo piorou o desempenho. Observando GD, IGD e LD, para 3 funções objetivos, as configurações com 3 e5 enxames apresentaram bons resultados. No entanto, quando o número de objetivo cresceu, esses algoritmos sofreram uma deterioração grande, tanto em termos de convergência quanto em diversidade. Assim, para os demais números de objetivos a configuração com 30 enxames obteve os melhores resultados. Para o indicador *Convergence*, todos os diferentes números de enxames obtiveram um resultado semelhante. Cada configuração se aproximou de pelo menos uma fronteira desconexa e todos tiveram resultados equivalentes para esse indicador. Observando a Figura 7.46, todas as configurações apresentaram deterioração dos indicadores quando o número de




(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.45: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Extremos, DTLZ6.

objetivos cresceu, mesmo a configuração com 30 enxames. Isso ocorre devido ao alto número de subfronteiras desconexas do problema DTLZ7 com muitos objetivos.

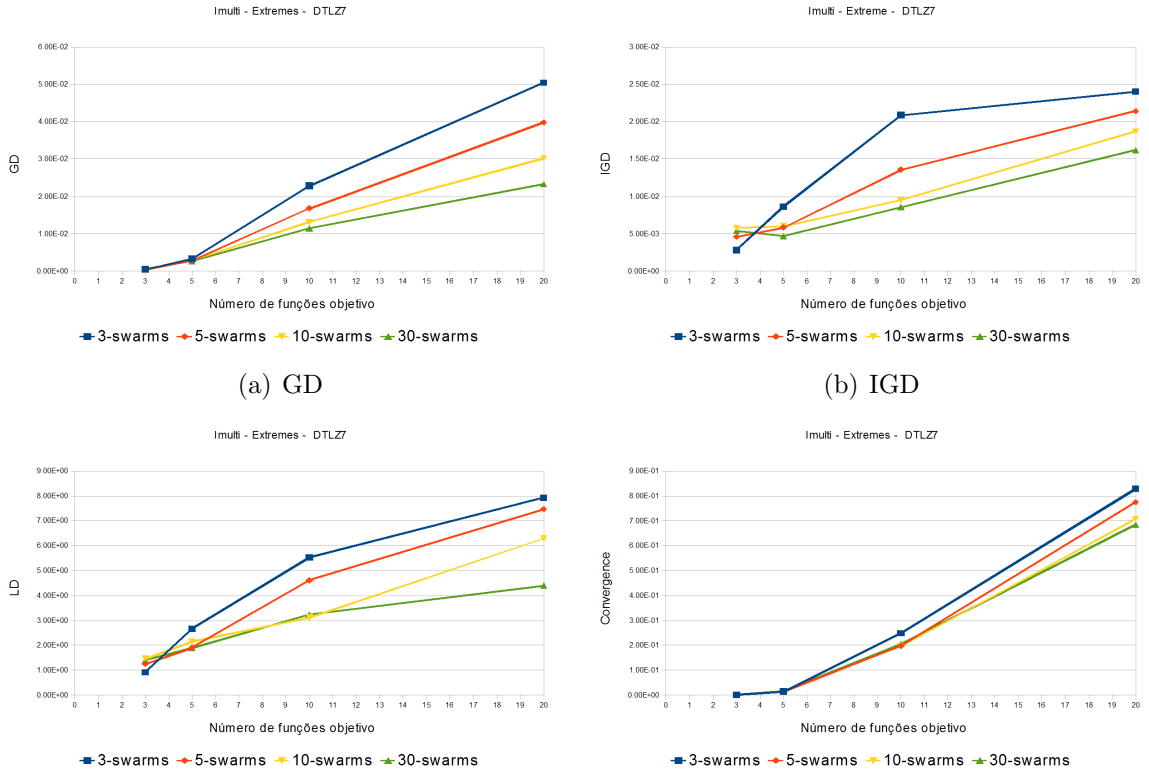
Em resumo, o I-Multi Extremos funciona bem quando o número de enxames é menor que o número de funções objetivo. Quando isto acontece, o algoritmo aumenta a convergência em direção a cada dimensão do espaço de busca, mas também possui boa diversidade, devido às sementes escolhidas de forma aleatória. Devido a essa questão, para muitos objetivos, os melhores resultados são obtidos utilizando um conjunto grande de enxames. Em geral, quanto maior o número de enxames, maior é a diversidade de pesquisa. Este comportamento, por vezes, evita uma boa convergência, como no problema DTLZ6. No entanto o aumento do número de partículas em cada enxame pode introduzir maior convergência ao algoritmo.

A última variante analisada é o de I-Multi Aleatório. Diferente das variantes anteriores, esta abordagem não extrai qualquer informação adicional a partir do conjunto inicial de

Tabela 7.19: I-Multi Extremos

| | | DTLZ2 | | | | DTLZ4 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Swarms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **3** | 5,69E-04 | 2,99E-03 | 7,18E-01 | 2,10E-03 | 2,22E-03 | 1,02E-03 | 6,36E-01 | 7,85E-04 |
| | **5** | 7,04E-04 | 1,68E-03 | 4,76E-01 | 2,86E-03 | 2,40E-03 | 4,67E-04 | 3,94E-01 | 7,36E-04 |
| | **10** | 1,06E-03 | 4,45E-04 | 1,67E-01 | 6,18E-03 | 3,56E-03 | 3,08E-04 | 2,67E-01 | 8,29E-04 |
| | **30** | 5,53E-04 | 2,75E-04 | 8,15E-02 | 4,50E-03 | 4,39E-03 | 2,57E-04 | 2,45E-01 | 1,01E-03 |
| **5** | **3** | 2,49E-03 | 8,25E-03 | 1,25E+00 | 9,31E-03 | 3,94E-03 | 1,52E-03 | 1,00E+00 | 1,34E-03 |
| | **5** | 3,03E-03 | 2,76E-03 | 9,26E-01 | 9,55E-03 | 4,76E-03 | 8,72E-04 | 7,72E-01 | 1,61E-03 |
| | **10** | 2,16E-03 | 1,49E-03 | 6,35E-01 | 1,29E-02 | 5,81E-03 | 3,76E-04 | 5,70E-01 | 1,55E-03 |
| | **30** | 1,50E-03 | 9,08E-04 | 3,23E-01 | 1,61E-02 | 6,93E-03 | 3,52E-04 | 5,21E-01 | 2,34E-03 |
| **10** | **3** | 1,00E-02 | 8,65E-03 | 1,23E+00 | 5,96E-02 | 5,95E-03 | 1,94E-03 | 1,40E+00 | 3,57E-03 |
| | **5** | 6,83E-03 | 7,35E-03 | 1,14E+00 | 3,85E-02 | 7,83E-03 | 1,48E-03 | 1,23E+00 | 3,96E-03 |
| | **10** | 4,10E-03 | 2,17E-03 | 7,95E-01 | 1,71E-02 | 8,25E-03 | 7,55E-04 | 8,94E-01 | 3,83E-03 |
| | **30** | 2,82E-03 | 1,42E-03 | 6,10E-01 | 1,83E-02 | 7,14E-03 | 5,13E-04 | 7,29E-01 | 3,75E-03 |
| **20** | **3** | 1,29E-02 | 9,84E-03 | 1,22E+00 | 1,15E-01 | 6,37E-03 | 3,10E-03 | 1,41E+00 | 4,42E-03 |
| | **5** | 1,09E-02 | 8,58E-03 | 1,15E+00 | 9,08E-02 | 7,03E-03 | 2,54E-03 | 1,40E+00 | 6,28E-03 |
| | **10** | 8,11E-03 | 6,42E-03 | 9,60E-01 | 7,28E-02 | 7,97E-03 | 1,95E-03 | 1,33E+00 | 7,89E-03 |
| | **30** | 3,42E-03 | 1,56E-03 | 6,55E-01 | 2,21E-02 | 8,53E-03 | 1,05E-03 | 8,21E-01 | 7,14E-03 |

| | | DTLZ6 | | | | DTLZ7 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Swarms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **3** | 2,28E-03 | 2,24E-03 | 4,36E-01 | 2,33E-07 | 4,63E-04 | 2,81E-03 | 9,15E-01 | 1,85E-04 |
| | **5** | 5,82E-06 | 2,00E-03 | 4,17E-01 | 2,13E-07 | 3,80E-04 | 4,58E-03 | 1,24E+00 | 2,16E-04 |
| | **10** | 4,03E-06 | 5,06E-04 | 1,10E-01 | 1,12E-07 | 2,88E-04 | 5,79E-03 | 1,46E+00 | 2,25E-04 |
| | **30** | 2,79E-06 | 1,47E-05 | 9,34E-03 | 3,11E-08 | 5,11E-04 | 5,38E-03 | 1,39E+00 | 1,96E-04 |
| **5** | **3** | 2,78E-01 | 3,01E-03 | 5,42E-01 | 1,58E-05 | 3,25E-03 | 8,63E-03 | 2,66E+00 | 1,44E-02 |
| | **5** | 6,78E-02 | 1,76E-03 | 4,63E-01 | 2,63E-07 | 2,85E-03 | 5,83E-03 | 1,90E+00 | 1,44E-02 |
| | **10** | 1,01E-01 | 1,57E-03 | 4,14E-01 | 9,17E-07 | 2,90E-03 | 5,98E-03 | 2,14E+00 | 1,50E-02 |
| | **30** | 1,85E-01 | 1,14E-03 | 2,92E-01 | 1,50E-06 | 2,67E-03 | 4,69E-03 | 1,89E+00 | 1,65E-02 |
| **10** | **3** | 1,42E-01 | 6,61E-03 | 1,09E+00 | 3,85E-02 | 2,28E-02 | 2,09E-02 | 5,53E+00 | 2,49E-01 |
| | **5** | 1,14E-01 | 6,37E-03 | 1,01E+00 | 6,68E-02 | 1,68E-02 | 1,35E-02 | 4,62E+00 | 1,98E-01 |
| | **10** | 8,20E-02 | 3,03E-03 | 7,21E-01 | 1,34E-06 | 1,31E-02 | 9,50E-03 | 3,11E+00 | 1,96E-01 |
| | **30** | 1,08E-01 | 3,48E-03 | 6,92E-01 | 2,95E-05 | 1,14E-02 | 8,52E-03 | 3,23E+00 | 2,04E-01 |
| **20** | **3** | 1,24E-02 | 8,70E-03 | 1,41E+00 | 1,21E-01 | 5,04E-02 | 2,40E-02 | 7,93E+00 | 8,29E-01 |
| | **5** | 1,44E-02 | 8,47E-03 | 1,39E+00 | 1,28E-01 | 3,98E-02 | 2,14E-02 | 7,46E+00 | 7,75E-01 |
| | **10** | 4,29E-02 | 4,53E-03 | 8,52E-01 | 1,61E-01 | 3,02E-02 | 1,87E-02 | 6,29E+00 | 7,08E-01 |
| | **30** | 8,03E-02 | 3,42E-03 | 7,40E-01 | 3,74E-02 | 2,33E-02 | 1,62E-02 | 4,40E+00 | 6,84E-01 |




(a) GD (b) IGD

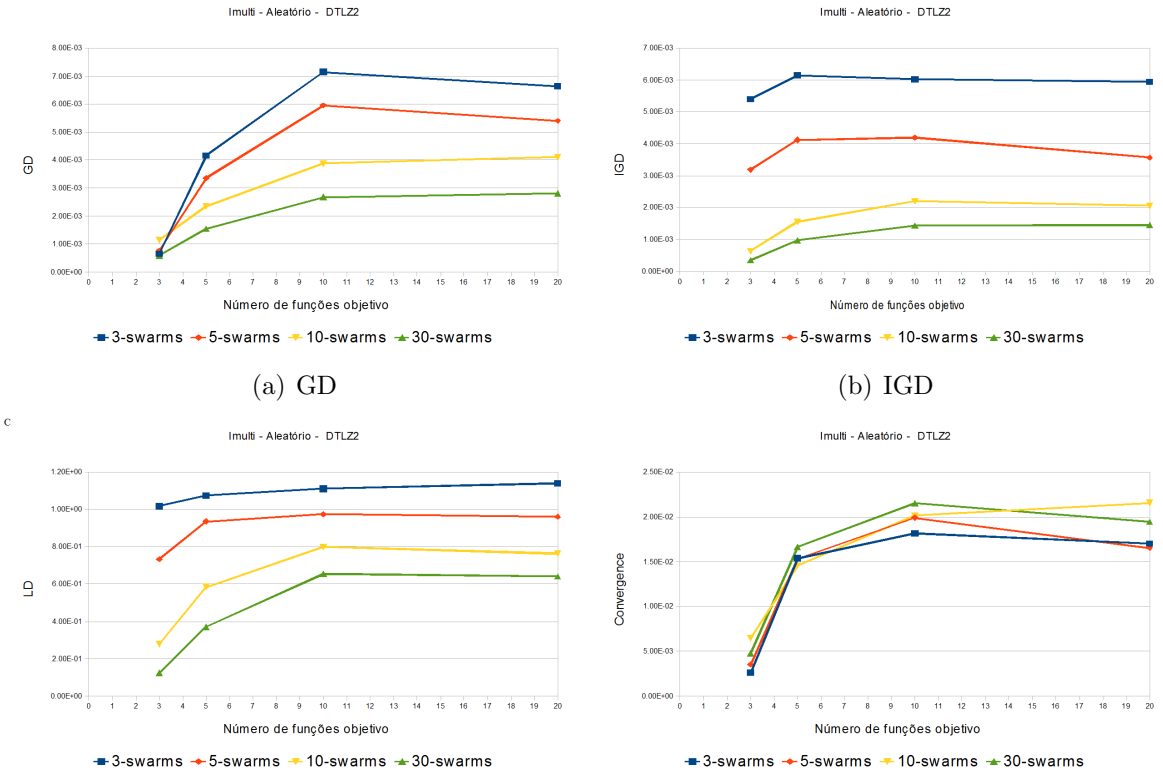
(c) LD (d) *Convergence*

Figura 7.46: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Extremos, DTLZ7.

soluções não dominadas para obter as sementes. Para o I-Multi Centroid, a busca pode ser limitada pelo algoritmo de agrupamento. Um agrupamento ruim pode gerar sementes iniciais ruins para cada enxame. Para o I-Multi Extremos, o número de enxames influencia a pesquisa, uma vez que, se o número de enxames é menor do que o número de funções objetivo, o algoritmo não funciona bem. O I-Multi Aleatório introduz uma abordagem mais simples, em que as sementes de cada enxame são selecionados aleatoriamente a partir do conjunto inicial de soluções. Os resultados são apresentados na Tabela 7.20 e nas Figuras 7.47 a 7.50.

Para DTLZ2, a configuração com 30 enxames obteve os melhores resultados em termos de convergência e diversidade. Ela obteve os melhores valores para GD, IGD e LD para quase todas os números de funções objetivo. Os resultados da medida *Convergence* foram semelhantes para quase todos os números de enxames. Isto mostra, para DTLZ2, que um pequeno número de soluções em cada *sub-swarm* não há necessariamente uma grande

perda na convergência em direção à Fronteira de Pareto. A Figura 7.47 mostra que a configuração com o maior número de enxames controlou melhor a degradação das medidas de qualidade. Em resumo, para DTLZ2, quanto maior o número de enxames, melhor é a busca. Além disso, o processo aleatório de escolha de sementes foi suficiente para o algoritmo geração de uma $PF_{aprox}$ diversificada.



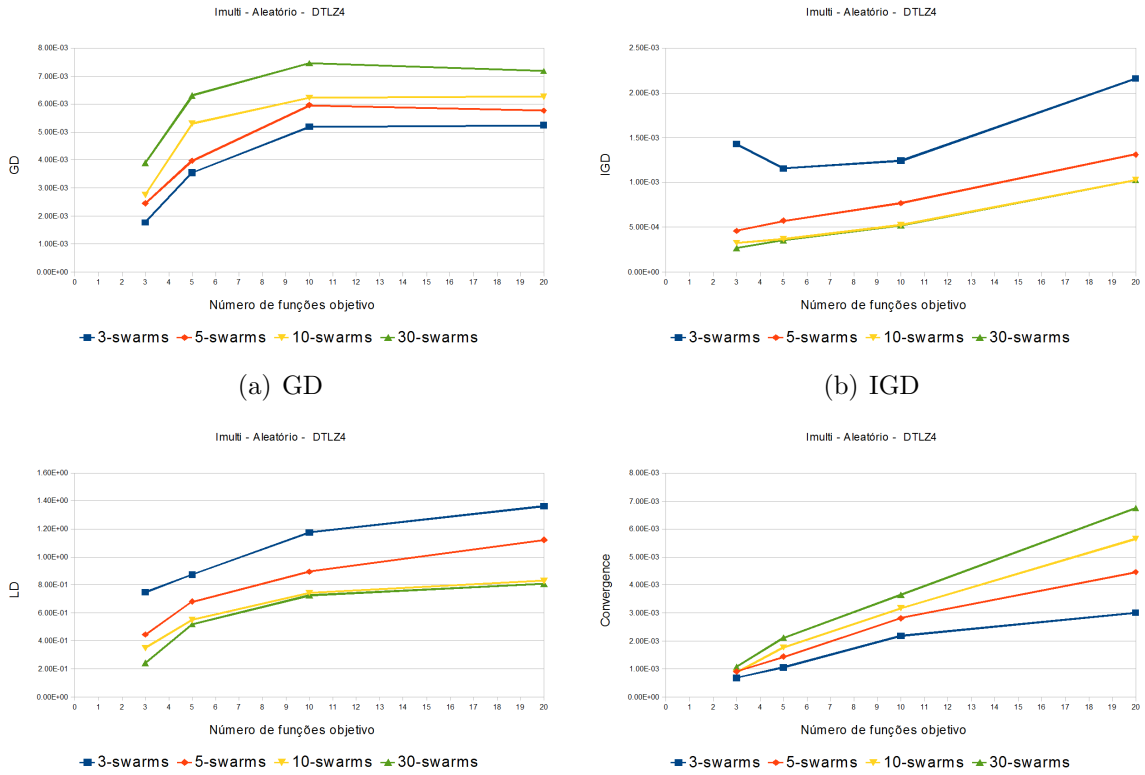

(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.47: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Aleatório, DTLZ2.

Para o DTLZ4 os resultados foram semelhantes às demais variantes do I-Multi. Mais uma vez, as configurações com menor número de enxames ficaram presas na região mais densa do espaço de objetivos. A configurações com 3 e 5 enxames obtiveram os melhores valores de GD e *Convergence*, mas os piores valores de IGD e LD. As configurações com 10 e 30 enxames apresentaram os melhores IGD e LD. Isso mostra que essas configurações evitaram as regiões mais densas e cobriram diferentes áreas do espaço de objetivos. No entanto, essas configurações perderam poder de convergência quando o número de objetivos cresceu, como observado na Figura 7.48. Em resumo, as variantes com maior número de

enxame foram melhores para DTLZ4, no entanto, recomenda-se uma execução com mais soluções em cada enxame para introduzir maior convergência na busca. Na Figura 7.48 pode-se ver o comportamento do I-Multi Aleatório para o DTLZ4. Os enxames menores controlam melhor a deterioração das medidas de convergência, enquanto que os maiores enxames tiveram uma menor deterioração das medidas de diversidade, quando o número de objetivos cresceu.



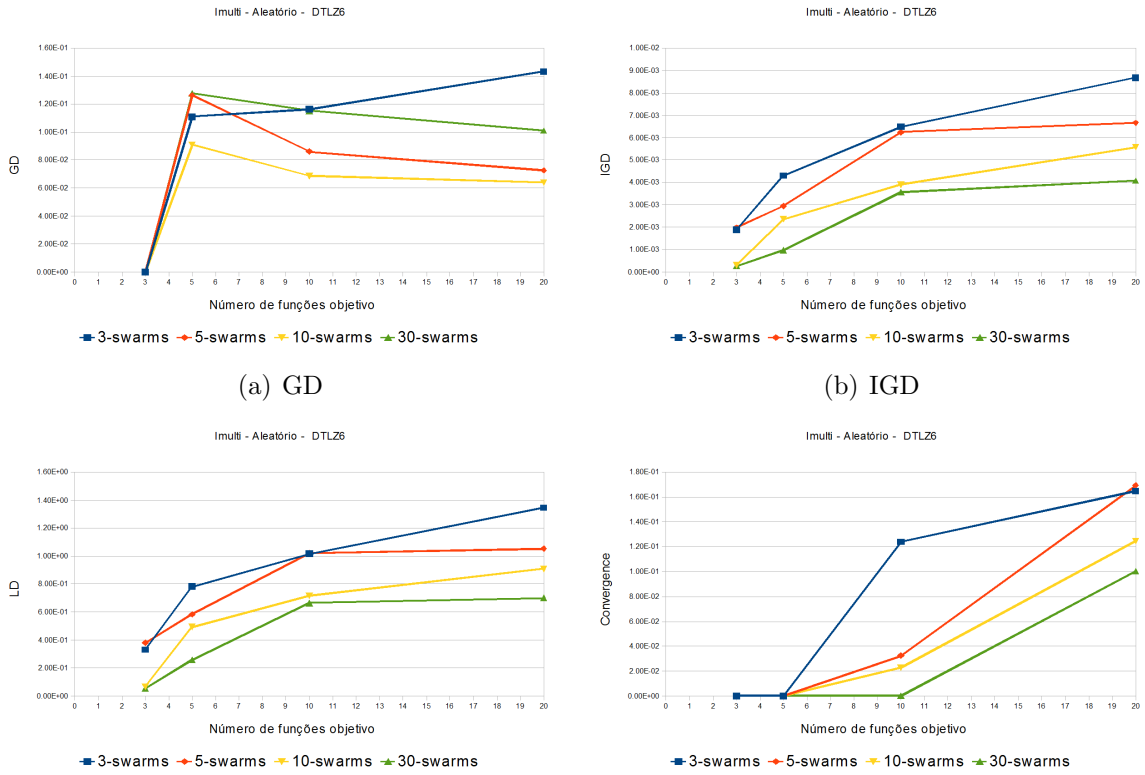

(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.48: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Aleatório, DTLZ4.

Para DTLZ6 as configurações com 10 e 30 enxames obtiveram os melhores valores para as medidas de diversidade (melhores valores do IGD e LD). Observando o GD, as configurações com 5 e 10 enxames obtiveram os melhores resultados, especialmente para os maiores número de funções objetivo. Mais uma vez, a configuração com 30 enxames obteve o melhor valor de *Convergence* para quase todas as funções objetivo. Assim como nos outros algoritmos I-Multi, essa configuração obteve boa diversidade, porém não evitou alguns ótimos locais em algumas regiões da fronteira (melhor Convergence, porém perde no

GD). Além disso, essa configuração obteve resultados muito bons para 3 funções objetivo. Em resumo, nenhuma das configurações obteve destaque. As configurações com 10 e 30 enxames apresentaram um bom resultado, mas recomenda-se aumentar o número de soluções em cada enxame para evitar os ótimos locais. Na Figura 7.49 , pode-se observar que o algoritmo com 30 enxame conseguiu controlar bem a diversidade da pesquisa (menor deterioração para IGD e LD) e também os bons resultados da configuração 10 enxames.




(a) GD

(b) IGD

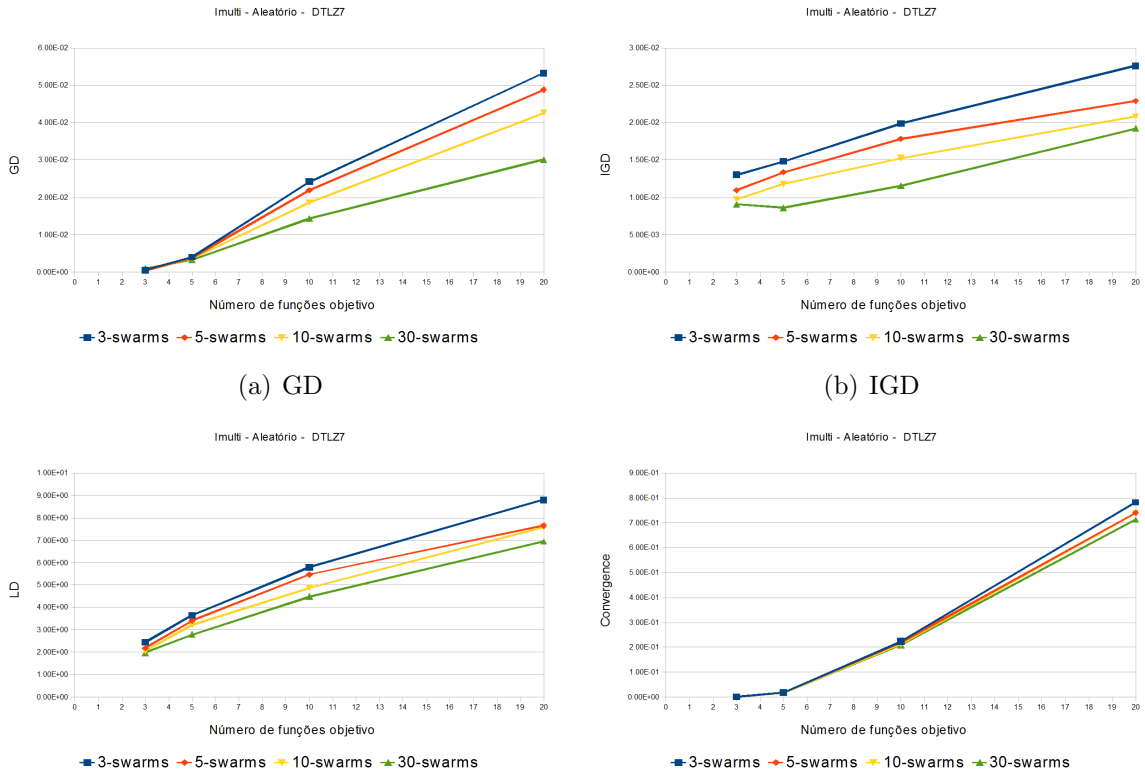
(c) LD

(d) *Convergence*

Figura 7.49: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Aleatório, DTLZ6.

Para o último problema, DTLZ7, a configuração com 30 enxames apresentou os melhores resultados. Da mesma forma que as outras variantes do I-Multi, essa configuração obteve os melhores resultados em termos de diversidade e convergência. Porém quando o número de fronteiras desconexas se torna grande, todas as variantes sofreram uma grande deterioração (Figura 7.50). O algoritmo com 30 enxames obteve os melhores valores de GD e IGD para quase todos os números de objetivos. Ele também conseguiu um bom valor para o LD. Em resumo, quanto maior o número de enxames, melhores são os resul-

tados para um problema com fronteiras desconexas. O maior número de enxames facilita a cobertura dos diferentes subfronteiras. Um importante aspecto do I-Multi Aleatório é que foi possível cobrir diferentes subfronteiras utilizando a seleção das sementes de forma aleatória. A Figura 7.50 mostra que com 30 enxames pode-se controlar melhor a deterioração dos indicadores quando o número de objetivo cresce. Porém também é possível observar que esses valores tornam-se semelhantes para um número alto de objetivos.




(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.50: Valores médios de GD, IGD, LD e *Convergence* para o I-Multi Aleatório, DTLZ7.

Fazendo uma análise conjunta de todos os diferentes problemas, pode-se concluir que para o algoritmo I-Multi Aleatório, quanto maior é o número de enxames, melhor é o desempenho. Em quase todos os diferentes cenários, as configurações com os maiores números de enxames obtiveram os melhores resultados. Além disso, a estratégia de seleção de sementes de forma aleatória foi suficiente para realizar a busca em diferentes áreas do espaço e obter bons valores para as medidas de diversidade.

Por fim, observando os resultados dos diferentes esquemas de escolha de semente, na

Tabela 7.20: I-Multi Aleatório

| | | DTLZ2 | | | | DTLZ4 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Swarms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **3** | 6,52E-04 | 5,40E-03 | 1,02E+00 | 2,60E-03 | 1,78E-03 | 1,43E-03 | 7,50E-01 | 6,83E-04 |
| | **5** | 7,68E-04 | 3,19E-03 | 7,33E-01 | 3,52E-03 | 2,45E-03 | 4,59E-04 | 4,45E-01 | 9,12E-04 |
| | **10** | 1,15E-03 | 6,35E-04 | 2,77E-01 | 6,42E-03 | 2,76E-03 | 3,24E-04 | 3,50E-01 | 8,81E-04 |
| | **30** | 5,95E-04 | 3,49E-04 | 1,24E-01 | 4,80E-03 | 3,90E-03 | 2,66E-04 | 2,41E-01 | 1,08E-03 |
| **5** | **3** | 4,15E-03 | 6,14E-03 | 1,07E+00 | 1,54E-02 | 3,55E-03 | 1,16E-03 | 8,75E-01 | 1,05E-03 |
| | **5** | 3,35E-03 | 4,12E-03 | 9,34E-01 | 1,53E-02 | 3,98E-03 | 5,71E-04 | 6,81E-01 | 1,44E-03 |
| | **10** | 2,35E-03 | 1,55E-03 | 5,84E-01 | 1,46E-02 | 5,31E-03 | 3,67E-04 | 5,49E-01 | 1,77E-03 |
| | **30** | 1,54E-03 | 9,76E-04 | 3,71E-01 | 1,66E-02 | 6,30E-03 | 3,54E-04 | 5,18E-01 | 2,12E-03 |
| **10** | **3** | 7,15E-03 | 6,02E-03 | 1,11E+00 | 1,82E-02 | 5,18E-03 | 1,24E-03 | 1,17E+00 | 2,18E-03 |
| | **5** | 5,95E-03 | 4,19E-03 | 9,74E-01 | 1,99E-02 | 5,95E-03 | 7,69E-04 | 8,95E-01 | 2,82E-03 |
| | **10** | 3,87E-03 | 2,20E-03 | 7,99E-01 | 2,02E-02 | 6,22E-03 | 5,26E-04 | 7,45E-01 | 3,17E-03 |
| | **30** | 2,68E-03 | 1,44E-03 | 6,55E-01 | 2,15E-02 | 7,46E-03 | 5,21E-04 | 7,27E-01 | 3,66E-03 |
| **20** | **3** | 6,64E-03 | 5,95E-03 | 1,14E+00 | 1,70E-02 | 5,24E-03 | 2,16E-03 | 1,36E+00 | 3,01E-03 |
| | **5** | 5,40E-03 | 3,57E-03 | 9,60E-01 | 1,65E-02 | 5,76E-03 | 1,31E-03 | 1,12E+00 | 4,46E-03 |
| | **10** | 4,10E-03 | 2,05E-03 | 7,63E-01 | 2,15E-02 | 6,26E-03 | 1,03E-03 | 8,29E-01 | 5,64E-03 |
| | **30** | 2,81E-03 | 1,46E-03 | 6,41E-01 | 1,94E-02 | 7,19E-03 | 1,03E-03 | 8,09E-01 | 6,75E-03 |

| | | DTLZ6 | | | | DTLZ7 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Swarms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **3** | 5,79E-06 | 1,89E-03 | 3,31E-01 | 1,32E-07 | 4,91E-04 | 1,30E-02 | 2,45E+00 | 2,90E-04 |
| | **5** | 6,34E-04 | 1,99E-03 | 3,79E-01 | 1,30E-07 | 4,05E-04 | 1,09E-02 | 2,18E+00 | 2,53E-04 |
| | **10** | 3,68E-06 | 3,16E-04 | 6,61E-02 | 5,10E-08 | 3,42E-04 | 9,76E-03 | 2,06E+00 | 2,42E-04 |
| | **30** | 2,77E-06 | 2,73E-04 | 5,34E-02 | 4,49E-08 | 9,69E-04 | 9,09E-03 | 1,98E+00 | 2,38E-04 |
| **5** | **3** | 1,11E-01 | 4,31E-03 | 7,83E-01 | 1,04E-05 | 3,97E-03 | 1,48E-02 | 3,65E+00 | 1,81E-02 |
| | **5** | 1,26E-01 | 2,94E-03 | 5,84E-01 | 1,07E-05 | 3,83E-03 | 1,34E-02 | 3,41E+00 | 1,92E-02 |
| | **10** | 9,09E-02 | 2,35E-03 | 4,94E-01 | 4,04E-06 | 3,67E-03 | 1,18E-02 | 3,22E+00 | 1,57E-02 |
| | **30** | 1,28E-01 | 9,73E-04 | 2,59E-01 | 1,25E-06 | 3,25E-03 | 8,59E-03 | 2,80E+00 | 1,59E-02 |
| **10** | **3** | 1,16E-01 | 6,49E-03 | 1,02E+00 | 1,24E-01 | 2,42E-02 | 1,99E-02 | 5,79E+00 | 2,24E-01 |
| | **5** | 8,61E-02 | 6,24E-03 | 1,02E+00 | 3,23E-02 | 2,19E-02 | 1,78E-02 | 5,47E+00 | 2,20E-01 |
| | **10** | 6,87E-02 | 3,89E-03 | 7,16E-01 | 2,30E-02 | 1,86E-02 | 1,53E-02 | 4,88E+00 | 2,13E-01 |
| | **30** | 1,15E-01 | 3,56E-03 | 6,65E-01 | 9,80E-05 | 1,43E-02 | 1,15E-02 | 4,49E+00 | 2,09E-01 |
| **20** | **3** | 1,43E-01 | 8,68E-03 | 1,35E+00 | 1,65E-01 | 5,33E-02 | 2,76E-02 | 8,80E+00 | 7,83E-01 |
| | **5** | 7,26E-02 | 6,66E-03 | 1,05E+00 | 1,00E+00 | 4,88E-02 | 2,29E-02 | 7,65E+00 | 7,40E-01 |
| | **10** | 6,39E-02 | 5,57E-03 | 9,08E-01 | 1,25E-01 | 4,26E-02 | 2,08E-02 | 7,58E+00 | 7,39E-01 |
| | **30** | 1,01E-01 | 4,08E-03 | 7,01E-01 | 1,00E-01 | 3,01E-02 | 1,92E-02 | 6,95E+00 | 7,13E-01 |

maioria dos cenários as configurações com maiores números de enxames se destacaram. Assim, nos próximas etapas desse conjunto de experimentos será adotado o número de 30 enxames para o algoritmo I-Multi. A próxima seção irá confrontar as três abordagens diferentes do I-Multi com o objetivo de observar qual delas apresentou o melhor desempenho em cada cenário diferente. Além disso, será analisado o objetivo principal do I-Multi: compor diferentes algoritmos de arquivamento através de múltiplos exames é melhor que utilizar os métodos separadamente em um algoritmo MOPSO. Com esse propósito, as diferentes abordagens do I-Multi são comparadas com o algoritmo SMPSO com o Arquivador Ideal e com o Arquivador MGA.

### 7.2.8 I-Multi versus métodos de arquivamento

Nesse experimento as três variantes de I-Multi são confrontadas entre si e com algoritmos MOPSO utilizando o Arquivador Ideal e o Arquivador MGA [9]. As seções anteriores discutiram a influência dos parâmetros do I-Multi em cenários com muitas funções objetivo. A partir desses experimentos os parâmetros selecionados foram: região de busca foi definida inicialmente em $0,5$ e reduzida proporcionalmente em cada iteração para $0,1$. O número de iterações de particionamento foi definido como 5. Para o número de enxames, em geral, quanto maior o número de enxames, melhor é o desempenho do algoritmo, para todas as variantes. Portanto, neste conjunto de experimentos o número de enxames para I-Multi foi definido como 30 enxames, para todos os problemas. A única exceção foi para o I-Multi Centroid , para o problema DTLZ4. Nesta situação são usados 5 enxames.

As três variantes do I-Multi são comparadas com o algoritmo SMPSO com o Arquivador Ideal (chamado aqui Ideal) e o SMPSO com o Arquivador MGA (chamado aqui MGA). Ambos os algoritmos foram executados com o mesmo conjunto de parâmetros, diferindo apenas pelo método de arquivamento. Os parâmetros base do SMPSO são apresentados na Tabela 7.1. A população foi limitada a 200 partículas, o tamanho $N$ do arquivo foi definido com 200 soluções. O método de escolha do líder é o mesmo utilizado pelo SMPSO, escolha do líder por distância de *Crowding*. Esses algoritmos foram executados com o mesmo número de avaliações de fitness realizado pelo I-Multi (obtida através das configurações descritas no parágrafo anterior): um total de 385850 avaliações de fitness. Todos os algoritmos foram executados 30 vezes independentes.

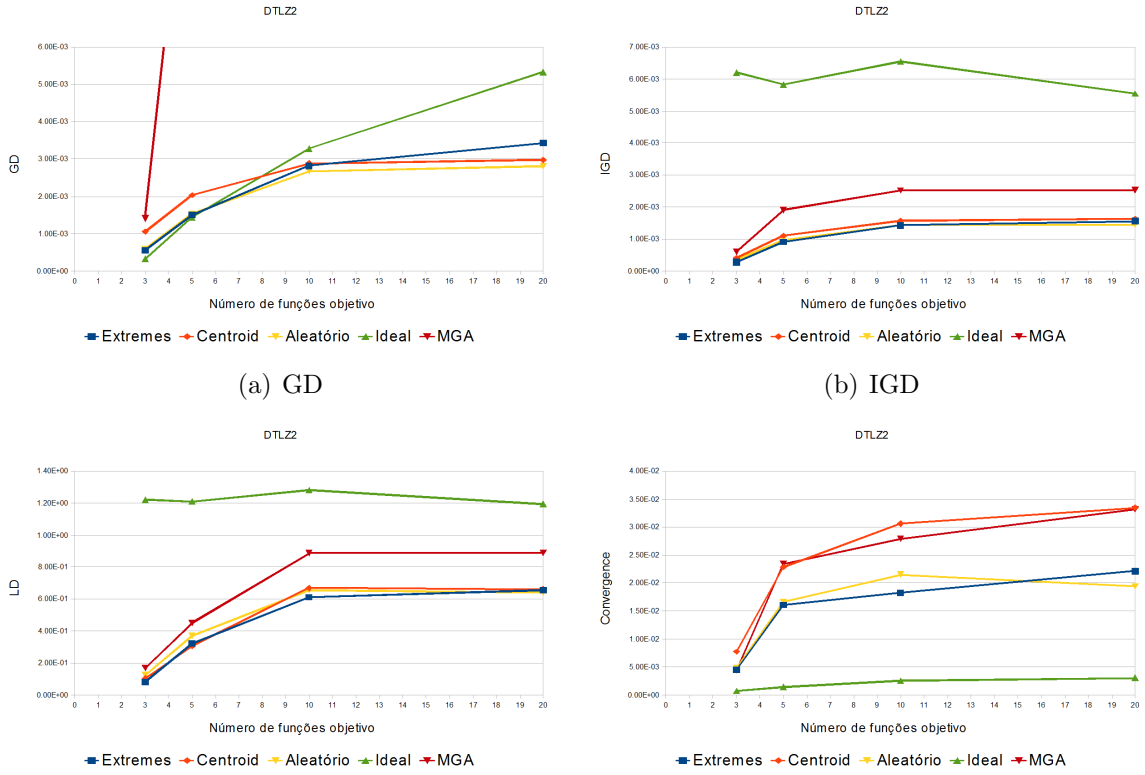
Esse conjunto de experimentos irá observar se o I-Multi supera os algoritmos MOPSO tradicionais e se a combinação de arquivadores produz uma melhora na busca, tanto em termos de diversidade, quanto de convergência. Além disso, será feita uma análise de qual variante do I-Multi tem o melhor desempenho em cada cenário. Os resultados são apresentados na Tabela 7.21 e nas Figuras 7.51 e 7.54.

O primeiro problema analisado é o DTLZ2. Observando GD, as diferentes variantes do I-Multi apresentaram um resultado muito semelhante ao Ideal. Ideal obteve o melhor resultado das medidas de convergência para quase todos os números de objetivos. Como

esse método favorece a convergência sobre a diversidade espera-se um bom valor de GD e *Convergence*. No entanto, pelo menos uma variante I-Multi obteve um valor equivalente de GD, para cada um dos números de objetivos. Além disso, o algoritmo I-Multi sofreu uma menor deterioração do indicador GD quando o número de objetivos cresceu (ver Figura 7.51(a)) e apresentou o melhor valor desse indicador para 20 funções objetivo. Os resultados do I-Multi Extremos e do I-Multi Aleatório podem ser destacados. O I-Multi Extremos apresentou resultados muito bons para 3 , 5 e 10 números de objetivos. No entanto, como discutido antes, ele perde o desempenho quando o número de objetivos se aproxima do número de enxames. Para a medida de *Convergence*, a Ideal conseguiu o melhor resultado para todos os números de objetivo. Algoritmo MGA obteve um valor pobre de GD e *Convergence* quando comparado aos demais. Analisando as medidas de diversidade, IGD e LD, novamente o I-Multi obteve um resultado muito bom. I-Multi Extremos e I-Multi Aleatório obtiveram os melhores valores para ambas as medidas, mas, novamente, I-Multi Extremos perdeu desempenho para um número elevado de funções objetivo. I-Multi Centroid também obteve um bom valor de LD, equivalente às outras variantes para quase todos os números de objetivos. Em geral, o I-Multi controlou bem a deterioração da diversidade (Figuras 7.52(b) e 7.51(c)). O MGA também obteve um bom valor de IGD e controlou bem a deterioração, mas foi superado pelo I-Multi.

Em resumo, para DTLZ2, o I-Multi obteve o melhor resultado, especialmente o I-Multi Aleatório. Combinando as diferentes abordagens de arquivamento foi possível melhorar os resultados do algoritmo MOPSO em termos de convergência e de diversidade. As variantes do I-Multi apresentaram resultados semelhantes aos do Ideal em termos de convergência e superaram o MGA em termos de diversidade. As três variantes obtiveram características de diversidade semelhantes, no entanto o I-Multi Centroid não obteve o mesmo poder de convergência que as outras variantes. A abordagem aleatória provou ser muito eficaz, uma vez que foi possível a obtenção de uma boa convergência e boa diversidade usando essa estratégia.

O segundo problema é o DTLZ4. Mais uma vez, o algoritmo I-Multi superou tanto o Ideal e o MGA. O Ideal obteve os melhores valores de GD e *Convergence* para quase todas



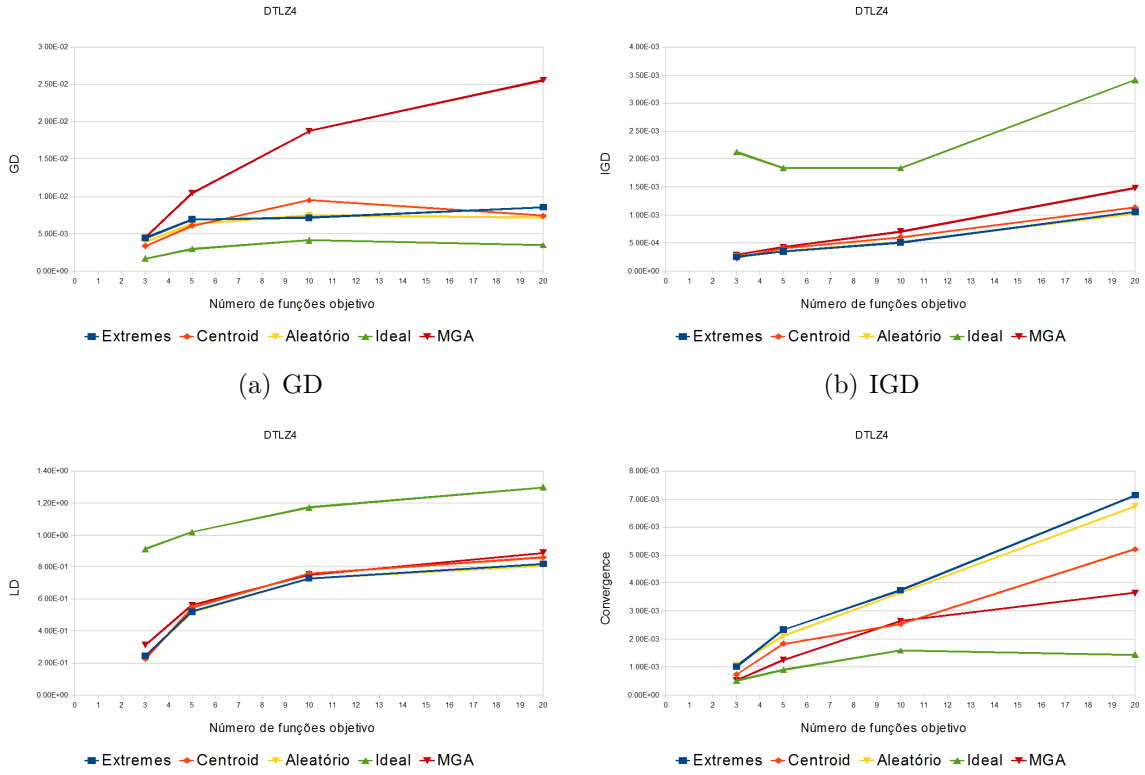

(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.51: Valores médios de GD, IGD, LD e *Convergence* para o problema DTLZ2, I-Multi x MGA x Ideal.

as funções objetivo, porém apresentou um valor baixo para as medidas de diversidade. Isso mostra que o algoritmo ficou preso na região mais densa da fronteira de Pareto. O I-Multi também obteve um bom valor de GD e *Convergence*, mas foi superado pelo Ideal em quase todos os números de objetivos. Tanto o Ideal quanto as diferentes variantes do I-Multi controlaram bem a deterioração das medidas de convergência (Figuras 7.52(a) e 7.52(d)). As três variantes de I-Multi apresentaram os melhores valores de IGD e LD e controlaram melhor a deterioração da diversidade (Figuras 7.52(b) e 7.52(c)). O I-Multi Aleatório obteve os melhores resultados de IGD e LD para quase todos os números de funções objetivo, especialmente para as maiores dimensões. I-Multi Extremos teve bons resultados, mas perdeu desempenho quando o número de objetivos cresceu. I-Multi Centroid teve um resultado semelhante ao I-Multi Aleatório, mas foi superado nas maiores dimensões. MGA teve valores de GD ruins e foi superado por I-Multi em termos de diversidade.

Em resumo, para DTLZ4, o I-Multi foi novamente mais eficaz. Ele obteve a melhor diversidade, o que representa que evitou melhor a região mais densa do DTLZ4. Ele também obteve bons valores de GD. I-Multi Aleatório foi o melhor algoritmo, seguido de perto pelo I-Multi Centroid.




(a) GD (b) IGD

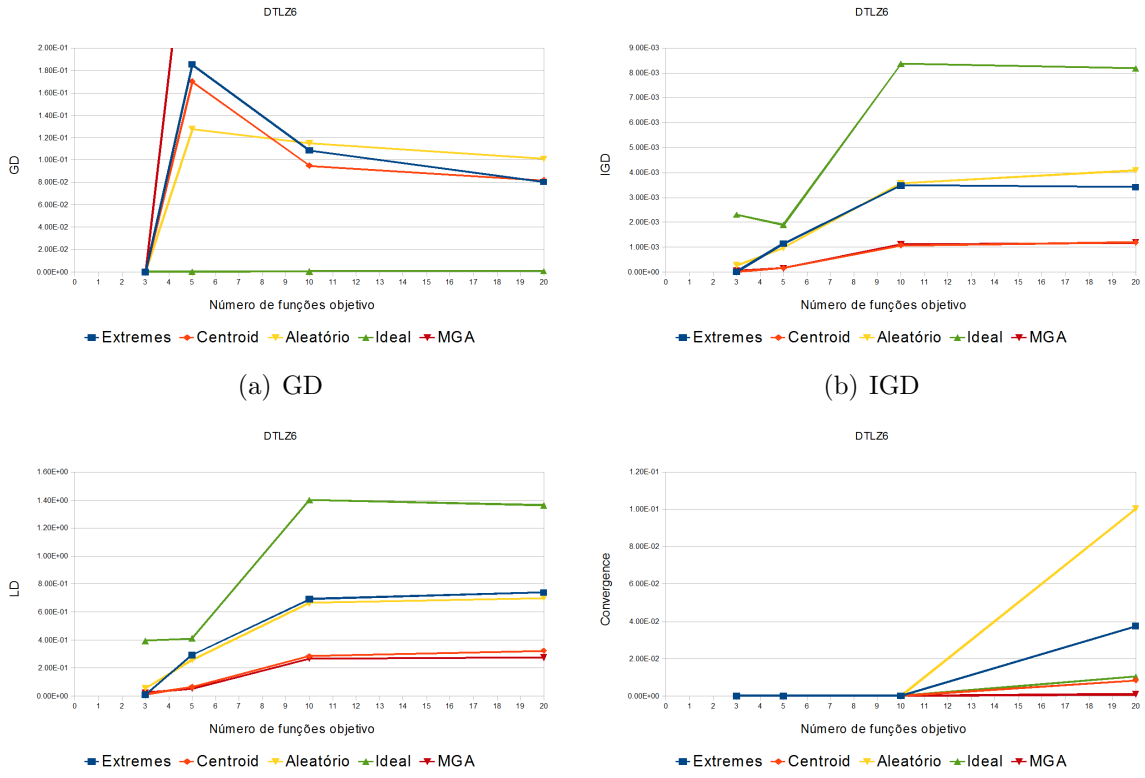
(c) LD (d) *Convergence*

Figura 7.52: Valores médios de GD, IGD, LD e *Convergence* para o problema DTLZ4, I-Multi x MGA x Ideal.

DTLZ6 apresentou as principais dificuldades ao I-Multi. O algoritmo foi capaz de obter boa diversidade, mas sofreu alguns problemas com múltiplos ótimos locais. Todas as três variantes não tiveram os melhores valores de GD e *Convergence*. Para o GD, Ideal obteve os melhores valores para quase todos os números de funções objetivo. Esse algoritmo também controlou melhor a deterioração no valor dos indicadores (Figura 7.53(a)). As três variantes de I-Multi obtiveram um valor semelhante de GD, pior do que o Ideal, mas melhor do que MGA. I-Multi Centroid e I-Multi Extremos também tiveram um bom valor para o indicador *Convergence* para 3, 5 e 10 funções, mas MGA obteve o melhor resultado para 20 objetivos. Observando as medidas de diversidade, IGD e LD, o I-Multi

apresentou melhores resultados para todos os números de objetivos. Nesse problema, o I-Multi Centroid obteve bons resultados em comparação com as outras variantes e controlou melhor a deterioração (Figuras 7.53(b) e 7.53(c)).

Em resumo, o algoritmo I-Multi obteve um resultado equivalente ao Ideal e ao MGA. Ele foi superado em termos de convergência pelo Ideal, mas obteve uma melhor diversidade. Também apresentou um resultado equivalente ao MGA em termos de diversidade, porém obteve melhor convergência. Entre as variantes do I-Multi, o I-Multi Centroid apresentou os melhores resultados e controlou melhor a diversidade e convergência. No entanto, o I-Multi enfrentou mais problemas no DTLZ6 e não apresentou os mesmos bons resultados obtidos no DTLZ2 e DTLZ4. Uma solução possível é aumentar o número de soluções em cada enxame. Fazendo isso, espera-se que o algoritmo evite mais ótimos locais.




(a) GD (b) IGD

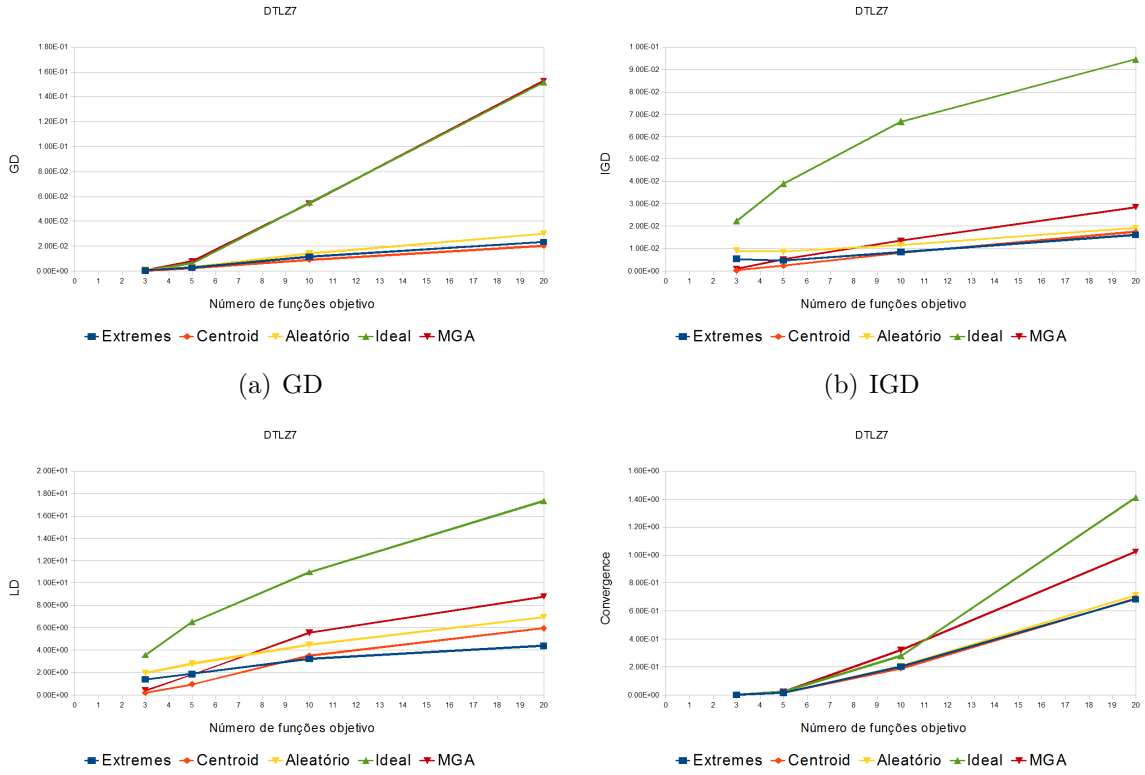
(c) LD (d) *Convergence*

Figura 7.53: Valores médios de GD, IGD, LD e *Convergence* para o problema DTLZ6, I-Multi x MGA x Ideal.

Finalmente, com o DTLZ7 podemos observar como cada algoritmo trabalha com uma

fronteira de Pareto desconexa. Aqui, o I-Multi superou tanto o Ideal quanto o MGA. Observando o GD e *Convergence*, o I-Multi obteve os melhores valores para quase todos os números de objetivos. Além disso, I-Multi Extremos e I-Multi Centroid superaram o I-Multi Aleatório. Nas Figuras 7.52(a) e 7.52(d) pode-se notar que o I-Multi sofreu uma menor deterioração que o Ideal e o MGA. Para as medidas de diversidade, de novo, o I-Multi apresentou melhores resultados que os outros algoritmos. Tanto o I-Multi Extremos quanto o I-Multi Centroid obtiveram os melhores valores de IGD e LD, para quase todas as funções objetivo. Isso representa que esses algoritmos abrangeram uma maior quantidade de subfronteiras desconexas, especialmente o I-Multi Centroid. No entanto, quando o número de objetivos aumenta, o número de subfronteiras desconexas torna-se muito alto. Uma vez que há várias subfronteiras a serem cobertas, o número de enxames é insuficiente para cobrir a fronteira de Pareto inteira e o I-Multi apresentou um resultado Equivalente ao MGA em termos de diversidade. Nas Figuras 7.53(b) e 7.53(c) pode-se observar que as três variantes de I-Multi apresentaram uma grande deterioração dos indicadores para as maiores dimensões do espaço de objetivos, especialmente em termos de LD (o que representa que a $PF_{aprox}$ está mais afastada de algumas regiões da $PF_{real}$). Em resumo, o I-Multi obteve um melhor desempenho do que o MGA e o Ideal para um problema com fronteira desconexa, tanto em termos de convergência e diversidade. No entanto esse algoritmo perde desempenho em dimensões muito grandes. I-Multi Centroid obteve os melhores resultados em termos de diversidade. Uma vez que as soluções estão desconectadas no espaço de objetivos, a estratégia de encontrar grupos mostrou-se ser mais eficiente. Uma possível maneira de melhorar os resultados do I-Multi Centroid no DTLZ7, é a implementação de um mecanismo que permite que o algoritmo explore grupos diferentes em cada iteração do algoritmo de particionamento. Além disso, aumentar o número de enxames e utilizar um melhor algoritmo de agrupamento pode melhorar os resultados.

Em resumo, usando uma estratégia cooperativa com múltiplos enxames buscando unir as qualidades do arquivador MGA do arquivador Ideal, mostrou ser uma boa maneira de lidar com a Otimização com Muitos Objetivos. Através da análise de diferentes cenários




(a) GD (b) IGD

(c) LD (d) *Convergence*

Figura 7.54: Valores médios de GD, IGD, LD e *Convergence* para o problema DTLZ7, I-Multi x MGA x Ideal.

com problemas com muitos objetivos, o algoritmo proposto, I-Multi, superou os algoritmos que usam apenas os arquivadores Ideal ou MGA. Combinando as propriedades de diversidade de MGA (para criar um conjunto de soluções inicial diversificado) com uma busca *multi-swarm* com o arquivador Ideal (com o objetivo de obter maior convergência) melhorou os resultados, tanto em termos de diversidade e convergência.

As três diferentes variantes I-Multi propostas obtiveram bons resultados e cada um se destacou em diferentes cenários. O I-Multi Aleatório foi mais eficaz no DTLZ2. Esse algoritmo também se destacou no DTLZ4, onde foi mais eficaz em evitar a região mais densa. Também para esse problema, o I-Multi Centroid obteve um bom resultado, semelhante ao I-Multi Aleatório. A abordagem I-Multi Centroid se destacou no problema DTLZ7. A estratégia de agrupamento das soluções na geração das sementes foi eficaz para a uma melhor cobertura da fronteira de Pareto desconexa. O I-Multi Extremos, em geral, apresentou bons resultados. Ele combina uma estratégia que utiliza algumas

Tabela 7.21: Resultados dos indicadores para a análise dos algoritmos com múltiplos enxames

| | | DTLZ2 | | | | DTLZ4 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Algorithms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **I-Extremes** | 5,53E-04 | 2,75E-04 | 8,15E-02 | 4,50E-03 | 4,39E-03 | 2,57E-04 | 2,45E-01 | 1,01E-03 |
| | **I-Centroid** | 1,05E-03 | 4,12E-04 | 1,04E-01 | 7,72E-03 | 3,33E-03 | 2,29E-04 | 2,26E-01 | 7,25E-04 |
| | **I-Random** | 5,95E-04 | 3,49E-04 | 1,24E-01 | 4,80E-03 | 3,90E-03 | 2,66E-04 | 2,41E-01 | 1,08E-03 |
| | **Ideal** | 3,25E-04 | 6,20E-03 | 1,22E+00 | 7,29E-04 | 1,70E-03 | 2,13E-03 | 9,12E-01 | 5,09E-04 |
| | **Mga** | 1,41E-03 | 5,99E-04 | 1,69E-01 | 4,37E-03 | 4,45E-03 | 2,92E-04 | 3,12E-01 | 5,21E-04 |
| **5** | **I-Extremes** | 1,50E-03 | 9,08E-04 | 3,23E-01 | 1,61E-02 | 6,93E-03 | 3,52E-04 | 5,21E-01 | 2,34E-03 |
| | **I-Centroid** | 2,03E-03 | 1,11E-03 | 3,05E-01 | 2,28E-02 | 7,29E-01 | 4,08E-04 | 5,47E-01 | 1,83E-03 |
| | **I-Random** | 1,54E-03 | 9,76E-04 | 3,71E-01 | 1,66E-02 | 6,30E-03 | 3,54E-04 | 5,18E-01 | 2,12E-03 |
| | **Ideal** | 1,44E-03 | 5,82E-03 | 1,21E+00 | 1,45E-03 | 2,93E-03 | 1,84E-03 | 1,02E+00 | 9,04E-04 |
| | **Mga** | 1,32E-02 | 1,90E-03 | 4,50E-01 | 2,34E-02 | 1,04E-02 | 4,28E-04 | 5,63E-01 | 1,25E-03 |
| **10** | **I-Extremes** | 2,82E-03 | 1,42E-03 | 6,10E-01 | 1,83E-02 | 7,14E-03 | 5,13E-04 | 7,29E-01 | 3,75E-03 |
| | **I-Centroid** | 2,88E-03 | 1,57E-03 | 6,70E-01 | 3,06E-02 | 9,47E-03 | 6,04E-04 | 7,60E-01 | 2,54E-03 |
| | **I-Random** | 2,68E-03 | 1,44E-03 | 6,55E-01 | 2,15E-02 | 7,46E-03 | 5,21E-04 | 7,27E-01 | 3,66E-03 |
| | **Ideal** | 3,28E-03 | 6,55E-03 | 1,28E+00 | 2,56E-03 | 4,12E-03 | 1,84E-03 | 1,17E+00 | 1,60E-03 |
| | **Mga** | 3,34E-02 | 2,51E-03 | 8,87E-01 | 2,79E-02 | 1,87E-02 | 7,07E-04 | 7,53E-01 | 2,65E-03 |
| **20** | **I-Extremes** | 3,42E-03 | 1,56E-03 | 6,55E-01 | 2,21E-02 | 8,53E-03 | 1,05E-03 | 8,21E-01 | 7,14E-03 |
| | **I-Centroid** | 2,97E-03 | 1,64E-03 | 6,61E-01 | 3,34E-02 | 7,39E-03 | 1,14E-03 | 8,59E-01 | 5,21E-03 |
| | **I-Random** | 2,81E-03 | 1,46E-03 | 6,41E-01 | 1,94E-02 | 7,19E-03 | 1,03E-03 | 8,09E-01 | 6,75E-03 |
| | **Ideal** | 5,33E-03 | 5,54E-03 | 1,19E+00 | 3,05E-03 | 3,49E-03 | 3,42E-03 | 1,30E+00 | 1,44E-03 |
| | **Mga** | 3,91E-02 | 2,53E-03 | 8,88E-01 | 3,32E-02 | 2,56E-02 | 1,48E-03 | 8,90E-01 | 3,64E-03 |

| | | DTLZ6 | | | | DTLZ7 | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Obj** | **Algorithms** | **GD** | **IGD** | **LD** | **Con** | **GD** | **IGD** | **LD** | **Con** |
| **3** | **I-Extremes** | 2,79E-06 | 1,47E-05 | 9,34E-03 | 3,11E-08 | 5,11E-04 | 5,38E-03 | 1,39E+00 | 1,96E-04 |
| | **I-Centroid** | 1,86E-06 | 1,72E-05 | 1,28E-02 | 2,06E-08 | 3,51E-04 | 2,39E-04 | 2,03E-01 | 1,33E-04 |
| | **I-Random** | 2,77E-06 | 2,73E-04 | 5,34E-02 | 4,49E-08 | 9,69E-04 | 9,09E-03 | 1,98E+00 | 2,38E-04 |
| | **Ideal** | 7,76E-06 | 2,30E-03 | 3,94E-01 | 4,49E-07 | 2,84E-04 | 2,24E-02 | 3,56E+00 | 2,54E-04 |
| | **Mga** | 7,62E-06 | 5,92E-05 | 2,52E-02 | 3,00E-07 | 6,18E-04 | 1,01E-03 | 4,33E-01 | 4,77E-04 |
| **5** | **I-Extremes** | 1,85E-01 | 1,14E-03 | 2,92E-01 | 1,50E-06 | 2,67E-03 | 4,69E-03 | 1,89E+00 | 1,65E-02 |
| | **I-Centroid** | 1,70E-01 | 1,62E-04 | 6,35E-02 | 2,63E-07 | 2,38E-03 | 2,31E-03 | 9,64E-01 | 1,33E-02 |
| | **I-Random** | 1,28E-01 | 9,73E-04 | 2,59E-01 | 1,25E-06 | 3,25E-03 | 8,59E-03 | 2,80E+00 | 1,59E-02 |
| | **Ideal** | 3,17E-04 | 1,89E-03 | 4,12E-01 | 5,85E-07 | 6,11E-03 | 3,89E-02 | 6,51E+00 | 2,32E-02 |
| | **Mga** | 3,48E-01 | 1,52E-04 | 5,32E-02 | 7,80E-07 | 7,73E-03 | 5,15E-03 | 1,82E+00 | 2,39E-02 |
| **10** | **I-Extremes** | 1,08E-01 | 3,48E-03 | 6,92E-01 | 2,95E-05 | 1,14E-02 | 8,52E-03 | 3,23E+00 | 2,04E-01 |
| | **I-Centroid** | 9,50E-02 | 1,07E-03 | 2,86E-01 | 3,97E-06 | 9,00E-03 | 8,10E-03 | 3,52E+00 | 1,90E-01 |
| | **I-Random** | 1,15E-01 | 3,56E-03 | 6,65E-01 | 9,80E-05 | 1,43E-02 | 1,15E-02 | 4,49E+00 | 2,09E-01 |
| | **Ideal** | 4,90E-04 | 8,37E-03 | 1,40E+00 | 7,50E-05 | 5,48E-02 | 6,67E-02 | 1,10E+01 | 2,80E-01 |
| | **Mga** | 3,99E-01 | 1,11E-03 | 2,67E-01 | 1,45E-05 | 5,42E-02 | 1,35E-02 | 5,55E+00 | 3,21E-01 |
| **20** | **I-Extremes** | 8,03E-02 | 3,42E-03 | 7,40E-01 | 3,74E-02 | 2,33E-02 | 1,62E-02 | 4,40E+00 | 6,84E-01 |
| | **I-Centroid** | 8,18E-02 | 1,19E-03 | 3,23E-01 | 8,50E-03 | 2,02E-02 | 1,75E-02 | 5,95E+00 | 6,91E-01 |
| | **I-Random** | 1,01E-01 | 4,08E-03 | 7,01E-01 | 1,00E-01 | 3,01E-02 | 1,92E-02 | 6,95E+00 | 7,13E-01 |
| | **Ideal** | 7,83E-04 | 8,19E-03 | 1,36E+00 | 1,03E-02 | 1,52E-01 | 9,45E-02 | 1,73E+01 | 1,41E+00 |
| | **Mga** | 4,02E-01 | 1,18E-03 | 2,75E-01 | 9,71E-04 | 1,53E-01 | 2,85E-02 | 8,78E+00 | 1,02E+00 |

informações sobre as soluções anteriores (guiando alguns enxames para os extremos) com a abordagem aleatória. Essa fusão apresentou uma abordagem robusta que gerou bons resultados para problemas diferentes, como DTLZ2 e DTLZ7. Contudo, essa abordagem sofre uma deterioração grande quando o número de objetivos se aproxima do número de enxames, uma vez que apenas os extremos são tratados nesta situação. A principal dificuldade do I-Multi foi enfrentada no DTLZ6. Todas as estratégias foram incapazes de evitar os ótimos locais e geraram uma $PF_{aprox}$ com baixa convergência em direção à $PF_{real}$.

## 7.3 Discussão dos resultados

Esta seção apresenta um resumo das análises empíricas apresentadas neste capítulo. Essa discussão aborda duas diferentes visões dos resultados, a primeira destacando quais foram os melhores métodos para cada conjunto de experimentos executados e a segunda discutindo quais foram os melhores algoritmos para cada problema separadamente. A seguir serão apresentadas as conclusões de cada conjunto de experimentos.

O primeiro conjunto de experimentos avaliou as técnicas baseadas em novas relações de preferência. Nesse contexto, o algoritmo CDAS-MOPSO apresentou os melhores resultados. Esse algoritmo consegue obter bons resultados em termos de convergência e diversidade mesmo quando utilizado em problemas com grandes dimensões no espaço de objetivos. Uma limitação do CDAS-MOPSO refere-se à escolha do parâmetro $S_i$. Nem sempre o mesmo valor do parâmetro consegue obter os melhores resultados em termos de convergência e diversidade, sendo necessário definir o valor do parâmetro dependendo do contexto que o algoritmo será utilizado. Além disso, apesar dessa técnica reduzir a deterioração da busca também em termos de diversidade, o CDAS-MOPSO privilegia convergência. As técnicas de ranking propostas conseguiram reduzir a deterioração quando utilizadas em um algoritmo MOPSO, mas seus resultados foram superados pelo CDAS-MOPSO.

Os métodos de arquivamento foram propostos com o objetivo de melhorar o resultado de algoritmos MOPSO explorando uma característica importante desses algoritmos. Dentre os algoritmos propostos, o Arquivador Hiperplano e o Arquivador Ideal se destacaram. Esses métodos mostram um grande poder de convergência, inclusive quando aplicados em problemas com muitos objetivos. O Arquivador Distribuído conseguiu bons resultados em termos de diversidade, especialmente em um problema com uma fronteira desconexa. Além disso, foi mostrado que o Arquivador Hiperplano guia a busca do algoritmo MOPSO para uma região próxima do ponto de referência escolhido. Por fim, um algoritmo MOPSO utilizando o Arquivador Ideal e diferentes métodos de escolha de líder se mostrou competitivo com o CDAS-MOPSO, com a vantagem que não é necessária a escolha de uma parâmetro extra. Porém, da mesma forma que na análise do CDAS-

MOPSO, tanto arquivadores proposto nesta tese, quanto os arquivadores da literatura, privilegiaram a busca em termos de convergência ou em termos de diversidade. Não foi possível a definição de um método que consiga melhorar os resultados de um algoritmo MOPSO em ambos os aspectos.

Visando produzir um algoritmo MOPSO que consiga bons resultados tanto em termos de convergência quanto em diversidade, esta tese explorou uma abordagem com múltiplos enxames. Assim, foi proposto um algoritmo, chamado de I-Multi, que compõe diferentes métodos de arquivamento através de um algoritmo *multi-swarm*. Nesse algoritmo foram utilizados os métodos de arquivamento Ideal e MGA. A análise empírica apresentada neste capítulo, configurou os principais parâmetros do algoritmo. Nessa configuração, uma conclusão importante é que em geral, quanto maior é o número de exames, melhor é o resultado do algoritmo. Além da configuração dos parâmetros, diferentes variantes do I-Multi foram confrontadas com um algoritmo MOPSO usando o Arquivador Ideal e com o Arquivador MGA. Os resultados mostraram que o algoritmo com múltiplos enxames que combina os métodos de arquivamento supera os algoritmos que utilizam os métodos de arquivamento separadamente. O I-Multi conseguiu melhorar os resultados dos arquivadores tanto em termos de convergência e diversidade para a maioria dos cenários com muitos objetivo.

A segunda análise refere-se a cada problema separadamente. Como discutido na Seção 7.1.2 os problemas de *benchmark* da família DTLZ utilizados possuem diferente características. Nas seções anteriores foi discutido como cada algoritmo se comportou em cada cenário. Essa discussão irá apresentar um conclusão geral sobre cada problema DTLZ.

O primeiro problema utilizado foi o DTLZ2. Esse problema é utilizado para medir a escalabilidade dos algoritmos quando o número de funções objetivo cresce. Para esse problema, os algoritmos que privilegiam a convergência se destacaram. Os algoritmos CDAS-fronteira e os métodos de arquivamento Ideal e Hiperplano obtiveram bons resultados em termos de convergência no problema DTLZ2 mesmo com muitas funções objetivo. O algoritmo que apresentou os melhores resultados nesse problema foi o I-Multi que

superou o arquivador Ideal em termos de convergência e o arquivador MGA em termos de diversidade. Em resumo, métodos que escolheram uma região do espaço de objetivos para cobrir conseguiram bons resultados no DTLZ2. Além disso, o uso de múltiplos enxames possibilitou uma melhora tanto tem termos de convergência, quanto de diversidade.

O problema DTLZ4 introduziu as maiores dificuldades para os algoritmos propostos nessa tese. Esse problema possui um região do espaço de objetivos mais povoada. Essa característica do problema induz a busca dos algoritmos a encontrarem somente pontos na região povoada e não cobrirem toda a fronteira de Pareto. Para esse problema, o CDAS-fronteira não obteve bom resultado, pois o uso da técnica CDAS guiou os o algoritmo para a região mais povoada da fronteira de Pareto. Da mesma forma que no CDAS-fronteira, o arquivador Ideal também ficou preso na região mais povoada, porém, o arquivador Hiperplano obteve bons resultados em termos diversidade quando escolheu pontos de referências nas regiões menos povoadas. O algoritmo I-Multi obteve os melhores resultados em termos de diversidade e o uso de múltiplos enxames se mostrou mais indicado para evitar as regiões mas densas do espaço de objetivos. Em resumo, é possível obter bons resultados para o para um problema com uma região densa guiando a busca para regiões específicas do espaço de objetivos. Porém se a região escolhida for a região mais povoada o algoritmo tende a concentrar em poucos pontos e perde muita diversidade.

O problema DTLZ6 possui múltiplos ótimos locais. Além disso, sua fronteira de Pareto é definida por somente uma curva e não toda a superfícies da esfera. Nessa caso, os principais algoritmos propostos nesse trabalho obtiveram bons resultados. Tanto o CDAS-SMPSO quanto os métodos de arquivamento conseguiram bons resultados em termos de convergência, ou seja, conseguiram evitar os ótimos locais. Além disso, pela simplicidade do formato da fronteira de Pareto esses métodos obtiveram bons valores em termos de diversidade. O algoritmo I-Multi teve maiores dificuldades nesse problema. A estratégia de utiliza diferentes múltiplos enxames foi útil para cobrir uma maior parte da fronteira de Pareto, porém, o algoritmo não obteve um resultado tão bom em termos de convergência e encontrou dificuldades em evitar alguns ótimos locais. Para um problema mais simples com múltiplos ótimos locais, a estratégia de escolher regiões do espaço de objetivos e guia

o enxames para essas regiões se mostrou a mais adequada.

Por fim, o problema DTLZ7 possui um fronteira de Pareto desconexa. O algoritmo CDAS-SMPSO não obteve bons resultados. A estratégia de limitar a região de busca não funcionou nesse cenário e o algoritmo obteve resultados piores que o algoritmo SMPSO. Os métodos de arquivamento proposto conseguiram bons resultados em termos de convergência, porém cobriram poucas fronteiras desconexas. O algoritmo I-Multi obteve os melhores resultados. A estratégia de utilizar múltiplos enxames se mostrou útil em cobrir várias subfronteiras, além disso o algoritmo consegui bons resultados em termos de convergência. Assim, para um problema com fronteiras desconexas a melhor abordagem é espalhar a busca por diferentes regiões do espaço de objetivos utilizando múltiplos enxames.

## 7.4 Considerações finais

Esta tese buscou propor novos métodos especialmente projetados para a metaheurística da Otimização por Nuvem de Partículas Multiobjetivo aplicada a Problemas com Muitos Objetivos. As técnicas propostas exploram diferentes aspectos da busca de algoritmos MOPSO e buscaram enfatizar características importantes desses algoritmos. Este capítulo teve como objetivo validar todas as técnicas propostas através de diferentes cenários com problemas com muitos objetivos. Os experimentos utilizaram uma metodologia que combinou diferentes Problemas de Otimização com Muitos Objetivos através de análises empíricas dos resultados de um conjunto de indicadores de qualidade. Nesta análise, buscou-se identificar o comportamento das técnicas em termos de convergência e diversidade variando-se o número de objetivos dos problemas.

Fazendo uma análise conjunta de todos os experimentos, conclui-se que as técnicas especialmente projetadas para a Otimização com Muitos Objetivos melhoram os resultados da metaheurística MOPSO neste contexto. Em geral, é mais fácil conseguir um bom resultado em termos de convergência, porém é possível combinar diferentes técnicas com o objetivo de também se obter uma boa diversidade.

# CAPÍTULO 8

# CONCLUSÕES

Esta tese de doutorado explorou novas estratégias para a Otimização por Nuvem de Partículas, aplicadas na Otimização com Muitos Objetivos. Existem na literatura diversos algoritmos que trabalham com problemas multiobjetivo, porém esses algoritmos enfrentam diversos problemas quando esse número de objetivos cresce. Nosso trabalho estudou os problemas enfrentados na Otimização com Muitos Objetivos.

Para o desenvolvimento dessas novas propostas, inicialmente este trabalho de doutorado executou uma pesquisa bibliográfica através da leitura, discussão e implementação de diversos algoritmos publicados recentemente nas principais revistas e conferências da área. Esses trabalhos relacionados abordaram temas como algoritmos evolucionários multiobjetivo, a otimização por nuvem de partículas e a otimização multi/muitos objetivos.

Através de um amplo estudo da área três objetivos foram traçados: aplicar métodos da Otimização com Muitos Objetivo em algoritmos MOPSO; desenvolver novos algoritmos e métodos baseados na metaheurística MOPSO; efetuar uma análise empírica para validar todos os algoritmos desenvolvidos. A partir desses objetivos, esta tese desenvolveu um conjunto de métodos e algoritmos que buscaram explorar a criação de novas relações de preferências, métodos de arquivamento e múltiplos enxames.

Dentre a exploração das novas relações de preferência foram propostos os algoritmos CDAS-MOPSO [16] [7] e Ranking-SMPSO [13] [25]. Além disso, dois novos métodos foram propostos, chamados de *Balanced Ranking* e Combinação de Ranking [8]. Outro caminho explorou métodos de arquivamento de soluções não dominadas. Nessa etapa foram desenvolvidos quatro novos métodos de arquivamento Arquivador Ideal, Arquivador Distribuído, Arquivador Distribuído pela Busca [9] [7] e Arquivador Hiperplano. Por fim, a última etapa desta tese explorou algoritmos MOPSO com múltiplos enxames, o que gerou a proposta do algoritmo *Iterated Multi-Swarm*.

Para validar todos os métodos propostos e para um melhor entendimento da dificuldade em se lidar com problemas com muitos objetivos foi realizado um conjunto de estudos empíricos, tendo ênfase em se observar aspectos como a convergência em direção da fronteira de Pareto e diversidade das soluções na busca dos algoritmos. Assim, este estudo utilizou um conjunto de indicadores de qualidade e problemas de *benchmarking* com muitos objetivos, que variaram entre 5 e 20 objetivos.

A conclusão com maior ênfase nesta tese é que é possível reduzir a deterioração de MOEAs em MaOPs através de técnicas específicas para a Otimização com Muitos Objetivos. Além disso, a metaheurística MOPSO possui importantes características que podem ser exploradas com sucesso para reduzir essa deterioração.

Os algoritmos MOPSO que usam métodos da literatura conseguiram melhorar o desempenho do MOPSO para Problemas com Muitos Objetivos, em especial o algoritmo CDAS-MOPSO. Dentre os algoritmos e métodos desenvolvidos se destacam os Arquivadores Ideal e Hiperplano. Esses métodos obtiveram bons resultados em termos de convergência e diversidade em diferentes cenários com muitos objetivos. Eles foram comparados com diversos métodos de arquivamento da literatura e conseguiram obter melhores resultados em diferente situações. O que se destaca nos métodos é a habilidade em convergir para a fronteira de Pareto. Além disso, os resultados obtidos por um algoritmo MOPSO que utiliza essas técnicas foram melhores do que os resultados do algoritmo CDAS-MOPSO.

O algoritmo baseado em múltiplos enxames obteve os melhores resultados. A idea de compor diferentes métodos de arquivamento através de múltiplos enxames se mostrou mais eficiente do que usar um algoritmo MOPSO com os métodos separadamente. O algoritmo I-Multi conseguiu bons resultados tanto em termos de convergência quanto em diversidade em diferentes cenários com muitos objetivos.

## 8.1 Limitações

Apesar dos bons resultados obtidos pelos algoritmos desenvolvidos, este estudo encontrou algumas limitações. A primeira limitação é referente a questão da convergência versus

diversidade. Em geral, muitos algoritmos multiobjetivo enfrentam dificuldades em prover tanto convergência quanto diversidade na busca. Essa questão se agrava em Problemas com Muitos Objetivos, devido as dificuldades desses problemas. Nesse contexto, a maioria dos métodos desenvolvidos e também dos métodos da literatura privilegiou um em compensação ao outro. Os métodos como os arquivadores Ideal e Hiperplano, obtiveram resultados muito bons em termos de convergência (inclusive em problemas com múltiplos ótimos locais), porém encontram dificuldade em diversificar a busca. O algoritmo com múltiplos enxames foi especialmente desenvolvido com o intuito de compor métodos de arquivamento para obter bons resultados tanto em convergência como em diversidade e houve uma melhora significativa nessa questão. Porém, ainda há espaço na pesquisa para produzir um novo método que consiga introduzir tanto convergência quando diversidade na Otimização com Muitos Objetivos, em especial em espaços de objetivos com um alto número de funções.

Outra limitação é relativa aos problemas utilizados para validar os algoritmos. Nenhuma técnica foi utilizada num problema real. Apesar de os problemas DTLZ serem um importante parâmetro para a medição da qualidade dos algoritmos, é importante o uso dos algoritmos desenvolvidos em problemas reais para testar a escalabilidade dos métodos propostos.

Uma limitação importante na validação dos métodos propostos é enfrentada na escolha dos indicadores de qualidade. Como discutido no capítulo anterior, existem diversas métricas de qualidade na literatura. Porém, como não há um consenso sobre as melhores medidas que devem ser utilizadas os estudos desenvolvidos ficam limitados às interpretações das medidas utilizadas. Por exemplo, muitas vezes, medidas tais como o *spacing* e o GD privilegiam conjuntos de aproximação pequenos. Da mesma forma que a medida Hipervolume [84] pode ser mal interpretada dependendo da localização do conjunto de aproximação e da forma da fronteira de Pareto. Para atenuar essa limitação, esta tese utilizou um conjunto de indicadores para reduzir os erros na interpretação desses resultados.

## 8.2 Trabalhos futuros

Vários trabalhos futuros podem ser desenvolvidos através de extensões dos métodos desenvolvidos neste capítulo. São delineados alguns desafios futuros. Nos experimentos, foi mostrado que as técnicas propostas conseguem obter bons resultados em Problemas de Otimização com Muitos Objetivos. Porém, ainda existem algumas limitações, como a dificuldade em se trabalhar com fronteiras desconexas e a falta de diversidade em alguns problemas.

Motivado por essas dificuldades, pela busca de resultados melhores e pelo campo de pesquisa ainda em aberto são propostos alguns trabalhos futuros. Baseado nos experimentos, observa-se que dois fatores são importantes para se obter bons resultados com muitos objetivos: reduzir o número de soluções não-dominadas que guiam a população para a fronteira de Pareto e fazer uma boa escolha do líder, que pode introduzir mais convergência e mais diversidade na busca. A partir desses fatores alguns caminhos são:

**Proposta de novos métodos para a escolha do líder**: Um caminho a ser seguido é propor novas métodos para a escolha dos líderes que produzam comportamentos no enxame que possibilitem bons resultados em problemas com muitos objetivos. Uma estrtégia é guiar as partículas para os líderes mais próximos ao joelho da fronteira de Pareto. Assim, as soluções no extremos são descartadas e menos soluções não-dominadas são definidas. Para isto, dois métodos podem ser desenvolvidos:

a) Um método em que os líderes são as soluções não-dominadas mais próximas à solução ideal. Em cada iteração, é obtida a solução ideal, com os melhores valores dos objetivo encontrados até então. A partir da definição da solução ideal, somente os líderes próximos a esta solução são considerados;

b)Um método em que cada solução selecione o seu líder com o objetivo de se aproximar do joelho da fronteira de Pareto. Nesse método, uma partícula escolhe o seu líder com intuito de se movimentar para próximo do centro do espaço de objetivos. Assim, uma solução em um extremo escolhe um líder no outro extremo. Uma solução mais próxima ao joelho, busca um líder também próximo ao joelho.

Esses novos métodos de escolha do líder são propostos não somente para serem aplica-

dos em problemas com muitos objetivos, mas para melhorar os resultados dos algoritmos MOPSO em todos os propósitos. Além disso, eles podem ser aplicados juntamente com as técnicas para muitos objetivos discutidas em nosso trabalho.

**Proposta de métodos para a redução do número de soluções não-dominadas**: Como discutido anteriormente, reduzir o número de soluções não-dominadas gera bons resultados para problemas com muitos objetivos. Com muitas soluções não-dominadas é mais difícil ter pressão em direção à fronteira de Pareto e é mais complicado se aproximar desta fronteira. Podem ser obtidos novos métodos e novas relações de preferência que guiem a busca para boas regiões da fronteira de Pareto, partindo do princípio de tornar dominadas soluções que não são interessantes, como por exemplo, soluções nos extremos da fronteira de Pareto.

**Novas abordagens com múltiplos enxames para problemas com muitos objetivos**: As abordagens com múltiplos enxames obtiveram os melhores resultados nesta tese. Porém, existem diversos tópicos que podem ser explorados. Primeiro, novos algoritmos podem ser propostos. Esses algoritmos podem compor os métodos de arquivamento de diferentes formas, como por exemplo, se comunicar através de diferentes topologias ou utilizar um arquivo externo compartilhado. Além disso, o algoritmo I-Multi possibilita diversas extensões, como o reinício de *sub-swarms* que apresentem um desempenho pior que os demais ou o desenvolvimento de novos métodos para as escolhas das sementes.

***Multi-swarms* hibridizados com abordagens de decomposição**: Dentre os MOEAs, o MOEA/D se destaca pois consegue obter melhores resultados que os algoritmos estado-da-arte para problemas com muitos objetivos. Porém, apesar de obter bons resultados, esse algoritmo ainda sofre problemas quando aplicado a MaOPs. Assim, a ideia é construir uma PSO *multi-swarms* baseado em decomposição em que a troca de informações de cada população possibilite a otimização de problemas com muitos objetivos.

**Melhoramento dos métodos ranking propostos**: Os métodos de ranking propostos apresentaram bons resultados iniciais. Porém eles ainda não foram comparados com diferentes técnicas para muitos objetivos. Além disso, pode ser feito também um amplo estudo comparando os métodos propostos com os rankings apresentados em [40].

Dentre esses métodos, o *Global Detriment* apresenta uma ideia semelhante ao *Balanced* Ranking, porém só leva em consideração a diferença máxima entre os objetivo e não leva em consideração se a solução tem bons valores de objetivos.

**Aplicação dos algoritmos desenvolvidos em problemas da Engenharia de Software** : A Engenharia de Software Baseada em Busca [15] utiliza algoritmos de busca em problemas da Engenharia de Software. Dentre esses, é comum a existência problemas que possibilitem a existência de mais de três funções objetivo. Esse trabalho futuro tem como objetivo testar os métodos propostos num situação real e prover um melhor resultado para os problemas da Engenharia de Software.

# BIBLIOGRAFIA

[1] ADRA, S.; FLEMING, P. Diversity management in evolutionary many-objective optimization. **IEEE Transactions on Evolutionary Computation**, vol. 15, n. 2, p. 183–195, 2011.

[2] AGUIRRE, H.; TANAKA, K. Space partitioning evolutionary many-objective optimization: Performance analysis on MNK-landscapes. **Transactions of the Japanese Society for Artificial Intelligence**, vol. 25, n. 2, p. 363–376, 2010.

[3] ALVAREZ-BENITEZ, J.; EVERSON, R.; FIELDSEND, J. A MOPSO algorithm based exclusively on pareto dominance concepts. COELLO COELLO, C.; AGUIRRE, A.; ZITZLER, E., editores, **Evolutionary Multi-Criterion Optimization**, volume 3410 da Lecture Notes in Computer Science, p. 459–473. Springer Berlin / Heidelberg, Guanajuato, MX, 2005.

[4] BENTLEY, P. J.; WAKEFIELD, J. P. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. CHAWDHRY, P. K.; ROY, R.; PANT, R. K., editores, **Soft Computing in Engineering Design and Manufacturing**, p. 231–240. Springer-Verlag, Londres, GB, 1998.

[5] BRANKE, J.; MOSTAGHIM, S. About selecting the personal best in multi-objective particle swarm optimization. RUNARSSON, T.; BEYER, H.-G.; BURKE, E.; MERELO-GUERVóS, J. J.; WHITLEY, L.; YAO, X., editores, **Parallel Problem Solving from Nature - PPSN IX**, volume 4193 da Lecture Notes in Computer Science, p. 523–532. Springer Berlin / Heidelberg, Reykjavik, IS, 2006.

[6] BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. **IEEE Swarm Intelligence Symposium (SIS 2007)**. IEEE Press, Honolulu, US, 2007. p. 120–127.

[7] BRITTO, A.; POZO, A. I-MOPSO: A suitable PSO algorithm for many-objective optimization. **2012 Eleventh Brazilian Symposium on Neural Networks**. IEEE Computer Society, Curitiba, BR, 2012. p. 166–171.

[8] BRITTO, A.; POZO, A. Novos métodos baseados em ranking para problemas com muitos objetivos aplicados à otimização por nuvem de partículas. **X Congresso Brasileiro de Inteligência Computacional**. Fortaleza, BR, 2011.

[9] BRITTO, A.; POZO, A. Using archiving methods to control convergence and diversity for many-objective problems in particle swarm optimization. **2012 IEEE Congress on Evolutionary Computation (CEC 2012)**. IEEE Press, Brisbane, AUS, 2012. p. 605–612.

[10] BROCKHOFF, D.; ZITZLER, E. Objective reduction in evolutionary multiobjective optimization: Theory and applications. **Evolutionary Computation**, vol. 17, n. 2, p. 135–166, 2009.

[11] BROCKHOFF, D.; ZITZLER, E. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. RUNARSSON, T.; BEYER, H.-G.; BURKE, E.; MERELO-GUERVóS, J.; WHITLEY, L.; YAO, X., editores, **Parallel Problem Solving from Nature - PPSN IX**, volume 4193 da Lecture Notes in Computer Science, p. 533–542. Springer Berlin / Heidelberg, 2006.

[12] CAMPOS JUNIOR, A.; POZO, A.; DUARTE JR., E. P. The impact of the topology on multiple swarms particle optimization using asynchronous communication. **The 4th Workshop on Parallel Architectures and Bioinspired Algorithms (WPABA)**. Galveston Island, US, 2011. p. 2–11.

[13] CARVALHO, A. B.; POZO, A. Analyzing the control of dominance area of solutions in particle swarm optimization for many-objective. **2010 10th International Conference on Hybrid Intelligent Systems**. Atlanta, US, 2010. p. 103–108.

[14] CARVALHO, A. B.; POZO, A. The control of dominance area in particle swarm optimization algorithms for many-objective problems. **2010 Eleventh Brazilian**

**Symposium on Neural Networks**. IEEE Computer Society, São Bernardo do Campo, BR, 2010. p. 140–145.

[15] CARVALHO, A. B.; POZO, A.; VERGILIO, S. R. A symbolic fault-prediction model based on multiobjective particle swarm optimization. **Journal of Systems and Software**, vol. 83, p. 868–882, maio. 2010.

[16] CARVALHO, A. B. D.; POZO, A. Measuring the convergence and diversity of cdas multi-objective particle swarm optimization algorithms: A study of many-objective problems. **Neurocomputing**, Amsterdam, The Netherlands, 75, p. 43–51, 2012.

[17] CASTRO, O.; BRITTO, A.; POZO, A. A comparison of methods for leader selection in many-objective problems. **2012 IEEE Congress on Evolutionary Computation (CEC 2012)**. Brisbane, AUS, 2012. p. 1–8.

[18] CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. **IEEE Transactions on Evolutionary Computation**, vol. 6, n. 1, p. 58–73, 2002.

[19] COELLO, C. A.; CORTÉS, N. C. Solving multiobjective optimization problems using an artificial immune system. **Genetic Programming and Evolvable Machines**, Hingham, MA, USA, vol. 6, n. 2, p. 163–190, 2005.

[20] COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. **Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[21] COELLO COELLO, C. A.; LECHUGA, M. S. MOPSO: a proposal for multiple objective particle swarm optimization. **2002 IEEE Congress on Evolutionary Computation (CEC '02)** IEEE Computer Society, Washington, US, 2002. p. 1051–1056.

[22] CORNE, D.; KNOWLES, J. Some multiobjective optimizers are better than others. **2003 IEEE Congress on Evolutionary Computation (CEC '03)**. Canberra, AUS, 2003. p. 2506–2512.

[23] CORNE, D. W.; KNOWLES, J. D. Techniques for highly multiobjective optimisation: some nondominated points are better than others. **9th annual conference on Genetic and evolutionary computation (GECCO '07)**. New York, USA, 2007. p. 773–780.

[24] DAS, I. On characterizing the knee of the Pareto curve based on normal-boundary intersection. **Structural Optimization**, 18, n. 2–3, p. 107–115, 1999.

[25] DE CARVALHO, A. B.; POZO, A. Using different many-objective techniques in particle swarm optimization for many objective problems: An empirical study. **International Journal of Computer Information Systems and Industrial Management Applications**, USA, vol. 3, p. 96–107, 2011.

[26] DE CASTRO, L. N.; VON ZUBEN, F. J. Artificial immune systems: Part I - basic theory and application. relatório técnico tr-dca 01/99, UNICAMP, Campinas, BR, 1999.

[27] DEB, K.; JAIN, H. Handling many-objective problems using an improved NSGA-II procedure. **2012 IEEE Congress on Evolutionary Computation (CEC 2012)**. Brisbane, AUS, 2012. p. 1–8.

[28] DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, vol. 6, n. 2, p. 182–197, 2002.

[29] DEB, K.; MOHAN, M.; MISHRA, S. Towards a quick computation of well-spread Pareto-optimal solutions. FONSECA, C.; FLEMING, P.; ZITZLER, E.; THIELE, L.; DEB, K., editores, **Evolutionary Multi-Criterion Optimization**, volume 2632 da Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Faro, PT, 2003. p. 222–236.

[30] DEB, K.; SUNDAR, J.; N, U. B. R.; CHAUDHURI, S. Reference point based multi-objective optimization using evolutionary algorithms. **8th annual conference on Genetic and evolutionary computation (GECCO '08)**. New York, USA, 2006. p. 635–642.

[31] DEB, K.; THIELE, L.; LAUMANNS, M.; ZITZLER, E. Scalable multi-objective optimization test problems. **2002 IEEE Congress on Evolutionary Computation (CEC '02)**. Washington, US, 2002. p. 825–830.

[32] DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, Cambridge, MA, USA, vol. 7, p. 1–30, 2006.

[33] DI PIERRO, F. **Many-objective evolutionary algorithms and applications to water resources engineering**. Tese de Doutorado, University of Exeter, GB, agosto. 2006.

[34] DRECHSLER, N.; DRECHSLER, R.; BECKER, B. Multi-objective optimisation based on relation favour. Editores ZITZLER, E.; THIELE, L.; DEB, K.; COELLO COELLO, C.; CORNE, D. ZITZLER, E.; THIELE, L.; DEB, K.; COELLO COELLO, C.; CORNE, D., editores, **Evolutionary Multi-Criterion Optimization**. volume 1993 da Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001. p. 154–166.

[35] DURILLO, J. J.; GARCÍA-NIETO, J.; NEBRO, A. J.; COELLO, C. A.; LUNA, F.; ALBA, E. Multi-objective particle swarm optimizers: An experimental comparison. **5th International Conference on Evolutionary Multi-Criterion Optimization (EMO '09)**. Springer-Verlag Berlin, Heidelberg, Nantes, FR, 2009. p. 495–509.

[36] EL-ABD, M.; KAMEL, M. S. A taxonomy of cooperative particle swarm optimizers. **International Journal of Computational Intelligence Research**, USA, vol. 4, n. 2, p. 137–144, 2008.

[37] FIELDSEND, J. E.; UK, E. Q.; SINGH, S. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. **The 2002 U.K. Workshop on Computional Intelligence**. 2002. p. 34–44.

[38] FLEMING, P.; PURSHOUSE, R.; LYGOE, R. Many-objective optimization: An engineering design perspective. COELLO COELLO, C.; HERNÁNDEZ AGUIRRE, A.; ZITZLER, E., editores, **Evolutionary Multi-Criterion Optimization**, volume 3410 da Lecture Notes in Computer Science, p. 14–32. Springer Berlin / Heidelberg, Guanajuato, MX, 2005.

[39] FREITAS, A. A. A critical review of multi-objective optimization in data mining: a position paper. **SIGKDD Explor. Newsl.**, New York, NY, USA, 6, p. 77–86, December. 2004.

[40] GARZA-FABRE, M.; PULIDO, G. T.; COELLO, C. A. Ranking methods for many-objective optimization. MICAI '09. **Proceedings of the 8th Mexican International Conference on Artificial Intelligence**. Springer-Verlag Berlin / Heidelberg, MX, 2009. p. 633–645.

[41] GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[42] HUANG, V. L.; SUGANTHAN, P. N.; LIANG, J. J. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems: Research articles. **International Journal of Intelligent Systems**, New York, NY, USA, 21, p. 209–226, 2006.

[43] HUBAND, S.; HINGSTON, P.; BARONE, L.; WHILE, L. A review of multiobjective test problems and a scalable test problem toolkit. **IEEE Transactions on Evolutionary Computation**, vol. 10, n. 5, p. 477–506, 2006.

[44] HUGHES, E. MSOPS-II: A general-purpose many-objective optimiser. **2007 IEEE Congress on Evolutionary Computation (CEC 2007)**. Cingapura, 2007. p. 3944 –3951.

[45] ISHIBUCHI, H.; AKEDO, N.; OHYANAGI, H.; NOJIMA, Y. Behavior of EMO algorithms on many-objective optimization problems with correlated objectives. **2011 IEEE Congress on Evolutionary Computation (CEC 2011)**. New Orleans, USA, 2011. p. 1465–1472.

[46] ISHIBUCHI, H.; TSUKAMOTO, N.; NOJIMA, Y. Evolutionary many-objective optimization: A short review. **2008 IEEE Congress on Evolutionary Computation (CEC 2008)**. Hong Kong, 2008. p. 2419–2426.

[47] JAIMES, A. L.; COELLO, C. A. C. Study of preference relations in many-objective optimization. **11th Annual conference on Genetic and evolutionary computation (GECCO '09)**, ACM, Montreal, CA, 2009, p. 611–618.

[48] JAIMES, A. L.; COELLO, C. A. C.; CHAKRABORTY, D. Objective reduction using a feature selection technique. **10th annual conference on genetic and evolutionary computation (GECCO '08)**. ACM, Atlanta, USA, 2008. p. 673–680.

[49] JAIMES, A. L.; QUINTERO, L. V. S.; COELLO, C. A. C. Ranking methods in many-objective evolutionary algorithms. **Nature-Inspired Algorithms for Optimisation**, vol. 193, Springer Berlin Heidelberg, p. 413–434, 2009.

[50] JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM Computer Survey**, New York, NY, USA, vol. 31, n. 3, p. 264–323, 1999.

[51] JASZKIEWICZ, A. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. **IEEE Transactions on Evolutionary Computation**, vol. 6, n. 4, p. 402 – 412, 2002.

[52] JUNIOR, O. R. C.; BRITTO, A.; POZO, A. Self-controlling dominance particle swarm optimization. **2012 Eleventh Brazilian Symposium on Neural Networks**. IEEE Computer Society, Curitiba, BR, 2012. p. 172–177.

[53] JUNIOR, O. R. C.; BRITTO, A.; POZO, A. A study on the influence of domination control techniques and leader selection methods in many-objective problems. **First Brazilian Conference on Intelligent Systems (BRACIS 2012)**. BRACIS 2012, Curitiba, 2012.

[54] KENNEDY, J.; EBERHART, R. Particle swarm optimization. **IEEE International Conference on Neural Networks**. IEEE Press, 1995. p. 1942–1948.

[55] KENNEDY, J.; EBERHART, R. **Swarm intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[56] KIN CHOW, C.; TAT TSUI, H. Autonomous agent response learning by a multi-species particle swarm optimization. **2004 IEEE Congress on Evolutionary Computation (CEC'04)**. IEEE Press. Portland, USA, 2004. p. 778–785.

[57] KNOWLES, J.; CORNE, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. **1999 IEEE Congress on Evolutionary Computation (CEC'99)**. IEEE Press, Washington, USA, 1999. p. 3.

[58] KNOWLES, J. D.; CORNE, D. W. Approximating the nondominated front using the pareto archived evolution strategy. **Evolutionary Computation.**, Cambridge, MA, USA, vol. 8, p. 149–172, 2000.

[59] KÖPPEN, M.; YOSHIDA, K. Many-objective particle swarm optimization by gradual leader selection. **8th international conference on Adaptive and Natural Computing Algorithms, Part I (ICANNGA '07)**. Springer-Verlag Berlin / Heidelberg, Warsaw, PO, 2007. p. 323–331.

[60] LAUMANNS, M.; THIELE, L.; DEB, K.; ZITZLER, E. Combining convergence and diversity in evolutionary multiobjective optimization. **Evolutionary Computation**, Cambridge, MA, USA, vol. 10, p. 263–282, 2002.

[61] LAUMANNS, M.; ZENKLUSEN, R. Stochastic convergence of random search methods to fixed size Pareto front approximations. **European Journal of Operational Research**, vol. 213, n. 2, p. 414 – 421, 2011.

[62] LI, X. A non-dominated sorting particle swarm optimizer for multiobjective optimization. **Genetic and Evolutionary Computation (GECCO 2003)**, volume 2723 da Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Chicago, USA, 2003. p. 198–198.

[63] LINDROTH, P.; PATRIKSSON, M.; STRöMBERG, A.-B. Approximating the pareto optimal set using a reduced set of objective functions. **European Journal of Operational Research**, vol. 207, n. 3, p. 1519–1534, 2010.

[64] LÓPEZ-IBÁÑEZ, M.; KNOWLES, J.; LAUMANNS, M. On sequential online archiving of objective vectors. TAKAHASHI, R.; DEB, K.; WANNER, E.; GRECO, S., editores, **Evolutionary Multi-Criterion Optimization**, volume 6576 da Lecture Notes in Computer Science, p. 46–60. Springer Berlin / Heidelberg, 2011.

[65] LÓPEZ JAIMES, A.; COELLO COELLO, C. A. Some techniques to deal with many-objective problems. **11th Annual Conference Companion on Genetic and Evolutionary Computation Conference (GECCO '09)**. ACM, Montreal, CA, 2009. p. 2693–2696.

[66] MIETTINEN, K. **Nonlinear Multiobjective Optimization**. Kluwer Academic Publishers, Dordrecht, 1999. volume 12 da International Series in Operations Research and Management Science.

[67] MOSTAGHIM, S.; TEICH, J. Strategies for finding good local guides in multi-objective particle swarm optimization. **Swarm Intelligence Symposium (SIS '03)**. IEEE Computer Society, Indianapolis, USA, 2003. p. 26–33.

[68] MOSTAGHIM, S.; BRANKE, J.; SCHMECK, H. Multi-objective particle swarm optimization on computer grids. **9th annual conference on Genetic and evolutionary computation (GECCO '07)**. ACM, Londres, GB, 2007. p. 869–875.

[69] NEBRO, A. J.; LUNA, F.; ALBA, E.; DORRONSORO, B.; DURILLO, J. J.; BEHAM, A. Abyss: Adapting scatter search to multiobjective optimization. **IEEE Transactions on Evolutionary Computation**, vol. 12, n. 4, p. 439–457, 2008.

[70] NEBRO, A.; DURILLO, J.; GARCIA-NIETO, J.; COELLO, C. A. C.; LUNA, F.; ALBA, E. SMPSO: A new pso-based metaheuristic for multi-objective optimization. **IEEE symposium on Computational intelligence in miulti-criteria decision-making (mcdm '09)**. Nashville, USA, 2009. p. 66–73.

[71] NEBRO, A.; DURILLO, J.; LUNA, F.; DORRONSORO, B.; ALBA, E. Design issues in a multiobjective cellular genetic algorithm. OBAYASHI, S.; DEB, K.; POLONI, C.; HIROYASU, T.; MURATA, T., editores, **Evolutionary Multi-Criterion Optimization**, volume 4403 da Lecture Notes in Computer Science, p. 126–140. Springer Berlin / Heidelberg, 2007.

[72] PADHYE, N.; BRANKE, J.; MOSTAGHIM, S. Empirical comparison of MOPSO methods - guide selection and diversity preservation . **Evolutionary Computation**, p. 2516–2523, 2009.

[73] PARSOPOULOS, K. E.; TASOULIS, D. K.; VRAHATIS, M. N.; WORDS, K. Multiobjective optimization using parallel vector evaluated particle swarm optimization. **In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)**. ACTA Press, 2004. p. 823–828.

[74] PARSOPOULOS, K. E.; VRAHATIS, M. N. Recent approaches to global optimization problems through particle swarm optimization. **Natural Computing**, vol. 1, p. 235–306, 2002.

[75] PIERRO, F. D.; THIAM KHU, S.; SAVIC, D. A. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. **IEEE Transactions on Evolutionary Computation**, vol. 11, p. 17–45, 2007.

[76] PULIDO, G.; COELLO COELLO, C. Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. DEB, K., editor, **Genetic and Evolutionary Computation, GECCO 2004**, volume 3102 da Lecture Notes in Computer Science, p. 225–237. Springer Berlin / Heidelberg, 2004.

[77] PURSHOUSE, R. C.; JALBA, C.; FLEMING, P. J. Preference-driven co-evolutionary algorithms show promise for many-objective optimisation. **6th international conference on Evolutionary multi-criterion optimization (EMO'11)**. Springer-Verlag Berlin / Heidelberg, Ouro Preto, BR, 2011. p. 136–150.

[78] R Core Team. **R: A Language and Environment for Statistical Computing**. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[79] REYES-SIERRA, M.; COELLO, C. A. C. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. **International Journal of Computational Intelligence Research**, vol. 2, n. 3, p. 287–308, 2006.

[80] SATO, H.; AGUIRRE, H.; TANAKA, K. Self-controlling dominance area of solutions in evolutionary many-objective optimization. DEB, K.; BHATTACHARYA, A.; CHAKRABORTI, N.; CHAKROBORTY, P.; DAS, S.; DUTTA, J.; GUPTA, S.; JAIN, A.; AGGARWAL, V.; BRANKE, J.; LOUIS, S.; TAN, K., editores, **Simulated Evolution and Learning**, volume 6457 da Lecture Notes in Computer Science, p. 455–465. Springer Berlin / Heidelberg, 2010.

[81] SATO, H.; AGUIRRE, H. E.; TANAKA, K. Controlling dominance area of solutions and its impact on the performance of MOEAs. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization. Springer Verlag Berlin / Heidelberg , Guanajuato, MX, 2007. p. 5–20.

[82] SCHÜTZE, O.; LARA, A.; COELLO, C. A. C. On the influence of the number of objectives on the hardness of a multiobjective optimization problem. **IEEE Transactions Evolutionary Computation**, vol. 15, n. 4, p. 444–455, 2011.

[83] SIERRA, M. R.; COELLO, C. A. C. Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance. **Evolutionary Multi-Criterion Optimization (EMO 2005), LNCS 3410**. Springer-Verlag Berlin / Heidelberg, Guanajuato, MX, 2005. p. 505–519.

[84] VELDHUIZEN, D. A. V.; LAMONT, G. On measuring multiobjective evolutionary algorithm performance. **2000 IEEE Congress on Evolutionary Computation (CEC 2000)**. IEEE press, 2000. p. 204–211.

[85] WICKRAMASINGHE, U. K.; LI, X. Using a distance metric to guide pso algorithms for many-objective optimization. **11th Annual conference on Genetic and evolutionary computation (GECCO '09)**. ACM, Montreal, CA, 2009. p. 667–674.

[86] ZHANG, Q.; LI, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. **IEEE Transactions on Evolutionary Computation**, vol. 11, n. 6, p. 712–731, 2007.

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

[88] ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. **Evolutionary Computation**, vol. 8, p. 173–195, junho. 2000.

[89] ZITZLER, E.; KüNZLI, S. Indicator-based selection in multiobjective search. **8th International Conference on Parallel Problem Solving from Nature (PPSN VIII**. Springer Verlag Berlin / Heidelberg, Birmingham, GB, 2004. p. 832–842.

[90] ZITZLER, E.; LAUMANNS, M.; THIELE, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Editor ET AL., K. G. ET AL., K. G., editor, **Evolutionary Methods for Design, Optimisation, and Control**. CIMNE, Barcelona, ES, 2002. p. 95–100.

[91] ZITZLER, E.; THIELE, L.; LAUMANNS, M.; FONSECA, C. M.; DA FONSECA, V. G. Performance assessment of multiobjective optimizers: an analysis and review. **IEEE Transactions on Evolutionary Computation**, vol. 7, n. 2, p. 117–132, 2003.