Universidade Federal de Pernambuco
Centro de Informática

Pós-graduação em Ciência da Computação

Algoritmo Baseado em Enxame de Partículas para Otimização de Problemas com Muitos Objetivos

Elliackin Messias do Nascimento Figueiredo

Dissertação de Mestrado

Recife
Fevereiro de 2013

Universidade Federal de Pernambuco
Centro de Informática

Elliackin Messias do Nascimento Figueiredo

Algoritmo Baseado em Enxame de Partículas para Otimização de Problemas com Muitos Objetivos

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: *Profa. Teresa Bernarda Ludermir, PhD*
Co-orientador: *Prof. Dr. Carmelo José Albanez Bastos Filho*

Recife
Fevereiro de 2013

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Figueiredo, Elliackin Messias do Nascimento
Algoritmo baseado em enxame de partículas para otimização de problemas com muitos objetivos. / Elliackin Messias do Nascimento Figueiredo. - Recife: O Autor, 2013.
xxi, 120 p.: fig., tab.

Orientador: Teresa Bernarda Ludermir.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui bibliografia.

1. Inteligência artificial. 2. Inteligência computacional. 3. Computação evolucionária. I. Ludermir, Teresa Bernarda (orientadora). II. Título.

006.3 CDD (23. ed.) MEI2013 – 072

Dissertação de Mestrado apresentada por **Elliackin Messias do Nascimento Figueiredo** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Algoritmo Baseado em Enxame de Partículas para Otimização de Problemas com Muitos Objetivos**" orientada pela **Profa. Teresa Bernarda Ludermir** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE

_____

Profa. Aurora Trinidad Ramirez Pozo
Departamento de Informática / UFPR

_____

Prof. Carmelo José Albanez Bastos Filho
Departamento de Sistemas Computacionais / UPE
(Co-orientador)

Visto e permitida a impressão.
Recife, 25 de fevereiro de 2013.

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

À minha mãe Mazé, por todo apoio e ajuda, e também a minha família.

Agradecimentos

À minha mãe, Maria José do Nascimento Figueiredo, pelo amor, carinho e dedicação presente em toda a minha vida; e por toda ajuda prestada e motivação que tornou possível a realização de mais um sonho. Ao meu pai, Severino Guerra Figueiredo, por ter me ensinado valores como honestidade e trabalho. E ao meu irmão, Guilherme Raimundo do Nascimento Figueiredo, pela paciência e presença em minha vida.

À minha orientadora, Profa. Teresa, por sua colaboração e ensinamentos. Ao meu co-orientador, Prof. Carmelo, pela colaboração e pela gentileza de ter aceitado me orientar.

A Andreas Janecek pela ajuda e ensinamentos durante o desenvolvimento desse trabalho. E também a todas as outras pessoas que contribuíram direta ou indiretamente na realização dessa dissertação.

Teria sido isso a vida?
Pois muito bem:
Outra vez!

—FRIEDRICH NIETZSCHE (Assim Falava Zaratustra, 1883-1885)

Resumo

Otimização de Muitos Objetivos consiste na otimização de problemas com muitos objetivos, isto é, problemas multiobjetivos com um elevado número de objetivos, geralmente mais de três. Atualmente, essa área é uma área ativa com respeito ao campo de algoritmos evolucionários. Em problemas como esses, algoritmos que utilizam a dominância de Pareto como critério de atribuição de aptidão tais como MOEAs e MOPSOs se tornam inefetivos, pois praticamente todas as soluções da população tendem a ser tornar não-dominadas, levando a perda da pressão de convergência para a Frente de Pareto. A ineficácia desses algoritmos levou os pesquisadores a proporem estratégias alternativas a dominância de Pareto para lidar com esses problemas, principalmente para os MOEAs. Contudo, pouco tem sido feito no sentido de tornar os MOPSOs efetivos em problemas com muitos objetivos. Na literatura, os MOPSOs propostos para lidar com esses problemas apresentam muitas dificuldades, tais como parâmetros difíceis de ajustar, necessidade de conhecimento sobre o problema, e a incapacidade de convergência em problemas com multimodalidade. Nesse trabalho, um novo algoritmo baseado em enxame de partículas para problemas com muitos objetivos foi proposto e foi denominado de MOPSO-GD. O MOPSO-GD caracteriza-se por esquemas melhorados para (*i*) a seleção dos líderes sociais, (*ii*) seleção dos líderes cognitivos e (*iii*) poda do arquivo externo. Todos esses esquemas são baseados em um método de alta granularidade denominado de Detrimento Global. O Detrimento Global foi usado como um método para promover a convergência e promover a habilidade do MOPSO-GD de lidar com um grande número de objetivos. Para validar o MOPSO-GD, ele foi avaliado usando quatro problemas de teste escaláveis bem conhecidos (DTLZ{1,3,4,6}) com 5, 10, 15, 20, 30 e 50 objetivos; e foi comparado com duas abordagens baseadas em enxame de partículas (MOPSO-CDR e SMPSO) e dois algoritmos evolucionários estado da arte para problemas com muitos objetivos (CEGA e MDFA). Os resultados mostraram que o MOPSO-GD obteve bom desempenho em termos de convergência, enquanto manteve os níveis de diversidade do CEGA e do MDFA.

Palavras-chave: Otimização de Muitos Objetivos; Problemas com Muitos Objetivos; Enxame de Partículas; Detrimento Global.

Abstract

Many-objective optimization refers to the optimization of many-objective problems, that is, multiobjective problems containing a large number of objectives, typically more than three. Currently, many-objective optimization is one of the most active research areas in the field of evolutionary computation. Algorithms based on Pareto dominance such as MOEAs and MOPSOs are inadequate for these problems, because almost all solutions in the population become non-dominated, resulting in loss of convergence pressure for the Pareto front. Because this, some researches have proposed other strategies for leading with this problems, mainly for MOEAs. However, very little has been done to make the MOPSOs effective in these scenarios. In the literature, the MOPSOs applied in these problems presents many difficults, such as problem-dependent parameters whose optimal setting is difficult to find, they require some knowledge about the problem and they do not converge in problems with many local Pareto fronts. In this paper, we propose a new algorithm for many-objective problems which is based on particle swarm optimization. The algorithm called MOPSO-GD features improved schemes for (*i*) the selection of the social leaders, (*ii*) the update of the cognitive leaders, and (*iii*) the pruning of the external archive, which are based on a fine-grained ranking method called Global Detriment. The Global Detriment was used as a method in order to promote the convergence and the ability of the MOPSO-GD to deal with a large number of objectives. In order to validate our proposal we present a comparative study considering two PSO-based (MOPSO-CDR, SMPSO) and two state of the art evolutionary many-objective algorithms (CEGA, MDFA) using four well-known scalable benchmark problems (DTLZ{1,3,4,6}) with 5, 10, 15, 20, 30 and 50 objectives, respectively. The results showed that the MOPSO-GD obtained a good performance in terms of convergence, while it maintains the level of diversity similar to the CEGA and MDFA.

Keywords: Many-Objective Optimization; Many-Objective Problems; Multi-Objective Problems; Particle Swarm Optimization; Global Detriment.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

" *Não existem métodos fáceis para resolver problemas difíceis* "
– René Descartes

OTIMIZAÇÃO é uma área ubíqua. Ela está presente em processos de tomada de decisão e na análise de sistemas físicos, de modo que ela está presente em praticamente todas as áreas do conhecimento, da engenharia a ciência e da economia a ciências sociais. Otimização é uma atividade inerente ao ser humano. Investidores, por exemplo, buscam criar portfólios que evitem riscos excessivos enquanto alcançam uma alta taxa de retorno [NW99]. Industriais, por sua vez, desejam maximizar a eficiência de seus processos de produção [NW99]. Além disso, a natureza segue princípios de otimização. Sistemas físicos tendem para um estado de mínima energia. As moléculas em um sistema químico isolado, por exemplo, reagem umas com as outras até que a energia potencial dos seus elétrons seja minimizada [NW99]. Raios de luz viajam entre dois pontos no menor tempo possível (principio de Fermat) [Rao09, Yan10]. Esses são apenas alguns exemplos que mostram a abrangência e interdisciplinaridade dessa área do conhecimento. Ademais, como é possível inferir, otimização tem várias implicações de ordem prática, ao minimizar o custo e/ou aumentar a eficácia de um produto industrial por exemplo, como de ordem teórica, uma vez que muitos fenômenos naturais são regidos por princípios de otimização.

Para resolver um problema de otimização do mundo real, é necessário modelá-lo adequadamente. A construção de um modelo apropriado é o primeiro e geralmente o mais importante passo no processo de otimização [NW99]. Se o modelo é bastante simplista, ele não é capaz de fornecer informações úteis sobre o problema, mas se ele é bastante complexo, sua solução pode se tornar inviável. A *modelagem* de um problema de otimização consiste no processo de identificar as funções objetivos, as variáveis de decisão e as restrições do problema [NW99]. A função objetivo ou simplesmente objetivo é um medida quantitativa do desempenho do sistema ou processo em estudo. A função objetivo pode ser, por exemplo, o custo de um produto, o tempo gasto em um processo de produção em uma indústria, e a energia potencial de um sistema físico. A função objetivo depende de certas variáveis do sistema ou processo chamadas de variáveis de decisão ou de projeto. Essas variáveis, em muitos problemas práticos, não podem assumir valores arbitrários. Ao invés, elas devem satisfazer certas restrições e requisitos de modo a produzir uma solução de projeto viável [Rao09]. Tais restrições são coletivamente denominadas de restrições de projeto [Rao09].

Uma vez que o modelo do problema tenha sido formulado, o processo de otimização consiste em encontrar a melhor solução possível, isto é, encontrar a solução que maximiza ou minimiza a função objetivo e que satisfaça as restrições. Quando o problema de otimização consiste em encontrar a solução que faz com que a função objetivo atinja o seu menor valor possível, o problema é dito de minimização. Por outro lado, quando o problema de otimização consiste em encontrar a solução que faz com que a função objetivo atinja o seu maior valor possível, o problema é dito de maximização [Rao09].

Problemas de otimização com uma única função objetivo são denominados de problemas mono-objetivo. Contudo, em muitos problemas reais, é comum o caso em que mais de um objetivo é envolvido no processo de otimização [Zit99, Gar09, Coe06]. Por exemplo, no projeto de uma ponte, pode-se desejar minimizar o custo enquanto se maximiza a segurança [Coe06]. Esses problemas de otimização são chamados de Problemas Multiobjetivos (**MOP**, *Multiobjective Problem*). É importante destacar que geralmente os objetivos envolvidos no processo de otimização são conflitantes entre si [Zit99]. Se isso não acontece, o problema tem apenas uma única solução ótima, a menos de multimodalidade [1]. De fato, essa solução seria encontrada resolvendo-se cada objetivo em separado. Por outro lado, assumindo que haja conflitos entre os objetivos, os problemas multiobjetivos apresentam um conjunto de soluções consideradas ótimas.

A noção de otimalidade em problemas multiobjetivos é diferente daquela usada em um problema mono-objetivo. Enquanto em problemas mono-objetivo se busca encontrar a solução que otimiza, isto é, minimiza ou maximiza uma função objetivo. Tal conceito não pode ser aplicado em problemas multiobjetivos pois existem várias funções objetivo, geralmente conflitantes, a serem otimizadas simultaneamente. Assim, a noção geralmente adotada de otimalidade em problemas multiobjetivos é a que foi proposta por Edgeworth em 1881 [Edg81, Coe06] e que depois foi generalizada por Vilfredo Pareto em 1896 [Par96, Coe06]. Segundo essa noção, uma solução é dita ser ótima no sentido de Pareto, ou simplesmente Pareto ótima, se não for possível melhorar um objetivo sem piorar simultaneamente no mínimo outro objetivo. O conjunto das soluções que atendem essa condição é chamado de conjunto Pareto ótimo. As soluções desse conjunto quando projetadas no espaço objetivo formam um superfície denominada de Frente de Pareto. O espaço objetivo é o espaço formado quando as funções objetivos são tomadas como eixos coordenados.

Para ilustrar um problema de otimização multiobjetivo bem como o seu conceito de otimalidade, considere o exemplo de minimizar simultaneamente as funções (função de teste Schaffer [Sch85]):

$$\text{minimizar} \begin{cases} f_1(x) = x^2, \\ f_2(x) = (x-2)^2. \end{cases}$$

O gráfico da esquerda na Figura 1.1 apresenta os gráficos das duas funções. A função $f_1(x)$ tem valor mínimo (valor zero) no ponto $x_1^* = 0$. Por outro lado, esse mesmo ponto tem valor igual a 4 quando a função $f_2(x)$ é considerada, que corresponde a um valor relativamente alto, considerando que o menor valor dessa função também é zero. Para diminuir o valor da função

[1] Uma função é dita ser multimodal se ela contém vários picos ou vales [Rao09].

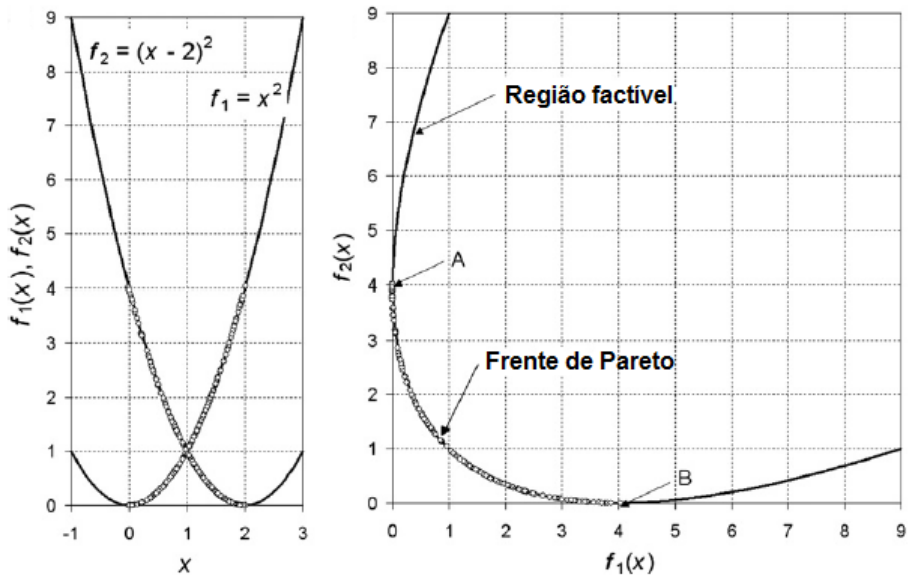


Figura 1.1 Conceitos envolvidos na otimização multiobjetiva (adaptado de [Sek10]).

$f_2(x)$ é necessário avançar para o lado direito do gráfico. À medida que se avança, contudo, o valor de $f_1(x)$ começa a aumentar de tal forma que, quando o valor mínimo de $f_2(x)$ é alcançado no ponto $x_2^* = 2$, a função $f_1(x)$ tem um valor alto ($f_1(x) = 4$). Após o ponto $x_2^* = 2$, as duas funções crescem simultaneamente. Como se pode observar por esse exemplo, no intervalo $[0,2]$ existe um conflito entre as funções pois um valor alto na função $f_1(x)$ corresponde um valor baixo na função $f_2(x)$ e vice-versa. O gráfico da direita na Figura 1.1 apresenta as duas funções em um único plano cartesiano parametrizadas pela variável x. Esse plano, nesse exemplo em particular, corresponde ao espaço objetivo. Como se pode observar pelo gráfico da esquerda na Figura 1.1, os pares ordenados $(f_1(x), f_2(x))$ quando x varia no intervalo de $[-1,3]$ formam uma superfície que, nesse exemplo em particular, consiste em uma curva. Essa superfície é denominada de região factível. O subconjunto dessa curva que na figura aparece em destaque é denominado de Frente de Pareto. A Frente de Pareto é formada pelo conjunto de pontos no espaço objetivo que representa o melhor compromisso possível entre os objetivos. Por melhor compromisso, entende-se o fato de que, dado um ponto nessa superfície, não é possível diminuir o valor de um dos objetivos desse ponto sem que se aumente o valor de outro objetivo. Por exemplo, considere o ponto $(1,9)$ no espaço objetivo que pertence a região factível. Essa solução não representa o melhor compromisso possível entre os objetivos, pois é possível diminuir o segundo objetivo sem aumentar o primeiro. O ponto $(1,1)$, por outro lado, é um exemplo de uma solução que representa o melhor compromisso possível. Ou seja, o ponto $(1,1)$ representa uma solução Pareto-ótima pois não é possível diminuir o primeiro objetivo sem aumentar o segundo e vice-versa. As soluções Pareto-ótima são todas consideradas ótimas pois, na ausência de qualquer preferência sobre os objetivos, não é possível afirmar que uma é melhor que a outra. Por exemplo, não há, a priori, nenhuma razão para preferir a solução $A = (4,0)$ sobre a solução $B = (1,1)$ ou vice-versa. A solução A tem um valor ruim no primeiro

objetivo, mas em compensação tem um valor muito bom para o segundo objetivo em relação a solução B. A solução B, por outro lado, tem um valor melhor que a solução A para o primeiro objetivo, mas tem um valor pior para o segundo objetivo.

Como já foi mencionado, problemas multiobjetivos surgem naturalmente em muitos problemas reais das mais variadas áreas do conhecimento. Em virtude dessa ubiquidade, muitas técnicas têm sido desenvolvidas ao longos dos anos com o objetivo de resolvê-los. Em particular, a comunidade de Pesquisa Operacional tem desenvolvido abordagens para resolver problemas multiobjetivos desde de 1950 [Coe06, Gar09]. Esses métodos, sejam eles lineares ou não-lineares, determinísticos ou estocásticos, são agrupados sobre a rubrica de *programação matemática* [CLV07]. Contudo, essas técnicas têm certas limitações ao serem utilizadas para resolverem problemas multiobjetivos [Coe06]. Por exemplo, muitas delas são suscetíveis a forma da Frente de Pareto e podem não funcionar quando a Frente de Pareto é côncava ou desconectada [Coe06]. Um exemplo disso é o método da soma ponderada, que consiste em transformar o problema multiobjetivo em um problema mono-objetivo por meio da combinação linear dos objetivos, cuja principal desvantagem é o fato de não poder gerar todas as soluções em problemas com Frente de Pareto não-convexas [CLV07, Zit99]. Além disso, muitas dessas técnicas geram somente uma única solução por execução. Dessa forma, várias execuções independentes, usando diferentes parâmetros, são necessárias para gerar vários pontos da Frente de Pareto [Coe06]. Esse é caso também do método da soma ponderada em que é necessário utilizar diferentes conjuntos de pesos para produzir diferentes pontos sobre a Frente de Pareto. Ou exemplo de técnica que precisa ser executada várias vezes, é o método da ε-restrição [CLV07, Adr07]. Nesse método uma das funções objetivos é escolhida para ser otimizada, enquanto as outras funcionam como restrições. Assim, essa técnica necessita que sejam especificados valores ε que não podem ser excedidos (no caso de minimização) pelas funções objetivos escolhidas como restrições. Variando esses valores em cada execução, é possível encontrar diferentes soluções Pareto-ótimas [Zit99, CLV07]. Esse procedimento de realizar várias execuções independentes pode ser custoso computacionalmente, dependo da aplicação, uma vez que pode se estar perdendo informação quando se encontra apenas uma única solução por vez [Zit99]. As limitações das técnicas de programação matemática, motivaram o uso de técnicas heurísticas para resolver problemas multiobjetivos.

Heurística é uma solução estratégica por tentativa e erro para produzir soluções aceitáveis para problemas complexos em um tempo razoável. Ao se usar algoritmos heurísticos, não há nenhuma garantia de que as soluções ótimas do problema sejam encontradas. Apesar disso, o que se espera é que esses algoritmos funcionem na maioria das vezes e produzam boas soluções. Uma das características dos algoritmos heurísticos é que não se sabe se esses algoritmos funcionarão e porque eles funcionam. O matemático inglês Alan Turing foi provavelmente o primeiro a usar algoritmos heurísticos na resolução de um problema complexo [Yan10]. Durante a Segunda Guerra Mundial, em 1940, Turing desenvolveu um algoritmo heurístico que ele denominou de *busca heurística* para buscar, entre 10^{22} combinações possíveis, as diferentes formas de decodificar as mensagens cifradas alemãs produzidas pelo Enigma [2] [Yan10]. A próxima seção apresenta alguns algoritmos heurísticos desenvolvidos para resolver proble-

[2] Alan Turing, em um relatório sobre "*Intelligent Machine*"em 1948, também mencionou ideias inovadoras sobre inteligência de máquina e aprendizado, redes neurais e algoritmos evolucionários [Yan10].

mas multiobjetivos. Em particular, ela dá ênfase a duas classes de algoritmos heurísticos (ou meta-heurísticas para ser mais preciso, como será visto mais adiante), a saber, os Algoritmos Evolucionários e a Otimização por Enxame de Partículas.

1.1 Algoritmos Evolucionários e Enxame de Partículas Para MOPs

Meta-heurísticas são uma família de técnicas de otimização que tem se tornado popular nas últimas três décadas [Tal09]. A palavra *heurística* tem sua origem na palavra grega *heuriskein*, que significa a arte de descobrir novas estratégias para resolver problemas [Tal09]. O prefixo *meta* também de origem grega tem a conotação de "alto nível"; de algo superior a algo. Sendo assim, meta-heurísticas são modelos que estão em um nível superior que podem ser usados para guiar o projeto de heurísticas subjacentes para resolver problemas de otimização específicos [Tal09]. As heurísticas são dependentes do problema; elas são projetadas e aplicadas a um problema em particular. As meta-heurísticas, por outro lado, são algoritmos gerais aplicáveis a uma grande variedade de problemas de otimização [Tal09].

A principal motivação para o uso de meta-heurísticas é que um grande número de problemas práticos de otimização são complexos e difíceis de resolver [Tal09]. Eles não podem ser resolvidos de uma maneira exata dentro de uma quantidade razoável de tempo [Tal09]. Assim, o uso de meta-heurísticas é uma das principais alternativa para a solução desses problemas [Tal09]. As meta-heurísticas resolvem problemas complexos usando princípios e regras simples que lhes permitem explorar gradualmente o espaço de possíveis soluções do problema. À medida que elas progridem, as meta-heurísticas descobrem regiões promissoras (com boas soluções) nesse espaço e gradualmente reduzem essas regiões realizando uma busca mais refinada, na qual espera-se encontrar a solução ótima. Contudo, não há nenhuma garantia de que tal solução seja encontrada. Em suma, as meta-heurísticas servem para resolver problemas complexos nos quais os métodos clássicos de otimização falham e/ou não conseguem fornecer uma solução em tempo prático [Tal09, Yan10]. Por métodos clássicos, entendem-se os métodos que se utilizam de informações analíticas do problema, entre esses métodos têm-se os métodos baseados em gradiente, o método dos multiplicadores de Lagrange e os métodos de programação matemática [NW99].

Ao longo das últimas três décadas, muitas meta-heurísticas foram desenvolvidas, e elas estão se tornando cada vez mais populares em diferentes áreas de pesquisa e na industria [Tal09, Yan10]. De fato, uma vasta maioria das técnicas de otimização modernas são geralmente heurísticas e/ou meta-heurísticas [Yan10]. Muitas delas imitam processos presentes na natureza, como a evolução das espécies, a cooperação entre as abelhas em uma colmeia, o sistema imunológico humano, entre outras [Tal09, Yan10]. Algumas delas são: Otimização por Colônia de Formigas, Sistemas Imunológicos Artificiais, Algoritmos Culturais, Recozimento Simulado, Algoritmos Evolucionários, Otimização por Enxames de Partículas, entre outras. Essas técnicas têm sido aplicadas nos mais variados tipos de problemas de otimização, tanto em uma formulação mono-objetiva como também multiobjetiva. Em particular, os Algoritmos Evolucionários foram uma das primeiras abordagens a serem aplicadas em problemas multiobjetivos, principalmente devido ao fato de utilizarem um conjunto de soluções que lhes permitiam encontrar várias soluções da Frente de Pareto em uma única execução [CLV07, Gar09, LKC+12].

Os Algoritmos Evolucionários (**EA**, *Evolutionary Algorithm*) são algoritmos que imitam os processos envolvidos na teoria da evolução para resolver problemas de otimização [Gar09, Eng07]. Para isso, esses algoritmos iniciam com um conjunto de soluções candidatas aleatórias. Esse conjunto de soluções é denominado de população. Essa população é então progressivamente melhorada por meio da modificação de suas soluções. Essas modificações são realizadas por regras que simulam os processos de cruzamento, mutação e seleção natural que regem a evolução das espécies. O cruzamento em um EA consiste em combinar duas ou mais soluções de alguma maneira para produzir duas ou mais soluções filhas. A ideia por trás disso é que se soluções pais são boas soluções, elas provavelmente produzirão soluções filhas melhores. A mutação cria alterações aleatórias em soluções existentes na população de modo que eventualmente possam gerar soluções com boa qualidade. Por fim, a seleção natural é simulada pela preservação das melhores soluções para a próxima geração (iteração). A qualidade de uma solução é dada por sua aptidão. Assim, uma solução é dita ser melhor que a outra se sua aptidão for maior do que a da outra solução. O modo como a aptidão é atribuída a cada solução da população depende do EA, isto é, cada EA tem seu próprio método de atribuição de aptidão.

O primeiro algoritmo evolucionário desenvolvido para lidar com problemas multiobjetivos foi proposto por David Schaffer em meados de 1980 e foi denominado de *Vector Evaluated Genetic Algorithm* (VEGA) [Sch84, Sch85]. A partir de então, muitos outros algoritmos foram desenvolvidos. Esses algoritmos são chamados genericamente de Algoritmos Evolucionários Multiobjetivos (**MOEA**, *Multiobjective Evolutionary Algorithm*) [CLV07]. O principal desafio em qualquer MOEA, e mais geralmente em qualquer algoritmo que se proponha a resolver um problema multiobjetivo, é conciliar dois requisitos, a saber, a convergência e a diversidade. A convergência consiste no fato de que todas as soluções produzidas pelo algoritmo devem pertencer à Frente de Pareto do problema, e a diversidade consiste em encontrar soluções que cubram toda a extensão da Frente de Pareto e que simultaneamente estejam uniformemente espalhadas sobre ela. Convergência e diversidade são requisitos geralmente conflitantes [Kha02].

Nessa perspectiva, surgiram os MOEAs da primeira geração [Coe06] que buscavam unir os requisitos de convergência e diversidade simultaneamente em um MOEA. Para promover a convergência, esses MOEAs passaram a incorporar um método de atribuição de aptidão baseado no conceito de dominância de Pareto. A dominância de Pareto é uma relação entre duas soluções que diz que uma solução A domina uma outra solução B, se A não é pior que B em todos os objetivos e se A é melhor que B em no mínimo um objetivo [CLV07]. A partir desse conceito, pode-se estabelecer comparações entre uma solução com o resto da população. Assim, uma solução é dita ser não-dominada se nenhuma solução da população a domina; e uma solução é dita dominada se existe pelo menos uma solução que a domina. O grande mérito dos MOEAs da primeira geração é a ênfase dada as soluções não-dominadas da população por meio de métodos de atribuição de aptidão que favoreciam essas soluções. Procedendo-se dessa maneira, as soluções tendem a progredir em direção à Frente de Pareto [Pur03].

A partir do final de 1990, iniciou-se o desenvolvimento dos MOEAs da segunda geração [Coe06]. Esses algoritmos também utilizavam métodos de atribuição de aptidão baseados em dominância de Pareto de modo a priorizar as soluções não-dominadas em relação às dominadas. No entanto, esses algoritmos também incorporavam novos mecanismos e conceitos. Em particular, esses algoritmos apresentavam mecanismos de elitismo, que consistiam em pre-

servar soluções não-dominadas encontradas durante todo o processo de busca [Coe06, Pur03, Adr07]. Além disso, nessa geração, novos métodos para promoção da diversidade foram propostos [Coe06, Pur03]. Sendo que alguns desses métodos eram livres de parâmetros. Atualmente, os algoritmos da segunda geração constituem o estado da arte da área, sendo que eles têm alcançado resultados bastante promissores em aplicações reais [CLV07].

Outra meta-heurística que também tem sido aplicado na solução de problemas multiobjetivos é a Otimização por Enxame de Partículas (**PSO**, *Particle Swarm Optimization*) [KE95]. O PSO é uma meta-heurística inspirada no comportamento social de organismos tais como aves, peixes e insetos. O PSO, assim como os algoritmos evolucionários, também utiliza um conjunto de soluções candidatas denominado de enxame, e cada uma dessas soluções é denominada de partícula. A principal ideia do PSO é que a troca de informações entre as partículas sobre regiões promissoras combinada com a experiência individual de cada partícula sobre regiões já visitadas por elas podem levar o algoritmo a encontrar a solução ótima do problema. O PSO foi originalmente proposto para resolver problemas mono-objetivo [KE95]. Entretanto, as suas características, tais como o fato de utilizaram um conjunto de soluções candidatas de modo similar aos algoritmos evolucionários, rápida convergência e simplicidade, fizeram dele uma alternativa promissora aos algoritmos evolucionários na solução de problemas multiobjetivos [RSC06, MS08, CLV07]. As várias formas de adaptar o PSO para problemas multiobjetivos levaram ao surgimento de vários algoritmos denominados genericamente de Otimização Multiobjetiva por Enxame de Partículas (**MOPSO**, *Multiobjective Particle Swarm Optimization*) [RSC06, CLV07]. Assim como ocorre com os MOEAs, os MOPSOs mais promissores utilizam dominância de Pareto para qualificar as soluções de modo a favorecer as soluções não-dominadas sobre as dominadas [ZQL$^+$11, CLV07, Pur03].

Devido à aplicação promissora de MOEAs e MOPSOs em problemas reais e ao potencial que tais abordagens possuem na solução de problemas complexos, esta dissertação trata da solução de problemas multiobjetivos por algoritmos evolucionários e por enxame de partículas. Em particular, ela trata de adaptar tais abordagens para lidarem com problemas multiobjetivos com mais de quatro objetivos. Essa questão será melhor detalhada na seção seguinte.

1.2 Motivação e Justificativa

Como discutido na seção anterior, os MOEAs da segunda geração têm alcançado sucesso na solução de problemas multiobjetivos, tanto práticos como teóricos. No entanto, eles têm sido aplicados a problemas com apenas dois ou três objetivos. Raramente, eles têm sido aplicados em problemas com mais de três objetivos. Para ilustrar essa afirmação, a Figura 1.2 apresenta o gráfico do número de publicações pelo número de objetivos.

Como se pode observar por esse gráfico, até 2007 existem poucos trabalhos lidando com problemas com mais de quatro objetivos. Contudo, muitos problemas reais envolvem um grande número de objetivos [Gar09, FPL05, RHH$^+$13], como por exemplo o campo de recursos de água [RHH$^+$13], a seleção de portifólio social [FLM$^+$13] e a calibração de motor de automóvel [LCF10]. Por essa razão, existe a necessidade de se aplicar MOEAs nesses problemas. Essa necessidade motivou os pesquisadores a estudarem a escalabilidade dos MOEAs em problemas dessa natureza. Nesse sentido, Khare [Kha02] foi provavelmente o primeiro a in-

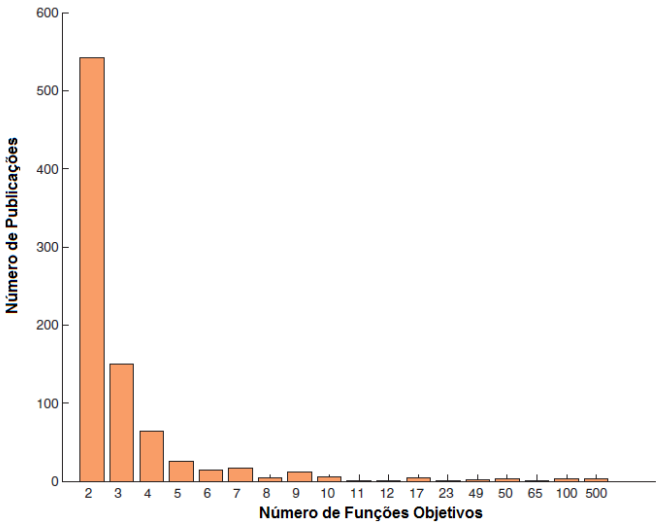


Figura 1.2 Ilustração gráfica de um problema multiobjetivo (adaptado de [CLV07]).

vestigar a escalabilidade dos MOEAs. Nesse trabalho, ele constatou a ineficácia dos principais MOEAs na solução de problemas com mais de quatro objetivos. Estudos seguintes, realizados por Purshouse e Fleming [PF03] e por Hughes [Hug05], mostraram que essa ineficácia se devia à dominância de Pareto. Farina e Amato [FA04] explicaram isso apresentando um argumento intuitivo de que a probabilidade de uma solução dominar outra diminui rapidamente com o número de objetivos. Devido a esse fato, todas as soluções da população em MOEAs se tornam não-dominadas entre si. Como consequência, todas as soluções têm a mesma importância e, portanto, qualquer solução pode guiar o processo de busca. Naturalmente, como não há preferência entre as soluções, os MOEAs baseados em dominância de Pareto se comportam como buscas aleatórias e, portanto, não há convergência em direção à Frente de Pareto. A ineficácia peculiar dos MOEAs baseados em dominância de Pareto e também de outras abordagens em problemas com mais de quatro objetivos levaram ao surgimento de uma nova linha de pesquisa denominada de Otimização de Muitos Objetivos [3] [RHH+13, GPC09]. E os problemas multiobjetivos de mais de quatro objetivo receberam a denominação de Problemas com Muitos Objetivos (**MaOP**, *Many-Objective Problem*) [CK07]. Atualmente, essa área está ativa devido principalmente à demanda por métodos eficazes capazes de lidar com um alto número de objetivos. A Figura 1.3 apresenta o número de publicações, ao longo dos anos, de MOEAs aplicados a MaOPs e MOPs. Nesse gráfico, o eixo do número de publicações está em escala logarítmica. Esse gráfico foi gerado usando a ferramenta SCOPUS [SCO13] realizando-se uma pesquisa por palavras chaves envolvendo os termos *multi-objective*, *multiobjective*, *evolutionary*, *genetic algorithm* para listar os MOEAs aplicados em problemas multiobjetivos e usando o termo *many-objective* no lugar de *multi-objective* e *multiobjective* para listar os MOEAs aplicados apenas a problemas com muitos objetivos. Como se pode notar, o método utilizado para

[3] Alguns autores [PF03, Hug05] utilizam o termo Otimização Evolucionária de Muitos Objetivos. Contudo, como existem algoritmos de outros paradigmas, como o Enxame de Partículas, que também tem sido aplicados a problemas multiobjetivos, decidiu-se por omitir o termo Evolucionário.

a geração do gráfico não apresenta nenhum rigor e portanto ele não fornece uma informação precisa do crescimento de publicações dos MOEAs. O objetivo foi apenas fornecer, de modo superficial, a diferença no número de publicações de MOEAs em MOPs e MaOPs. Tendo isso em vista, pode-se concluir, a partir do gráfico, que (1) o número de MOEAs aplicados em MaOPs vem aumentando no últimos anos, o que evidencia a atividade da área, e (2) o número de publicações de MOEAs aplicados em MOPs é muito maior que o número de MOEAs aplicados em MaOPs, o que de certo modo ratifica o gráfico da Figura 1.2.

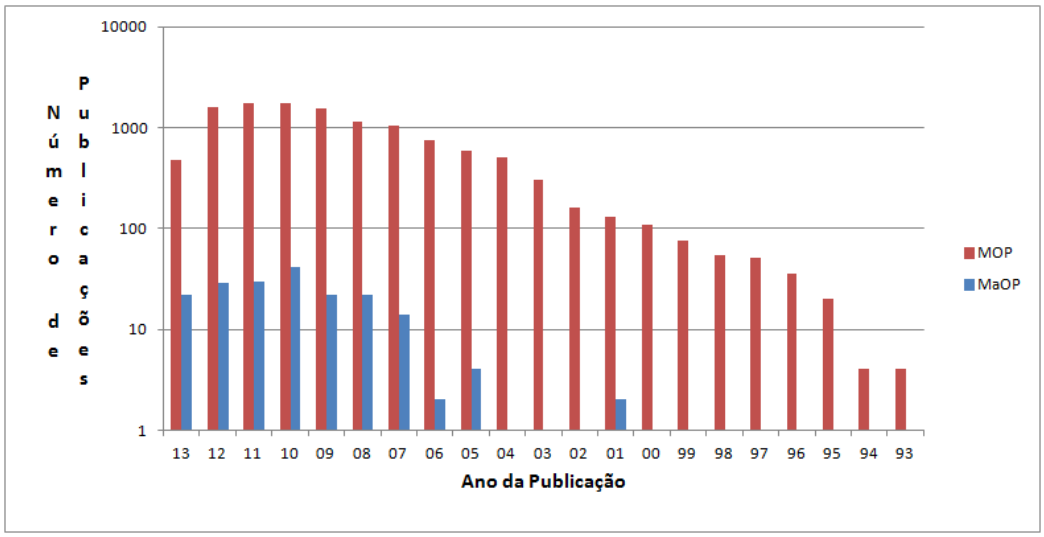


Figura 1.3 Comparativo do número de publicações com relação a aplicação de MOEAs em MOPs e MaOPs.

Devido às dificuldades da dominância de Pareto em discriminar soluções em MaOPs, os pesquisadores começaram a propor métodos de atribuição de aptidão alternativos à dominância de Pareto com o objetivo de melhorar a convergência dos MOEAs nesses problemas. Recentemente, dois MOEAs, o CEGA [GPC10] e o MDFA [GPC10], apresentaram uma boa capacidade de convergência para a Frente de Pareto e são atualmente o estado da arte na área de otimização de muitos objetivos.

Apesar desse desenvolvimento inicial dos MOEAs, pouco tem sido feito no sentido de adaptar os MOPSOs com o objetivo de melhorar a capacidade de convergência deles em problemas com muitos objetivos. É importante enfatizar que os MOPSOs são regidos por princípios e mecanismos diferentes daqueles presentes nos MOEAs [MS08], de modo que tal adaptação não é necessariamente natural. Os poucos trabalhos na literatura que realizaram essa adaptação apresentam desvantagens, tais como necessidade de informação do tomador de decisão, parâmetros difíceis de ajustar, e ainda incapacidade de alcançar a Frente de Pareto, principalmente em problemas com multimodalidade. Sendo assim, ainda existe a necessidade, devido as desvantagens dos atuais MOPSOs em lidar com MaOPs, de se desenvolver MOPSOs capazes de convergir em problemas com muitos objetivos.

1.3 Objetivos

Tendo em vista a necessidade de adaptar os MOPSOs para resolver problemas com muitos objetivos, em virtude das razões mencionadas na seção anterior, o objetivo geral desta dissertação é desenvolver um MOPSO que seja capaz de convergir nesses problemas; mesmo em problemas apresentando multimodalidade.

O objetivo específico desta dissertação é desenvolver um MOPSO com um método de atribuição de aptidão de alta capacidade discriminativa das soluções no lugar da dominância de Pareto com o objetivo de melhorar a sua convergência.

No desenvolvimento do objetivo específico, os seguintes objetivos secundários foram alcançados:

- Desenvolvimento de um *survey* sobre a área. Existe pouca literatura em português sobre o tema de otimização de muitos objetivos. Em grande parte por ele ser um campo de pesquisa recente e também pelos desafios impostos pela área. Assim, esta dissertação tem o objetivo de propagar e motivar o estudo desse tema que tem tantas aplicações práticas;

- Desenvolvimento de um *survey* sobre MOPSOs aplicados a problemas multiobjetivos. Até onde vai o conhecimento do autor, não existe na literatura, nenhum trabalho que exponha os principais avanços na área no que diz respeito aos MOPSOs em problemas com muito objetivo. O principal *survey* da área cobre apenas MOEAs [ITN08].

1.4 Estrutura da Dissertação

Esta dissertação está organizada como segue. No Capítulo 2, são apresentados os conceitos sobre otimização multiobjetiva que são fundamentais para o entendimento do resto da dissertação. Nesse capítulo também serão apresentados os conceitos básicos sobre algoritmos evolucionários, e sobre algoritmos evolucionários multiobjetivos. O Capítulo 3 apresenta o tema principal desta dissertação, ou seja, a área da Otimização de Muitos Objetivos. Nesse capítulo são apresentados os desafios acerca dessa área e os algoritmos do estado da arte para resolvê-los. O Capítulo 4 apresenta os conceitos sobre otimização multiobjetiva por enxame de partículas. Esse capítulo também apresenta as principais adaptações desses algoritmos para problemas com muitos objetivos. Nesse mesmo capítulo, a técnica proposta nesta dissertação para lidar com problemas com muitos objetivos, o MOPSO-GD, é apresentado. Em seguida, o Capítulo 6 apresenta o arranjo experimental. Posteriormente, os resultados dos experimentos são apresentados no Capítulo 7. Por fim, o Capítulo 8 apresenta a conclusão da dissertação, as considerações finais, e os trabalhos futuros.

CAPÍTULO 2

Otimização Multiobjetiva Por Algoritmos Evolucionários

" *Leiam Euler, leiam Euler...Ele é o mestre de todos nós* "
– Pierre Simon de Laplace

NSTE capítulo, são fornecidos os fundamentos para se compreender o restante desta dissertação. Para esse fim, os fundamentos de otimização multiobjetiva, de algoritmos evolucionários e de algoritmos evolucionários multiobjetivos serão apresentados sucessivamente e com detalhes suficientes para que os conceitos e algoritmos dos próximos capítulos possam se entendidos completamente. A Seção 2.1 apresenta os fundamentos de otimização multiobjetiva incluindo definições que estão presentes ao longo de toda dissertação. A Seção 2.2 apresenta uma breve introdução sobre algoritmos evolucionários em que conceitos como cruzamento, mutação e seleção são apresentados. Esses conceitos são importantes pois eles são fundamentais para se entender alguns dos algoritmos utilizados nesta dissertação. Posteriormente, serão apresentados a classe de algoritmos evolucionários que são aplicados à problemas multiobjetivos: os algoritmos evolucionários multiobjetivos. Essa seção é importante para que se possa entender as desvantagens desses algoritmos quando eles são aplicados em problemas com muitos objetivos. Em particular, eles são apresentados por razões históricas, uma vez que os primeiros estudos sobre as desvantagens da dominância de Pareto em problemas com muitos objetivos iniciaram na perspectiva desses algoritmos. Por fim, na Seção 2.4 são apresentadas as considerações finais.

2.1 Otimização Multiobjetiva

Um problema de otimização multiobjetivo ou simplesmente problema multiobjetivo (**MOP**, *Multiobjective Problem*) pode ser definido por um vetor de funções **f** que mapeia um vetor de n parâmetros (variáveis de decisão) para um vetor com m funções objetivos, com $m \geq 2$. Formalmente, esse problema pode ser definido, sem perda de generalidade [1], como:

[1] Nesta dissertação, a minimização de todos os objetivos é considerada em todas as definições.

$$\text{minimizar } \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}, \tag{2.1}$$

sujeito à

$$\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), ..., h_p(\mathbf{x})] = \mathbf{0}, \tag{2.2}$$

e

$$\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), ..., g_q(\mathbf{x})] \leq \mathbf{0}, \tag{2.3}$$

em que $\mathbf{x} = [x_1, x_2, ..., x_n]^T \in S \subset \mathbb{R}^n$ é o vetor de variáveis de decisão (ou, simplesmente vetor de decisão), $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$ são as funções objetivos e $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$ são, respectivamente, as inequações e equações que modelam as restrições do problema. Os vetores $\mathbf{y} = \mathbf{f}(\mathbf{x}) \forall \mathbf{x} \in \mathbb{R}^m$ são chamados de vetores objetivos. O conjunto $\mathbb{R}^n$ é chamado de espaço de variáveis de decisão ou espaço de decisão e o conjunto $\mathbb{R}^m$ é chamado de espaço objetivo. O conjunto $S \subset \mathbb{R}^n$ é o espaço de busca que corresponde ao conjunto de possíveis soluções para o problema [Coe99].

Neste ponto, é importante definir o significado da palavra minimizar no contexto de um problema de otimização multiobjetivo. Para isso, algumas definições são necessárias.

Definição 2.1 (**Conjunto Factível**). O conjunto factível $\Omega \subset \mathbb{R}^n$ é definido como o conjunto de vetores de decisão $\mathbf{x}$ que satisfazem as restrições g_i, h_j para $i = 1, 2, \ldots, p$ e $j = 1, 2, \ldots, q$. Ou seja,

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \text{ e } \mathbf{h}(\mathbf{x}) = \mathbf{0}\}. \tag{2.4}$$

A imagem de Ω no espaço objetivo é denominada de região factível e é denotada por $\mathscr{O}$. A Figura 2.1 ilustra a região factível de um problema multiobjetivo com dois objetivos. É importante destacar que $\Omega \subset S$ [Coe99]. A Figura 2.1 também apresenta todos os conceitos já apresentados. Na Figura 2.1, o gráfico da esquerda mostra o espaço de decisão $\mathbb{R}^n$ com a região factível Ω. Nessa figura, o gráfico da direita mostra o espaço objetivo. Essa figura também mostra o mapeamento entre um vetor no espaço de decisão para um vetor objetivo.

Definição 2.2 (**Dominância Fraca de Pareto**). Dados dois vetores $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, diz-se que o vetor $\mathbf{u}$ domina fracamente o vetor $\mathbf{v}$, denotado por $\mathbf{u} \preceq \mathbf{v}$, se e somente se, $u_i \leq v_i$, $\forall i \in \{1, 2, \ldots, m\}$ [Aze11, ZTL$^+$03].

Definição 2.3 (**Dominância de Pareto**). Dados dois vetores $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, diz-se que o vetor $\mathbf{u}$ domina o vetor $\mathbf{v}$, denotado por $\mathbf{u} \prec \mathbf{v}$, se e somente se, $u_i \leq v_i$, $\forall i \in \{1, 2, \ldots, m\}$ e se $\exists i \in \{1, 2, \ldots, m\}$ tal que $u_i < v_i$ [Aze11, ZTL$^+$03].

Definição 2.4 (**Dominância Forte de Pareto**). Dados dois vetores $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, diz-se que o vetor $\mathbf{u}$ domina estritamente (fortemente) o vetor $\mathbf{v}$, denotado por $\mathbf{u} \prec\prec \mathbf{v}$, se e somente se, $u_i < v_i, \forall i \in \{1, 2, \ldots, m\}$ [ZTL$^+$03].

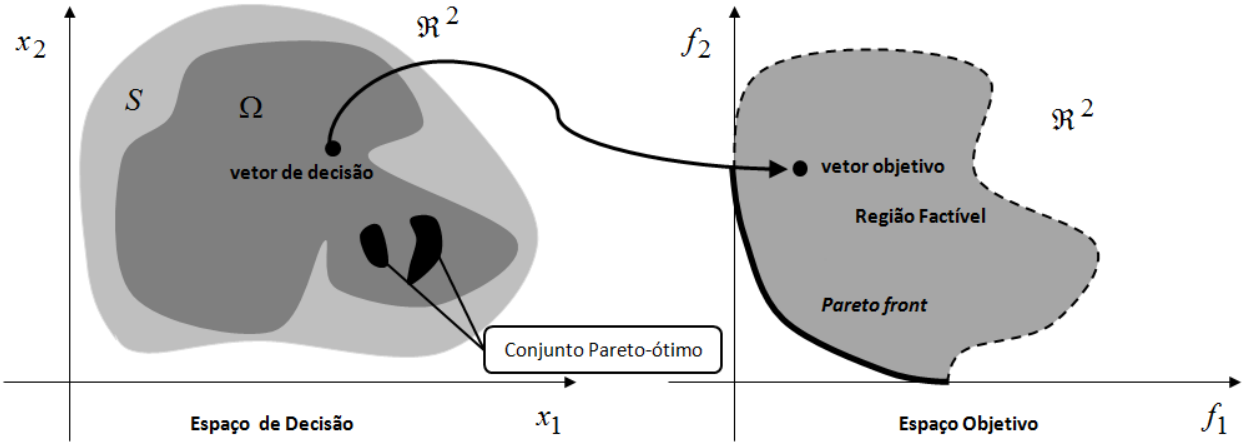


Figura 2.1 Relação entre o espaço de decisão e o espaço objetivo.

Definição 2.5 (**Soluções incomparáveis**). Dados dois vetores $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, diz-se que o vetor $\mathbf{u}$ é incomparável ao vetor $\mathbf{v}$, denotado por $\mathbf{u} \,||\, \mathbf{v}$, se e somente se, $\mathbf{u}$ não domina fracamente $\mathbf{v}$ e $\mathbf{v}$ não domina fracamente $\mathbf{u}$ [ZTL+03].

A Figura 2.2 apresenta a interpretação geométrica das possíveis relações de dominância existentes entre as soluções no espaço objetivo bidimensional. Uma solução A divide esse espaço em quatro regiões retangulares. As soluções na região retangular acima e a direita de A representam as soluções dominadas pela solução **A**. As soluções que estão na região retangular abaixo e a esquerda de A representam as soluções que dominam a solução A. Finalmente, as soluções que estão nas duas regiões retangulares restantes são soluções incomparáveis a solução A.

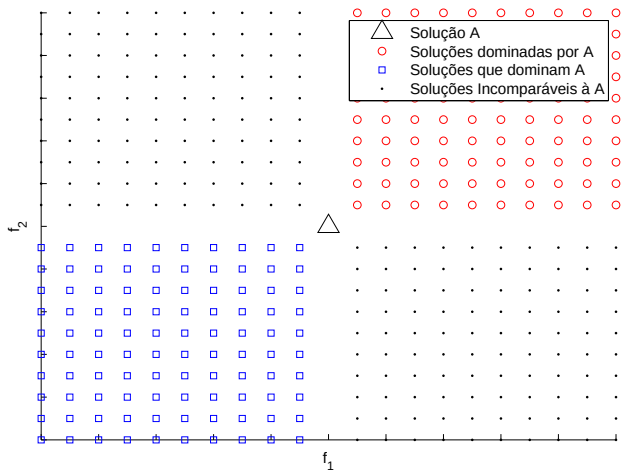


Figura 2.2 Relações possíveis entre as soluções no espaço objetivo.

Uma vez que o conceito de dominância de Pareto foi estabelecido, pode-se definir o conceito de otimalidade de Pareto conforme:

Definição 2.6 (**Otimalidade de Pareto**). Um vetor de variáveis de decisão $\mathbf{x}^* \in \Omega \subset \mathbb{R}^n$ é uma solução Pareto-ótima, ou uma solução eficiente, se e somente se, $\nexists \mathbf{x} \in \Omega$ tal que $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.

Definição 2.7 (**Conjunto Pareto-ótimo**). O conjunto Pareto-ótimo Ω^* é definido por:

$$\Omega^* = \{\mathbf{x} \in \Omega \,|\, \mathbf{x} \text{ é Pareto-ótimo}\}. \tag{2.5}$$

A Figura 2.1 mostra um exemplo de conjunto Pareto-ótimo disjunto.

Definição 2.8 (**Frente de Pareto global**). A Frente de Pareto global $\mathscr{PF}^*$ ou simplesmente Frente de Pareto é definida por:

$$\mathscr{PF}^* = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \,|\, \mathbf{x} \in \Omega^*\}. \tag{2.6}$$

Os vetores pertencentes à Frente de Pareto global são ditos serem vetores não-dominados.

Definição 2.9 (**Conjunto Pareto-ótimo local**). O conjunto Ω' é denotado de conjunto Pareto-ótimo local [Zit99] se e somente se $\forall \mathbf{x}' \in \Omega'$, $\nexists \mathbf{x} \in \Omega$ tal que $\mathbf{x} \prec \mathbf{x}'$ e $\left\|\mathbf{x} - \mathbf{x}'\right\| < \varepsilon$ e $\left\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}')\right\| < \delta$. em que $\|\cdot\|$ é uma métrica de distância e $\varepsilon > 0$, $\delta > 0$.

Definição 2.10 (**Frente de Pareto local**). Um conjunto $\mathscr{PF}'$ é uma Frente de Pareto local correspondente a um conjunto Pareto-ótimo local Ω' se e somente se

$$\mathscr{PF}' = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \,|\, \mathbf{x} \in \Omega'\}. \tag{2.7}$$

A Figura 2.3 apresenta exemplos de uma Frente de Pareto global e local em um espaço objetivo bidimensional. Cabe ressaltar que os conceitos de Frente de Pareto global e local equivalem aos conceitos de valor ótimo global e local em problemas mono-objetivo (otimização global).

Uma vez que as definições necessárias acerca de otimização multiobjetiva já foram apresentadas, pode-se agora definir no que consiste resolver um problema de otimização multiobjetivo.

Definição 2.11 (**Solução de um problema multiobjetivo**). Resolver um MOP consiste em determinar o conjunto Pareto-ótimo Ω^* a partir do conjunto Ω.

2.1.1 Algoritmo de Otimização Multiobjetivo

Um algoritmo cujo objetivo é resolver um MOP é denominado de algoritmo multiobjetivo. Como a Frente de Pareto global de um MOP $\mathscr{PF}^*$ pode conter um número enorme ou mesmo infinito de soluções Pareto-ótimas [Pur03], a tarefa de um algoritmo multiobjetivo é, em geral, fornecer uma aproximação desse conjunto [Gar09, Pur03] devido a limitações de recursos computacionais. Além disso, outro motivo para se fornecer uma aproximação da Frente de Pareto (ou do conjunto Pareto-ótimo) é a complexidade subjacente do problema que impede

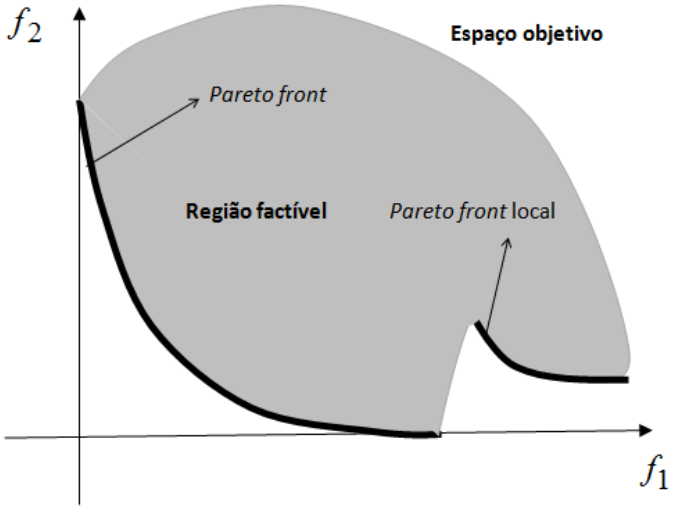


Figura 2.3 Ilustração de uma Frente de Pareto global e local.

a sua solução por métodos exatos [ZTL+03]. A aproximação de $\mathscr{PF}^*$ gerado por algum algoritmo multiobjetivo é denominado de *Conjunto aproximação* [ZTL+03, Gar09, CLV07] e é denotado, nesta dissertação, por PF. Em outras palavras, a saída produzida por um algoritmo multiobjetivo consiste em um conjunto de soluções incomparáveis, ou formalmente:

Definição 2.12 (Conjunto aproximação). Seja $PF \subseteq \mathscr{O}$ um conjunto de vetores objetivos, o conjunto PF é um conjunto aproximação se qualquer vetor de PF não domina fracamente qualquer outro vetor em PF [Gar09, ZTL+03].

Mais precisamente, o que se deseja obter de um algoritmo multiobjetivo consiste no conjunto de soluções no espaço de decisão que correspondem ao conjunto PF. Entretanto, nesta dissertação, o conjunto aproximação será considerado como a saída final de um algoritmo multiobjetivo.

Além das limitações de recursos computacionais, outra razão para gerar uma aproximação da Frente de Pareto é que o tomador de decisão é capaz de analisar apenas um conjunto limitado de soluções devido a limitações de tempo [LKC+12]. O *tomador de decisão* (**DM**, *Decision Maker*) é a pessoa ou conjunto de pessoas que se assume ter amplo conhecimento do problema e que é capaz de impor preferências entre as diferentes soluções de modo a tomar uma decisão final [Gar09].

A qualidade do conjunto aproximação PF produzido por um algoritmo pode ser avaliada segundo dois aspectos [2] [Kha02, LKC+12]:

- **Proximidade.** O conjunto aproximação PF deveria conter apenas vetores objetivos próximos da Frente de Pareto. Idealmente, todos os seus vetores deveriam pertencer também à Frente de Pareto;

[2] Em [Pur03], Purshouse destacou um outro aspecto além da proximidade e diversidade que é a pertinência. No entanto, nesta dissertação apenas a proximidade e a convergência são considerados.

- **Diversidade.** O requisito de diversidade normalmente diz respeito às soluções no espaço objetivo. No espaço objetivo, o conjunto aproximação *PF* deveria conter uma boa distribuição de soluções em termos de extensão e uniformidade, ou seja, o conjunto aproximação deveria cobrir a Frente de Pareto inteira e apresentar soluções uniformemente espalhadas sobre ela.

A Figura 2.4 ilustra um exemplo de conjunto aproximação ideal gerado por um algoritmo multiobjetivo. Nessa figura, todas as soluções estão sobre a Frente de Pareto. Além disso, elas cobrem toda a extensão da Frente de Pareto e estão uniformemente distribuídas sobre ela.

Por fim, para encerrar essa seção de definições, é importante enfatizar que todas as considerações, definições, equações e observações que serão apresentadas daqui em diante irão assumir a minimização de todas as funções objetivos.

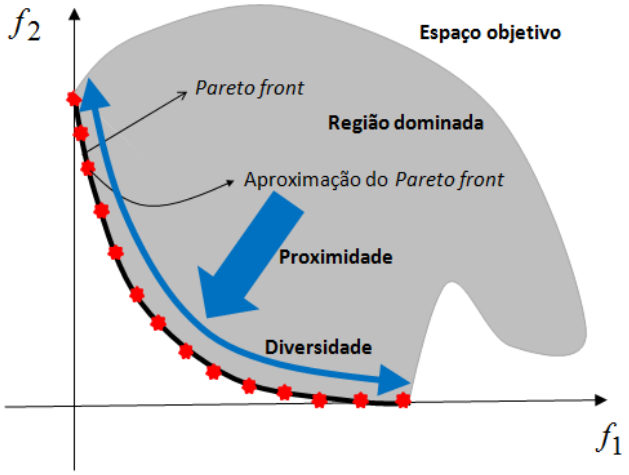


Figura 2.4 Requisitos desejáveis do conjunto aproximação de um algoritmo multiobjetivo.

2.2 Algoritmos Evolucionários

A evolução biológica pode ser pensada de modo simples como mudanças na forma e comportamento de organismos vivos (indivíduos) dentro de uma população entre gerações [Rid04]. Os princípios de funcionamento da evolução biológica são explicados pela teoria da seleção natural proposta por Charles Darwin em 1859 [Dar59]. A teoria Darwiniana da evolução diz que a evolução biológica ocorre por meio de três princípios:

1. O primeiro princípio diz que os indivíduos em uma população apresentam uma grande variedade de características entre si. Considere, por exemplo, a altura dos indivíduos dentro de uma população. Então, pode-se observar que os indivíduos dessa população apresentam diferentes alturas [Rid04];

2. O segundo princípio afirma que os indivíduos podem transmitir suas características aos seus descendentes por meio da reprodução (hereditariedade) [Rid04];

3. O terceiro principio declara que, em um mundo com recursos limitados tais como alimento e espaço, e com populações estáveis [Rid04, Eng07], os indivíduos competem entre si, em um sentido ecológico, para sobreviver e se reproduzir. Darwin denominou esse fenômeno de luta para existência. Nessa competição, os indivíduos com as características mais desejáveis, isto é, com as características que os tornam mais bem adaptados às condições ambientais, têm maiores chaves de sobreviver e se reproduzir [Rid04].

Desta forma, os indivíduos mais adaptados ao ambiente tendem a produzir mais descendentes que os outros indivíduos da população. A esse fenômeno dá-se o nome de seleção natural [Rid04]. As características desejáveis são então passadas para os descendentes e ao longo de várias gerações elas se tornam dominantes na população.

Na apresentação de sua teoria, Darwin constatou que os indivíduos apresentavam uma grande variabilidade em suas características, mas não explicou porque isso acontecia. Além disso, ele não apresentou uma boa explicação para a hereditariedade das características. Explicações plausíveis para esses dois fenômenos só foram possíveis com o desenvolvimento da genética Mendeliana. Isso levou ao surgimento da teoria neo-Darwinista da evolução. Essa teoria incorpora a teoria da seleção natural para explicar o processo de adaptação dos organismos juntamente com a teoria genética Mendeliana para explicar a variabilidade entre os indivíduos e a hereditariedade das características. Segundo a teoria neo-Darwinista, a variação entre os indivíduos se dá por meio da recombinação (em populações sexuadas) e mutação [Rid04].

Nessa perspectiva, a teoria da seleção natural tem dois aspectos importantes. O primeiro diz respeito aos processos que levam a **variação** entre os indivíduos que consistem na recombinação e mutação. E o segundo aspecto é o fato de que os indivíduos mais adaptados ao ambiente têm maiores chances de se reproduzir e sobreviver, isto é, a **seleção** natural.

À luz da teoria da seleção natural de Darwin, a evolução biológica pode ser pensada como um processo de otimização no sentido de que ela leva os indivíduos de uma população a se tornarem cada vez mais adaptados ao seu ambiente [Eng07]. Adaptação se refere às características que aumentam as chances de um indivíduo sobreviver e se reproduzir em um determinado ambiente [Rid04]. O processo adaptativo produz indivíduos bem projetados (bem ajustados) ao seu ambiente. Essa característica da evolução faz dessa teoria uma fonte de inspiração para a criação de algoritmos de busca (otimização). Nesse sentido, **Algoritmo Evolucionário** (**EA**, *Evolutionary Algorithm*) é qualquer algoritmo estocástico de busca inspirado nos conceitos da teoria neo-Darwinista da evolução [MF00, Eng07]. Os EAs inicialmente foram aplicados em problemas mono-objetivos. A motivação para usá-los se deve ao fato deles oferecerem várias vantagens em relação aos métodos clássicos de otimização, tais como: eles não necessitam de informação de gradiente [Van06], o que é importante no caso em que a função objetivo é não-diferenciável; conseguem fornecer boas soluções em problemas nos quais os métodos tradicionais falham, isto é, em problemas difíceis com, por exemplo, descontinuidades, multimodalidade e espaços de buscas enormes; e são conceitualmente simples [Adr07].

A ideia fundamental de um EA é iterativamente aplicar processos de *variação* e *seleção* em um conjunto, denominado de população, de soluções candidatas do problema até que um crité-

rio de parada seja satisfeito [Pur03]. Por meio da combinação desses dois processos, variação e seleção, espera-se que as soluções se tornem cada vez melhores (mais bem projetadas) ao longo de sucessivas iterações. Ou seja, espera-se que as soluções sejam cada vez mais próximas da solução ótima do problema. É importante assinalar que não há nenhuma garantia de que essa solução seja encontrada. No entanto, o poder desses algoritmos está em sua capacidade de fornecer boas aproximações para essas soluções [Pur03, Van06]. O princípio de funcionamento de um EA pode ser descrito pela equação (2.8) em que $P[t]$ é a população na geração atual t [MF00, Pur03]. Cada iteração de um EA recebe o nome de geração para enfatizar a inspiração biológica desses algoritmos.

$$P[t+1] = s(v(P[t])). \tag{2.8}$$

Como se observa na equação (2.8), um EA aplica processos de variação v, que consistem em mudanças probabilísticas que simulam a recombinação e a mutação em organismos vivos, na população da geração atual $P[t]$ de soluções candidatas, produzindo um conjunto de novas soluções candidatas $v(P[t])$. Cada nova solução, por sua vez, é selecionada, usando os processos de seleção s, para participar da geração seguinte $P[t+1]$ com base em sua habilidade/aptidão de resolver o problema. Os processos de variação e seleção são então aplicados nessa nova população, e esse processo é continuamente repetido até um critério de parada seja atingido [Pur03, BHS97].

O mecanismo de funcionamento de um EA pode ser expandido para representar um EA mais genérico. De acordo com Purshouse [Pur03] a estrutura geral de um EA pode ser descrita pela equação (2.9). Nessa nova equação, existem dois tipos de seleção: a seleção para variação (s_v) e a seleção para sobrevivência (s_s). A seleção para variação é o estágio no qual os indivíduos são escolhidos para participarem do processo de variação. Diferentemente, a seleção para sobrevivência é o estágio no qual as soluções produzidas no estágio de variação juntamente com as soluções da população da geração atual $P[t]$ são selecionadas para formarem a próxima geração $P[t+1]$ [Pur03].

$$P[t+1] = s_s(v(s_v(P[t])), P[t]). \tag{2.9}$$

Existem várias maneiras pelas quais os processos de seleção para variação, variação e seleção para sobrevivência podem ser formuladas. Dessa forma, existem várias formulações possíveis de EAs. Na literatura, destacam-se os Algoritmos Genéticos, as Estratégias de Evolução, a Evolução Diferencial e a Programação Genética [Van06, Pur03, BHS97].

Na próxima seção, os conceitos básicos e a terminologia adotada pelos EAs serão discutidos. Além disso, os estágios de seleção para variação, variação e seleção para sobrevivência serão explicados em mais detalhes.

2.2.1 Conceitos Básicos e Terminologia

Como foi explicado anteriormente, os organismos têm certas características que influenciam sua habilidade de sobreviver e reproduzir. Essas características são codificadas em longas cadeias de informação denominadas de DNA (*Deoxyribose Nucleic Acid*). O DNA consiste em uma molécula composta for duas fitas entrelaçadas em forma de dupla hélice, cada fita sendo

composta por unidades denominadas de nucleotídeos. Essa molécula foi descoberta por Watson e Crick em 1953 [Rid04].

O DNA provê um mecanismo físico de hereditariedade em quase todos os organismos vivos [Rid04]. O DNA carrega a informação para construir um novo organismo. Dentro do núcleo de células eucariontes, o DNA é carregado em estruturas denominadas de *cromossomo*. O DNA contém uma grande quantidade de *genes*, em que o gene é a menor unidade de hereditariedade. Os *genes* determinam muitos aspectos da anatomia e do comportamento dos organismos por meio do controle da produção de proteínas. Qualquer gene é localizado em um lugar específico no cromossomo chamado de *locus gênico*. As diferentes formas com que um gene pode se manifestar em um *locus* são chamados de *alelos* [Rid04].

Muita da terminologia usada para descrever um EA é emprestada da biologia evolutiva. Assim, em um EA, cada solução candidata de um problema de otimização representa um *indivíduo*[3]; e as variáveis de decisão desse indivíduo são codificadas em um *cromossomo*. Geralmente, esse cromossomo consiste em um vetor cujos componentes podem assumir valores de algum domínio; e cada componente desse vetor corresponde a um *gene*. Os valores possíveis que podem ser atribuídos a um gene a partir de um determinado domínio são chamados de *alelos*. O *locus* corresponde à posição do gene no cromossomo. Por fim, um conjunto de indivíduos formam uma *população* [Eng07].

Assim como ocorre na natureza, em um EA, o indivíduo tem dois níveis de informação: o *genótipo* e o *fenótipo*. O genótipo descreve a composição genética do indivíduo, ou seja, ele descreve quais alelos o indivíduo possui. A importância do genótipo é que é nele em que os processos de variação (cruzamento e mutação) ocorrem. O *fenótipo*, por sua vez, representa a solução do problema em si associada ao indivíduo. Isto é, o fenótipo corresponde a decodificação do genótipo para produzir a solução do problema. Para uma melhor explicação da relação entre fenótipo e genótipo o leitor pode consultar [BHS97, Van06].

Um outro conceito importante em EA é o de *aptidão* (*fitness*, em inglês). No modelo neo-Darwinista de evolução, indivíduos com as melhores características, isto é, os indivíduos mais adaptados ao ambiente têm maiores chances de sobreviver e se reproduzir (sobrevivência dos mais aptos). No contexto de EAs, a aptidão do indivíduo consiste em um valor geralmente proveniente da função objetivo que mede a qualidade desse indivíduo com relação ao resto da população. Quanto maior esse valor, melhor é o indivíduo com relação aos outros indivíduos. A aptidão do indivíduo é importante, pois ela é utilizada nos estágios de seleção para sobrevivência e variação, de modo que quanto maior for a aptidão de um indivíduo maiores são as suas chances de se reproduzir e de participar da geração seguinte. A partir disso, pode-se dizer que a função objetivo assume o papel do ambiente em EA.

Outro ponto importante do projeto de um EA é a definição da população inicial de indivíduos. O modo padrão de gerar uma população inicial em um espaço de busca é realizar uma amostragem segundo uma função de distribuição uniforme sobre o domínio permitido em cada gene no cromossomo. Usando essa forma de inicialização, pode-se assegurar que a população inicial consiste em uma representação uniforme de todo o espaço de busca. Com isso, evitam-se que regiões do espaço de busca sejam ignoradas pelo EA durante o processo de busca [Eng07].

Para exemplificar, os conceitos apresentados, considere o problema de minimizar a função

[3] Ao longo dessa dissertação, o termo indivíduo e solução serão utilizados intercambiavelmente.

bidimensional $f(x_1,x_2) = x_1^2 + x_2^2$. Considere a representação binária utilizando os símbolos 0 e 1. Suponha que cada variável do problema, x_1 e x_2, seja codificada como vetores de cinco componentes cujos valores são binários. Assim, o cromossomo de cada solução candidata $\mathbf{x} = [x_1,x_2]$ consiste em um vetor com dez componentes que assumem valores do conjunto $\{0,1\}$. A Figura 2.5 apresenta uma população de sete cromossomos. Cada componente em um cromossomo representa um gene, e cada um dos valores que o gene pode assumir representa um alelo. No exemplo, os alelos são os valores 0 e 1. Nessa figura, também fica evidente a diferença entre genótipo e fenótipo. Enquanto o genótipo, nesse exemplo, é a codificação binária da solução, o fenótipo é a solução em si $[21,18]$. Mais ainda, caso a função $f(x_1,x_2)$ representasse o custo de uma ponte de comprimento x_1 e altura x_2, o fenótipo da solução no exemplo dado corresponderia a instância de uma ponte com altura 21 e altura 18. A Figura 2.5 apresenta também a aptidão dos indivíduos.

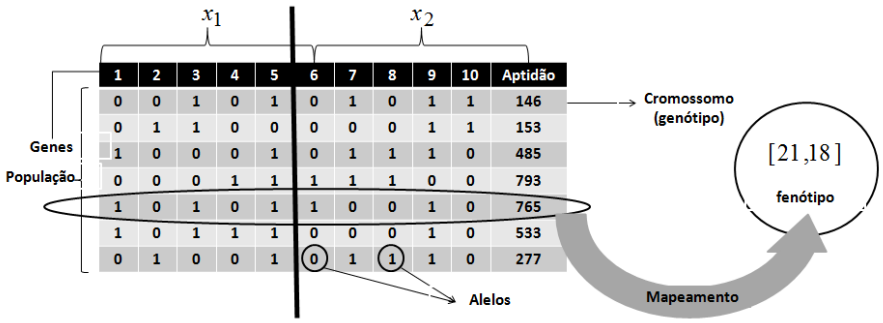


Figura 2.5 População de cromossomos em representação binária.

A representação real também é possível. A Figura 2.6 apresenta a representação real dos cromossomos da Figura 2.5. Como se observa na Figura 2.6, os próprios valores da variáveis de decisão são usados nos cromossomos dos indivíduos. Isso ocorreu porque a função $f(x_1,x_2)$ é uma função matemática de modo que o mapeamento foi direto, mas isso nem sempre é o caso. Novamente, o fenótipo da solução nesse exemplo corresponderia a instância de uma ponte com altura 21 e altura 18.

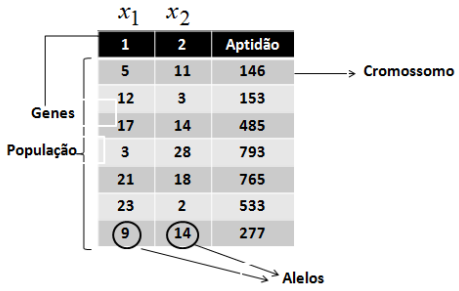


Figura 2.6 População de cromossomos em representação real.

2.2.2 Seleção para Variação

Nesse estágio, soluções da população da geração atual $P[t]$ são escolhidos para serem incluídos no chamado *mating pool*, no qual os processos de variação serão realizados para produzir novas soluções. O *mating pool* é uma população temporária criada para armazenar os indivíduos que participarão da etapa de variação.

No processo de seleção para variação, os indivíduos competem entre si para se reproduzirem com base em seu valor de aptidão, de modo que os que apresentam maior aptidão têm maiores chances de se reproduzir. Desta forma, a seleção para variação tende a enfatizar as melhores soluções. A seleção para variação é realizada por meio dos operadores de seleção. Um operador de seleção é um procedimento que escolhe indivíduos da população por meio da comparação das aptidões entre os indivíduos dessa população [BHS97]. Vários operadores de seleção têm disso propostos na literatura, tais como: seleção por torneio e a seleção proporcional [Eng07]. A seleção por torneio binário, por exemplo, seleciona dois indivíduos aleatoriamente da população. Esses dois indivíduos são então comparados entre si, e o individuo com maior aptidão é retornado pelo operador [Eng07, BHS97]. Já a seleção proporcional é um método de seleção que enviesa a seleção para os indivíduos com maior aptidão [Eng07]. Uma das formas de seleção proporcional é a seleção por roleta [4]. O pseudo-código do mecanismo de seleção por roleta é apresentado em Algoritmo 1.

Algoritmo 1: Pseudo-código da Seleção por Roleta.

```
Calcule a probabilidade p_i do indivíduo i ser selecionado;
Classifique as soluções de acordo com a probabilidade p_i em ordem decrescente;
Faça i = 1;
soma = p_i;
Escolha r ~ U(0,1);
enquanto soma < r faça
    i = i + 1;
    soma = soma + p_i;
Retorne x_i.
```

Antes de se aplicar a seleção por roleta, é necessário, primeiramente, calcular a probabilidade de um indivíduo ser selecionado. Para isso, é utilizada a equação (2.10):

$$p_i = \frac{aptidao(\mathbf{x}_i)}{\sum_{j=1}^{|P|} aptidao(\mathbf{x}_j)}, \tag{2.10}$$

em que p_i é a probabilidade do indivíduo $\mathbf{x}_i$ ser selecionado, $|P|$ é o número de indivíduos na população, e $aptidao(\mathbf{x}_i)$ é a aptidão do indivíduo $\mathbf{x}_i$. Em seguida, os indivíduos são classificados de acordo com a sua probabilidade em ordem decrescente. Posteriormente, um número r é

[4] A seleção por roleta requer que as aptidões sejam positivas [BHS97]. Além disso, ela é aplicada, em princípio, em problemas de maximação uma vez que ela favorece soluções com maior aptidão [BHS97]. Entretanto, todo problema de minimização pode ser convertido em um problema de maximização [Rao09].

retirado de uma distribuição de probabilidade uniforme ($r \sim U(0,1)$). A seleção então ocorre de forma similar a uma roleta de cassino. Nesse contexto, pode-se imaginar a roleta como sendo o intervalo $[0,1]$, o qual é dividido em $|P|$ setores de tamanho p_i. Assim, o intervalo $[0,1]$ é dividido nos seguintes intervalos: $[0,p_1)$, $[p_1,p_1+p_2)$, $[p_1+p_2,p_1+p_2+p_3)$, e assim sucessivamente. Nesse exemplo é assumido que $p_1, p_2, \ldots, p_{|P|}$ estão classificados em ordem decrescente. Exposto dessa forma, o indivíduo que será escolhido é aquele cujo intervalo o ponto r pertence. Por exemplo, se o ponto $r \in [0,p_1]$ então o indivíduo de índice 1 será selecionado. Caso $r \in [p_1,p_1+p_2]$ então o indivíduo de índice 2 será escolhido. O Algoritmo 1 detalha o que foi exposto de modo mais preciso.

2.2.3 Variação

Variação é o processo explorativo pelo qual novos indivíduos são introduzidos na população. É por meio dos processos de variação que ocorrem a exploração de novas regiões no espaço de busca e o processo de otimização se desenvolve [Adr07]. Sem variação, o processo de seleção levaria a convergência prematura do algoritmo para uma solução sub-ótima devido à falta de diversidade [Eng07, Adr07]. Cruzamento (ou recombinação) e mutação são os dois componentes que compõem o processo de variação em um EA [BHS97, Eng07]. O mecanismo de funcionamento desses componentes é similar à sua contrapartida biológica. Desse modo, a reprodução envolve geralmente a troca de material genético entre dois ou mais indivíduos que são transmitidos para os descendentes produzidos, enquanto a mutação corresponde a alterações aleatórias no material genético de um indivíduo [Eng07]. Cada um desses componentes é explicado nas próximas sub-seções.

2.2.3.1 Operadores de Cruzamento

Os operadores de cruzamento são procedimentos responsáveis por criar um ou mais indivíduos por meio da combinação de material genético aleatoriamente selecionado de dois ou mais indivíduos [BHS97, Adr07]. Os indivíduos nos quais os operadores de cruzamento são aplicados são aqueles que são selecionados na etapa de seleção para variação, e esses indivíduos são chamados de pais. E os indivíduos produzidos por esses operadores são chamados de descendentes ou filhos.

Os operadores de cruzamento são, em geral, aplicados probabilisticamente de modo que eles não são aplicados necessariamente em todos os pais [BHS97]. Assim, um operador de cruzamento é aplicado em cada par (ou grupo) de pais com uma probabilidade p_c [BHS97, Eng07]. A probabilidade p_c é chamada probabilidade de cruzamento ou taxa de cruzamento [Eng07]. O valor adequado para essa probabilidade depende do problema a ser resolvido [Pur03]. Entretanto, valores altos de probabilidade são geralmente escolhidos [Pur03].

Esses operadores atuam sobre cromossomos dos indivíduos, de modo que eles são divididos de acordo com a representação dos cromossomo sobre os quais eles são aplicados. Assim, por exemplo, existem operadores de cruzamento para representações binárias e representações reais [Eng07]. Um exemplo de operador de cruzamento para representação binária é o cruzamento em um único ponto [Eng07, BHS97]. Nesse operador, dois cromossomos trocam fragmentos de seu cromossomo a partir de um gene escolhido aleatoriamente. O funcionamento

desse operador é ilustrado na Figura 2.7.

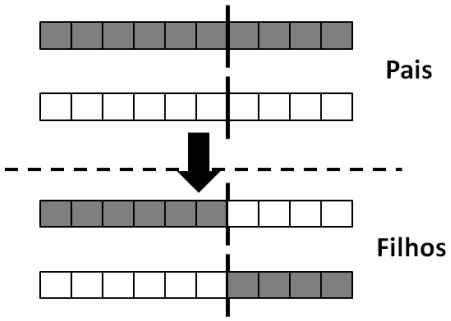


Figura 2.7 Funcionamento do operador de cruzamento em um único ponto.

O equivalente do cruzamento binário em um único ponto para cromossomos com representação real é o operador proposto por Deb e Agrawal [DA95] denominado cruzamento binário simulado (**SBX**, *Simulated Binary Crossover*) . O poder de busca do SBX é similar ao do cruzamento binário em um único ponto. O poder de busca de operador de cruzamento é definida como a probabilidade de criar uma solução filha arbitrária a partir de suas soluções pais preestabelecidas [DA95]. Para explicar o operador SBX, considere o caso unidimensional. Fazendo essa consideração, o operador SBX pode ser explicado como segue. Sejam dois pais, $p_1, p_2 \in \mathbb{R}$, então os filhos $c_1, c_2 \in \mathbb{R}$ produzidos são dados pela equação (2.11)

$$\begin{aligned} c_1 &= \frac{1}{2}[(p_1+p_2) - \beta^{'}|p_2-p_1|], \\ c_2 &= \frac{1}{2}[(p_1+p_2) + \beta^{'}|p_2-p_1|], \end{aligned} \tag{2.11}$$

com

$$\beta^{'} = \begin{cases} (2u)^{\frac{1}{\eta_c+1}} & \text{se } u \leq \frac{1}{2}, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}} & \text{caso contrário,} \end{cases} \tag{2.12}$$

em que $u \sim U(0,1)$, e $\eta_c > 0$ é o índice da distribuição do operador de cruzamento.

É importante destacar que o operador SBX exposto acima assume que não há nenhuma restrição sobre os valores de c_1 e c_2 produzidos; em outras palavras, $c_1, c_2 \in [-\infty, +\infty]$. Para calcular as soluções filhas em problemas em que o limite superior x_{max} e inferior x_{min} de uma variável de decisão são especificados, Deb [Deb00] modificou a equação (2.12) conforme a equação (2.13)

$$\beta^{'} = \begin{cases} (\alpha u)^{\frac{1}{\eta_c+1}} & \text{se } u \leq \frac{1}{\alpha}, \\ \left(\frac{1}{2-\alpha u}\right)^{\frac{1}{\eta_c+1}} & \text{caso contrário,} \end{cases} \tag{2.13}$$

em que $\alpha = 2 - \beta^{-(\eta_c+1)}$ e β é calculado como segue:

$$\beta = 1 + \frac{2}{c_2 - c_1} \min[(p_1 - x_{min}), (x_{max} - p_2)]. \tag{2.14}$$

É assumido aqui que $p_1 < p_2$. Uma simples modificação da equação acima pode ser feita para $p_1 > p_2$ [Deb00]. O procedimento acima faz com que haja uma probabilidade zero de criar uma solução filha fora do intervalo prescrito $[x_{min}, x_{max}]$.

O operador SBX gera descendentes simetricamente aos pais o que previne qualquer tipo de viés para qualquer um dos pais. Ou seja, uma vez que um filho é gerado, o segundo filho é gerado simetricamente ao primeiro em relação ao ponto central entre os dois pais [Pur03]. Para grandes valores de η_c existe uma alta probabilidade de que os filhos sejam criados próximos aos pais. Para valores pequenos de η_c, os filhos serão criados mais distantes dos pais.

Para o caso multidimensional, o procedimento descrito acima é aplicado variável a variável com uma probabilidade igual a $\frac{1}{2}$. Isto significa que para cada variável, amostra-se um valor r de uma função de distribuição uniforme $U(0,1)$. Se $r < \frac{1}{2}$ o procedimento acima de cruzamento é realizado, caso contrário, o valores dos pais são passados diretamente para os filhos.

2.2.3.2 Operadores de Mutação

Durante a reprodução de um célula, o seu DNA é fisicamente replicado para a célula filha. Em geral, uma cópia exata do DNA parental é produzida, entretanto erros podem ocorrer durante o processo de replicação, e uma versão modificada do DNA parental é transmitida para a célula filha. Esses erros na cópia do DNA são chamados de *mutação* [Rid04]. Mutações podem ocorrer em qualquer célula, mas, do ponto de vista da teoria da evolução, as mais importantes são aquelas que ocorrem durante a produção dos gametas (célula especiais utilizadas na reprodução dos organismos vivos) [Rid04]. Isso porque as mutações nos gametas podem ser transmitidas aos descendentes durante a reprodução dos organismos.

No contexto dos EAs, o processo de mutação é simulado pelos operadores de mutação. Esses operadores provocam modificações aleatórias nos genes dos indivíduos. A mutação juntamente com o cruzamento permite que o intervalo completo dos alelos seja acessível em cada gene. Desse modo, os operadores de mutação são responsáveis por introduzir diversidade na população, permitindo que regiões inexploradas do espaço de busca possam ser alcançadas.

Deb e Goyal [DG96] propuseram um operador de mutação para algoritmos genéticos com representação real que realiza variações em cada gene de acordo com uma função de distribuição de probabilidade polinomial. Esse operador é denominado de *mutação polinomial*. Esse operador de mutação é aplicado com uma probabilidade p_m sobre genes dos cromossomos produzidos pelo operador de cruzamento. A probabilidade p_m é chamada probabilidade de mutação. Essa probabilidade é geralmente um valor pequeno para assegurar que boas soluções não sejam perdidas. Assim, esse operador é aplicado probabilisticamente sobre cada variável de decisão de modo que a probabilidade de um cromossomo ser afetado pelo operador de mutação é dado pela equação (2.15)

$$p(\text{mutação ser aplicada em } \mathbf{x}_i) = 1 - (1 - p_m)^n. \tag{2.15}$$

em que n é a dimensão do espaço de busca.

Para explicar a mutação polinomial, considere novamente o caso unidimensional. Seja x o gene da solução original, y o gene da solução após a aplicação do operador de mutação (gene mutado), e Δ_{max} a pertubação máxima permitida sobre o gene x para produzir y. Então, a

mutação polinomial funciona como segue:

1. Amostrar um valor de uma função de distribuição uniforme no intervalo $[0,1]$, isto é, $u \sim U(0,1)$;
2. Calcule o fator de pertubação $\overline{\delta}$ usando a equação (2.16),

$$\overline{\delta} = \begin{cases} (2u)^{\frac{1}{\eta_m+1}} - 1 & \text{se } u \leq \frac{1}{2}, \\ \left(1 - [2(1-u)]^{\frac{1}{\eta_m+1}}\right) & \text{caso contrário.} \end{cases} \tag{2.16}$$

   em que η_m é o índice da distribuição do operador de mutação.

3. Calcule o gene mutado y conforme a equação (2.17),

$$y = x + \overline{\delta}\Delta_{max}. \tag{2.17}$$

Para o caso multidimensional, esse procedimento pode ser repetido independentemente para cada variável de decisão. Cabe ressaltar que o procedimento apresentado apenas é usado para variáveis em que os limites superiores e inferiores não são especificados, isto é $x \in [-\infty, +\infty]$. Quando uma variável é restrita a um limite superior x_{max} e inferior x_{min}, isto é $x \in [x_{min}, x_{max}]$, a equação (2.18) é usada no lugar da equação (2.16),

$$\overline{\delta} = \begin{cases} \left[2u + (1-2u)(1-\delta)^{\eta_m+1}\right]^{\frac{1}{(\eta_m+1)}} - 1 & \text{se } u \leq \frac{1}{2}, \\ 1 - \left[2(1-u) + 2(u-\frac{1}{2})(1-\delta)^{\eta_m+1}\right]^{\frac{1}{(\eta_m+1)}} & \text{caso contrário,} \end{cases} \tag{2.18}$$

em que δ é dado pela equação (2.19)

$$\delta = \frac{\min\left[(x - x_{min}), (x_{max} - x)\right]}{(x_{max} - x_{min})}. \tag{2.19}$$

Essa equação assegura que nenhuma solução será criada fora do intervalo $[x_{min}, x_{max}]$. É importante mencionar que ao se usar essa equação, deve-se fazer $\Delta_{max} = x_{max} - x_{min}$ quando a equação (2.17) for aplicada.

2.2.4 Seleção para Sobrevivência

Após a aplicação dos operadores de variação, duas populações existem: a população da geração atual e a população de filhos. Sendo assim, a seleção para sobrevivência deve ser aplicada para determinar quais os indivíduos das duas populações participarão da próxima geração. Geralmente o tamanho da população em um EA é mantido constante ao longo das gerações. Por outro lado, a cada geração os operadores de variação produzem novas soluções. Desta forma, a seleção para sobrevivência é importante porque ela permite manter a população constante a partir da seleção de indivíduos que participarão da geração seguinte. Esse operador é geralmente implementado de modo determinístico em que as soluções com melhor aptidão automaticamente participam da geração seguinte, fenômeno conhecido como *elitismo* [Pur03].

Nos algoritmos Estratégias de Evolução, a seleção para sobrevivência ocorre por meio das estratégias (μ, λ) e $(\mu + \lambda)$ em que μ denota a população de pais e λ é a população de filhos. No esquema (μ, λ) a população de pais μ é substituída pela população de λ filhos. Ou seja, dos λ filhos produzidos, μ são selecionados para compor a próxima geração. No esquema $(\mu + \lambda)$ tanto a população de pais como a população de filhos competem para participarem da geração seguinte. Isto é, da população combinada de pais e filhos com $\mu + \lambda$ indivíduos, os μ indivíduos mais apitos passarão para a próxima geração. Existem outros métodos de seleção para sobrevivência e cada uma das formulações de EA têm a sua maneira de realizar essa seleção. Os métodos de seleção para variação já apresentados, por exemplo, podem ser utilizados também para a seleção para a sobrevivência [Pur03, BHS97].

2.3 Otimização Multiobjetiva por Algoritmos Evolucionários

Como já foi discutido, a otimização multiobjetiva consiste na otimização de mais de dois objetivos conflitantes entre si. A solução para esse problema consiste em um conjunto de soluções representando os melhores compromissos possíveis entre os objetivos, que, considerando a dominância de Pareto, representam as soluções Pareto-ótimas.

Para resolver MOPs, muitas técnicas foram propostas na literatura [CLV07]. As técnicas tradicionais para lidar com esse tipo de problema, em geral, realizavam uma transformação do problema multiobjetivo em um problema mono-objetivo [Adr07], e nessa conversão alguns parâmetros são normalmente introduzidos [Zit99]. O problema convertido era então resolvido por alguma técnica convencional de otimização mono-objetivo [Adr07]. Alguns representantes dessa classe de técnicas são o método da soma ponderada, o método da ε-restrição e a *Goal Programming* [Adr07, CLV07]. Como resultado da transformação do MOP em um problema com um único objetivo, uma única solução sobre a Frente de Pareto é encontrada em cada execução do algoritmo [Adr07, CLV07]. Para gerar uma boa aproximação da Frente de Pareto, esses algoritmos precisam ser executados várias vezes usando diferentes parâmetros, na expectativa de que, com isso, soluções diferentes sejam encontradas [Adr07]. Além dessa desvantagem, outras dificuldades dos métodos tradicionais podem ser enumeradas, tais como:

1. Alguns algoritmos são sensíveis a forma da Frente de Pareto. Métodos como o da soma ponderada, por exemplo, não conseguem encontrar soluções em regiões não-convexas da Frente de Pareto [Adr07];

2. Esses métodos podem envolver a agregação de objetivos em escalas diferentes [Adr07];

3. Esses métodos geralmente requerem informação de gradiente [Adr07].

Nesse sentido, os EAs se apresentam como bons candidatos para resolver MOPs devido, principalmente, à sua natureza baseada em população [LKC$^+$12]. Essa característica permite que os EAs ofereçam uma aproximação da Frente de Pareto em uma única execução do algoritmo [CLV07]. Os EAs permitem aproveitar a natureza multiobjetiva do problema em sua própria solução ao invés de convertê-lo em um problema mono-objetivo. Isso evita o problema de se agregar objetivos incomensuráveis (em unidades diferentes) como é comum nas

técnicas tradicionais. Outra vantagem da natureza baseada em população dos EA é o fato de que, ao lidarem com múltiplas soluções simultaneamente, o processo de busca ocorre de um modo cooperativo e sinergético, exigindo menos avaliações da função objetivo [Pur03, Adr07]. Por fim, as vantagens já mencionadas dos EAs de lidarem com espaços de buscas enormes, multimodalidade e não usarem informação de gradiente também se aplicam aos problemas multiobjetivos [Adr07, CLV07].

Em virtude de todas essas características mencionadas, existem na literatura muitas adaptações de EAs em MOPs [CLV07]. Sendo que qualquer umas dessas adaptações recebe o nome genérico de Algoritmo Evolucionários Multiobjetivo (**MOEA**, *Multiobjective Evolutionary Algorithm*) [CLV07]. As primeiras extensões de EA para problemas multiobjetivos datam de meados de 1980 [Coe06, Pur03]. Em particular, o algoritmo VEGA (*Vector Evaluated Genetic Algorithm*) proposto por David Schaffer é normalmente referido como a primeira extensão de um EA para tratar problemas multiobjetivos [Sch84, Sch85, Coe06]. No VEGA existe uma sub-população para cada objetivo. Os indivíduos de uma dada sub-população têm como aptidão o valor da função objetivo corresponde àquela sub-população. Esse mecanismo, entretanto, apresenta uma grande desvantagem pois leva a um fenômeno conhecido como especiação, em que as soluções apresentam um bom desempenho em um único objetivo e uma grande deterioração em outro [Pur03].

A próxima estratégia relevante foi o **ordenamento lexicográfico** com Algoritmo Genético [Coe06, Adr07, CLV07]. Nessa abordagem, o objetivo considerado o mais importante é otimizado primeiro. Em seguida, o segundo objetivo mais importante é otimizado, mas sujeito à condição de não piorar o valor obtido no primeiro objetivo. Depois, o terceiro objetivo é otimizado sob a condição de não piorar os valores obtidos no primeiro e segundo objetivo. Esse procedimento é repetido para todos os objetivos restantes. A desvantagem dessa técnica é que ela necessita que os objetivos sejam ordenados por importância, conhecimento que em alguns problemas não está disponível.

Apesar dessas propostas iniciais, o primeiro grande avanço no desenvolvimento dos MOEAs ocorreu a partir da sugestão dada David E. Goldberg em 1989 [FF95, Adr07, Coe06] de incorporar dominância de Pareto na atribuição de aptidão para as soluções da população. A sugestão de Goldberg é baseada no conceito de solução localmente não-dominada [Pur03] que é definida segundo a Definição 2.13.

Definição 2.13 (**Solução localmente não-dominada**)**.** Um solução $\mathbf{a} \in A$ (vetor no espaço objetivo) é dito ser localmente não dominado com respeito ao conjunto A de vetores objetivos, se e somente se $\nexists \mathbf{x} \in A$ tal que $\mathbf{x} \prec \mathbf{a}$.

É importante mencionar que uma solução é sempre não-dominada com respeito a um determinado conjunto. Além disso, uma solução localmente não-dominada não é necessariamente uma solução Pareto-ótima. Usando essa definição, pode-se dizer que um vetor no espaço objetivo é um vetor não-dominado globalmente se ele é localmente não-dominado por todos os vetores da região factível $\mathscr{O}$. Nesse caso, essa solução é Pareto-ótima. Outra definição importante é o de conjunto não-dominado [Kha02]:

Definição 2.14 (**Conjunto não-dominado**)**.** Um conjunto $\mathscr{N}$ é dito ser um conjunto não-dominado com respeito a um conjunto A de soluções, se $\mathscr{N}$ contém todas as soluções local-

mente não-dominadas de A [Zit99, Kha02]. Ou seja,

$$\mathscr{N} = \{a \mid a \text{ é localmente não-dominado com relação à } A\}. \tag{2.20}$$

Nesta dissertação, quando se mencionar apenas que um conjunto $\mathscr{N}$ é um conjunto não-dominado isto significa que todas as suas soluções são localmente não-dominadas com relação a ele.

Com essas definições, pode-se explicar a sugestão de Goldberg que consiste no método de atribuição de aptidão denominado de **ordenamento por não-dominação** [FF95, Coe06]. Nesse método, um conjunto $\mathscr{P}^{(1)}$ de soluções é ordenado do seguinte modo. Primeiramente, identificam-se as soluções localmente não-dominadas do conjunto $\mathscr{P}^{(1)}$. Essas soluções são inseridas em um conjunto $\mathscr{F}^{(1)}$. Em seguida, forma-se o conjunto $\mathscr{P}^{(2)} = \mathscr{P}^{(1)} - \mathscr{F}^{(1)}$. O procedimento é então repetido agora para o conjunto $\mathscr{P}^{(2)}$, isto é, as soluções localmente não-dominadas desse conjunto são inseridas no conjunto $\mathscr{F}^{(2)}$ e as soluções remanescentes formam o conjunto $\mathscr{P}^{(3)}$. Esse procedimento continua até se obter um conjunto $\mathscr{P}^{(s+1)}$ vazio. Com isso, no final desse processo, todas as soluções presentes inicialmente no conjunto $\mathscr{P}^{(1)}$ são distribuídas nos conjuntos $\mathscr{F}^{(1)}, \mathscr{F}^{(2)}, \cdots, \mathscr{F}^{(s)}$ que recebem o nome de *frentes*. Por fim, essas soluções recebem um valor de aptidão do seguinte modo: as soluções do conjunto $\mathscr{F}^{(1)}$ recebem valor 1, as do conjunto $\mathscr{F}^{(2)}$ recebem valor 2, e assim sucessivamente. De acordo com esse método de atribuição de aptidão, soluções com menor aptidão são preferidas, com isso espera-se que as soluções tendam a se aproximar da Frente de Pareto global com o passar das gerações. A Figura 2.8 apresenta um exemplo das frentes formadas pelo procedimento de ordenamento por não-dominação. Cada uma das frentes formadas é um conjunto não-dominado.

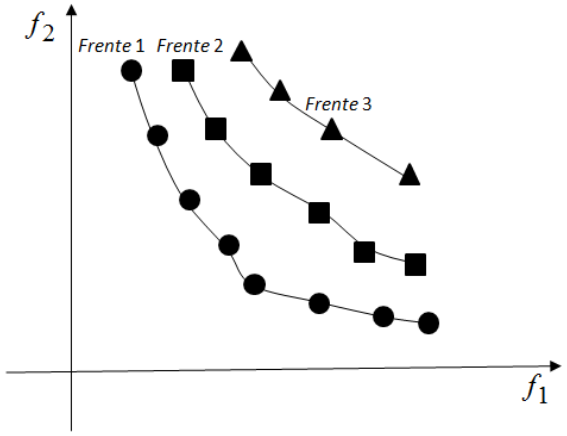


Figura 2.8 Exemplos de frentes geradas pelo método de ordenamento por não-dominação.

Goldberg também sugeriu uma técnica de *niching* para evitar a convergência dos algoritmos para um único ponto e desse modo preservar várias soluções distintas na população [Coe06, CLV07, Pur03]. Basicamente, técnicas de *niching* realizam uma análise de densidade em torno de todas as soluções dentro de uma população de modo a degradar a aptidão de soluções que estejam em conglomerados de soluções, isto é, estejam envolvidas por muitas soluções. Desse

modo, regiões menos densas no espaço objetivo são privilegiadas e o resultado global do *niching* é distribuir as soluções ao longo da superfície da Frente de Pareto [Coe06, Pur03].

O uso de dominância de Pareto como uma forma de atribuir aptidão para as soluções, de modo a favorecer as soluções localmente não-dominadas, juntamente com o uso de técnicas de *niching* levou a definição de novos algoritmos que formaram a primeira geração de MOEAs [Pur03, Adr07, Coe06]. Essa geração é apresentada na próxima seção.

2.3.1 Primeira Geração

No início de 1990, surgiram os MOEAs da primeira geração cujo principal aspecto é usar dominância de Pareto na definição da aptidão das soluções [Pur03]. Alguns dos mais importantes e citados MOEAs [Pur03, Adr07, Coe06, CLV07] que fazem parte dessa geração são descritos a seguir.

***Multi-Objective Genetic Algorithm* (MOGA)**

O MOGA foi proposto por Carlos M. Fonseca e Peter J. Fleming [FF93]. O MOGA usa uma estratégia de atribuição de aptidão conhecida como ranqueamento de dominância. Nesse método o valor de aptidão de uma solução consiste no número de soluções que dominam essa solução (mais um). Com isso, as soluções localmente não-dominadas na população recebem uma aptidão igual a um. Aqui novamente quanto menor for a aptidão de uma solução mais preferida se torna essa solução. Comparada à abordagem de ordenamento por não-dominação, esse método apresenta uma maior capacidade de resolução [Adr07]. Em outras palavras, a probabilidade de uma solução ter uma aptidão igual a outra é menor no ranqueamento de dominância que no método de ordenamento por não-dominação.

Niched Pareto Genetic Algorithm (NPGA)

O NPGA foi proposto por Jeffrey Horn e colaboradores [HNG94] e sua principal característica é um esquema de seleção por torneio usando dominância de Pareto. Nesse esquema, dois indivíduos são selecionados aleatoriamente da população e comparados contra um conjunto de referência de indivíduos retirados aleatoriamente da população (geralmente 10 % da população). Se uma das duas soluções é não-dominada pelo conjunto de referência, isto é, se nenhuma solução do conjunto de referência domina a solução, então ela é selecionada para variação. No caso em que as duas soluções são não-dominadas ou, similarmente, são simultaneamente dominadas pelo conjunto de referência (no mínimo uma solução do conjunto referência domina essas soluções), então uma técnica de *niching* é utilizada para decidir qual das duas soluções participará do processo de variação. Nesse caso, a solução presente na região menos densa do espaço objetivo é selecionada.

***Non-dominated Sorting Genetic Algorithm* (NSGA)**

O NSGA foi proposto por Srinivas e Deb [SD94]. O NSGA utiliza o mecanismo de ordenamento por não-dominação proposto por Goldberg para atribuir valores de aptidões para as soluções da população. Os indivíduos da primeira frente são inicialmente identificados da população atual e eles recebem um valor de aptidão alto. Em seguida, esses indivíduos compartilham suas aptidões usando um método de *niching*. No NSGA, esse compartilhamento é

alcançado dividindo a aptidão de cada solução por uma quantidade que é proporcional ao número de indivíduos ao redor dessa solução. O resultado dessa operação constitui o novo valor de aptidão dos indivíduos. Posteriormente, os indivíduos da segunda frente são processados. A esses indivíduos são atribuídos um valor de aptidão menor que o menor valor de aptidão encontrado na primeira frente após o processo de compartilhamento. Esse processo é repetido até a última frente seja processada. Após o processo de atribuição de aptidão, os indivíduos são selecionados para o processo de variação (reprodução) usando o método de seleção proporcional estocástica. Os filhos produzidos seguem então para a próxima geração e o algoritmo continua até que uma condição de parada seja satisfeita.

2.3.2 Segunda Geração

No final de 1990, surge a segunda geração de MOEAs [Adr07]. Esses algoritmos também utilizam a dominância de Pareto para atribuir um valor de aptidão às soluções, mas apresentam duas novas características, a saber, a presença do elitismo e o uso de métodos mais sofisticados para promoção da diversidade [Adr07, Coe06]. Alguns desses algoritmos são descritos nas próximas subseções.

2.3.2.1 Strength Pareto Evolutionary Algorithm (SPEA)

O *Strength Pareto Evolutionary Algorithm* (SPEA) foi proposto por Eckart Zitzler e Lothar Thiele [ZT99]. No SPEA, o elitismo ocorre por meio de uma estrutura chamada de arquivo externo (ou população externa) que consiste em um memória auxilar em que as soluções não-dominadas encontradas pelo algoritmo ao longo das gerações são armazenadas. A principal motivação para o uso do arquivo externo é o fato de que uma solução que é não-dominada com relação à população atual não é necessariamente não-dominada com respeito a todas soluções produzidas pelo algoritmo [Coe06]. Assim, o SPEA tem duas memórias: uma interna, que é a população de indivíduos responsável por realizar a busca, e uma memória externa, que é responsável por armazenar apenas soluções não-dominadas encontradas ao longo de todo processo de busca. O pseudo-código do SPEA é apresentado em Algoritmo 2.

No pseudo-código do SPEA apresentado, a função *Avalie* funciona do seguinte modo. Para cada solução no arquivo externo é atribuído um valor chamado de *intensidade* que consiste no número de soluções que ela domina na população interna P. Em seguida, a aptidão de um indivíduo na população interna é calculada com base nas intensidades das soluções do arquivo externo que dominam esse indivíduo. Em outras palavras, o SPEA primeiramente atribui a aptidão dos indivíduos do arquivo externo segundo o **Passo 1** e, em seguida, atribui a aptidão dos indivíduos da população de acordo com o **Passo 2**.

Passo 1 Para cada indivíduo $i \in AE$ é atribuído uma intensidade $S(i) \in [0,1)$. Seja n o número de soluções na *população interna* que são dominadas pelo indivíduo i. Então, $S(i)$ é definido como $S(i) = \frac{n}{|P|+1}$ em que $|P|$ é o tamanho da população. A aptidão de i denotada por $f(i)$ é então definida como $f(i) = S(i)$.

Passo 2 A aptidão de um indivíduo $j \in P$ é calculada somando-se as intensidades de todas as soluções do arquivo externo AE que dominam j. Ao final dessa soma, se adiciona 1 para

que as soluções do arquivo externo sejam preferidas às soluções da população. Assim, a aptidão de j é dada pela equação (2.21):

$$f(j) = 1 + \sum_{i \in AE, i \prec j} S(i), \quad \forall j \in P. \tag{2.21}$$

Um dos problemas com o uso do arquivo externo é que seu tamanho pode crescer rapidamente e isso por sua vez pode comprometer o processo de busca [CLV07, Zit99]. Assim, é necessário estabelecer um limite para o tamanho do arquivo externo e eliminar soluções caso esse limite seja ultrapassado. O processo de eliminação de soluções do arquivo externo é denominado nessa dissertação de poda. No SPEA, a poda é realizada por meio de uma técnica de agrupamento aglomerativo hierárquico denominada de método de ligação média. Assim, se o tamanho do arquivo externo $|AE|$ excede o número máximo de soluções permitidas AE_{max}, então AE_{max} agrupamentos são criados usando o método de ligação médio, e, em seguida, o representante de cada grupo é selecionado para formar o novo arquivo externo com exatamente AE_{max} soluções.

Algoritmo 2: Pseudo-código do SPEA.

```
1  Inicialize a população P^(0);
2  Crie um arquivo externo vazio AE^(0) = ∅;
3  t = 0;
4  enquanto condição de parada não for alcançado faça
       /* Obter as soluções localmente não-dominadas do
          conjunto P^(t)                                   */
5      AE^(t+1) = AE^(t) ∪ obter-conjunto-não-dominado(P^(t));
       /* Obter as soluções localmente não-dominadas do
          conjunto AE^(t+1)                                */
6      AE^(t+1) = obter-conjunto-não-dominado(AE^(t+1));
7      Realize a poda de AE^(t) se necessário;
8      para cada i ∈ AE^(t+1) faça
9          Avalie(i)
10     para cada j ∈ P^(t) faça
11         Avalie(j)
       /* MP é o mating pool                               */
       /* Torneio binário é usado na seleção para variação */
12     MP = seleção-para-variação(P^(t) ∪ AE^(t+1));
       /* Aplique operadores de cruzamento e mutação em MP para
          formar a nova população P^(t+1)                  */
13     P^(t+1) = aplique-operadores-de-variação(MP);
14     t = t + 1.
```

2.3.2.2 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

O *Strength Pareto Evolutionary Algorithm 2* (SPEA2) [ZLT01] apresenta três diferenças básicas com relação ao seu antecessor o *SPEA*, a saber: (1) um método de atribuição de aptidão que considera para cada indivíduo o número de indivíduos que ele domina bem como o número de indivíduos que o dominam; (2) ele usa uma técnica de estimação de densidade baseada no vizinho mais próximo; e (3) ele tem um novo método de poda que preserva as soluções de fronteira [CLV07] . O pseudo-código do SPEA2 é apresentado em Algoritmo 3.

Algoritmo 3: Pseudo-código do SPEA2.

1 Inicialize a população $P^{(0)}$;
2 Crie um arquivo externo vazio $AE^{(0)} = \emptyset$;
3 $t = 0$;
4 **enquanto** *condição de parada não for alcançado* **faça**
5 Calcule a aptidão das soluções de $AE^{(t)} \cup P^{(t)}$;
6 Faça $AE^{(t+1)} =$ `obter-conjunto-não-dominado`$(AE^{(t)} \cup P^{(t)})$;
7 Realize a poda de $AE^{(t+1)}$ se $|AE^{(t+1)}| > AE_{max}$;
8 Se o tamanho de $AE^{(t+1)}$ não atingiu o tamanho máximo AE_{max}, então preencha $AE^{(t+1)}$ com as $AE_{max} - |AE^{(t+1)}|$ melhores soluções de $AE^{(t)} \cup P^{(t)}$ de acordo com as suas aptidões;
`/* Torneio binário é usado na seleção para variação */`
9 $MP =$ `seleção-para-variação`$(P^{(t)} \cup AE^{(t+1)})$;
`/* Aplique operadores de cruzamento e mutação em` MP `para formar a nova população` $P^{(t+1)}$ `*/`
10 $P^{(t+1)} =$ `aplique-operadores-de-variação`(MP) ;
11 $t = t + 1$.

O método de atribuição de aptidão do SPEA2 ocorre do seguinte modo. Primeiramente, as intensidades dos indivíduos de $AE \cup P$ são determinadas. No SPEA2, a intensidade de um indivíduo $i \in AE \cup P$ é dada por:

$$S(i) = |\{j \,|\, j \in AE \cup P \wedge i \prec j\}| \tag{2.22}$$

Com base nos valores de intensidade, a aptidão grosseira $R(i)$ de um indivíduo $i \in (P \cup AE)$ é definida como:

$$R(i) = \sum_{j \in (P \cup AE), j \prec i} S(j) \tag{2.23}$$

Para evitar o caso em que dois ou mais indivíduos apresentem um mesmo valor de aptidão grosseira, o SPEA2 agrega a essa aptidão um método de estimação de densidade. Desse modo,

para cada indivíduo i ele calcula a sua densidade de acordo com:

$$D(i) = \frac{1}{2+\sigma_i^k} \tag{2.24}$$

em que σ_i^k é a distância do indivíduo i para o seu k-ésimo vizinho mais próximo. É importante destacar que $D(i) < 1$. Por fim, a aptidão de cada indivíduo é determinado por:

$$F(i) = R(i) + D(i). \tag{2.25}$$

Como se pode notar, o termo $R(i)$ é um número inteiro enquanto $D(i)$ é sempre um número real em $(0,1)$. Dessa forma, usando a aptidão $F(i)$ a dominância de Pareto é sempre privilegiada e no caso em que dois indivíduos possuem o mesmo valor em R a preferência se dá pelo termo D que é a densidade.

O mecanismo de poda no SPEA2 funciona do seguinte modo. Elimina-se o indivíduo com menor distância para o seu vizinho mais próximo. Caso haja empates, elimina-se o indivíduo com menor distância para o segundo vizinho mais próximo e assim sucessivamente. Esse procedimento é repetido até que o arquivo externo atinja o seu tamanho máximo permitido.

2.3.2.3 Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

O NSGA-II foi proposto por Deb *et al.* [DPAM02]. Diferentemente do NSGA, o NSGA II apresenta um novo mecanismo de estimação de densidade denominado de *crowding distance* (*CD*). A vantagem desse estimador é que ele não requer nenhum parâmetro do usuário. Geometricamente, o *crowding distance* de uma solução i corresponde ao semi-perímetro de um hiper-retângulo conectando as soluções adjacentes (os vizinhos imediatos) da solução i. A Figura 2.9 ilustra a interpretação geométrica do *crowding distance* em um espaço objetivo de duas dimensões. Como se pode observar pela Figura 2.9, as soluções adjacentes à solução i são as soluções a e b. O retângulo conectando as soluções a e b está hachurado. O *crowding distance* da solução i consiste no semi-perímetro desse retângulo, isto é, na soma das distâncias d_1 e d_2.

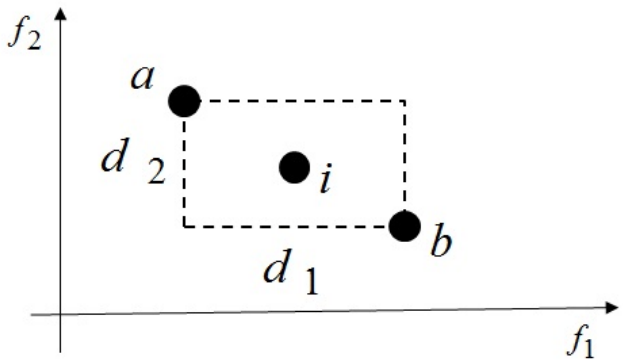


Figura 2.9 Exemplo do cálculo do *crowding distance* em um espaço objetivo bidimensional.

O pseudo-código do NSGA II é apresentado em Algoritmo 4. Como se observa, o NSGA II inicia de modo usual criando um população inicial $P^{(0)}$. Em seguida, se atribuem aptidões a esses indivíduos usando o ordenamento por não-dominação. Posteriormente, se realiza a seleção para variação usando torneio binário para formar o *mating pool MP*. Os operadores de cruzamento e reprodução são então aplicados nos indivíduos do *mating pool* para formar a população de filhos.

Algoritmo 4: Pseudo-código do NSGA II.

```
1  Inicialize a população P^(0) /* População na geração t = 0          */
2  classificação-por-não-dominação(P^(0));
   /* Torneio binário é usado na seleção para variação               */
3  MP = seleção-para-variação(P^(0)) ;
4  // Aplique operadores de cruzamento e mutação em MP para formar a população de filhos Q^(0)
   Q^(0) = aplique-operadores-de-variação(MP) ;
5  t = 0;
6  enquanto t < t_max faça
7      R^(t) = P^(t) ∪ Q^(t)  // Combine a população de pais com a de filhos ;
8      P^(t+1) = ∅  // População da próxima geração;
9      i = 1;
10     F = classificação-por-não-dominação(R_(t));
       /* F = (F_1, F_1, ...)                                          */
11     enquanto |P^(t+1)| + |F_i| ≤ N faça
12         calcule-crowding-distance(F_i);
13         P^(t+1) = P^(t+1) ∪ F_i;
14         i = i + 1;
15     se (N − |P^(t+1)|) > 0 então
16         calcule-crowding-distance(F_i);
17         classifique-por-crowding-distance(F_i) /* Classifique em
               ordem decrescente de crowding distance                 */
18         P^(t+1) = P^(t+1) ∪ F_i[1 : (N − |P^(t+1)|)] ;
19     MP = seleção-para-variação(P^(t+1)) ;
20     Q^(t+1) = aplique-operadores-de-variação(MP) ;
21     t = t + 1;
```

Após essa etapa, o NSGA II entra em um ciclo. No NSGA II, o elitismo assume uma forma diferente do elitismo presente no SPEA e no SPEA2. No NSGA II o elitismo ocorre por meio de uma seleção para sobrevivência do tipo $(\mu + \lambda)$. Assim, a população em uma geração t denotado por $P^{(t)}$ é combinada com a população de filhos dessa geração $Q^{(t)}$ para formar uma população $R^{(t)}$. É importante destacar que se N é o número de indivíduos na população de pais $P^{(t)}$, a população de filhos também tem um tamanho N e consequentemente

a população combinada $R^{(t)}$ tem tamanho $2N$. Portanto, é necessário selecionar N indivíduos de $R^{(t)}$ para formar a próxima geração de indivíduos $P^{(t+1)}$. No NSGA II isso é realizado usando novamente o ordenamento por não-dominação. Fazendo isso, a população $R^{(t)}$ é divida em várias frentes $F_1, F_2, F_3, \ldots, F_l$. Ao se fazer isso, a população $P^{(t+1)}$ é preenchida do seguinte modo. Primeiramente, tenta-se inserir completamente a frente F_1 em $P^{(t+1)}$. Se ela couber integralmente em $P^{(t+1)}$, isto é, se $|F_1| < N$, então todos os indivíduos de F_1 são inseridos em $P^{(t+1)}$. Esse procedimento é repetido para a frente F_2. Caso $|P^{(t+1)}| + |F_2| < N$, então todas as soluções de F_2 são inseridas em $P^{(t+1)}$. Esse procedimento é repetido para as próximas frentes até que se encontre uma frente F_k que não caiba integralmente em $P^{(t+1)}$. Quando essa frente F_k é encontrada, então calcula-se os CDs dos indivíduos desse conjunto e em seguida os indivíduos de F_k são classificados de acordo com CD em ordem decrescente. Desse modo, os $N - P^{(t+1)}$ primeiros indivíduos de F_k são inseridos em $P^{(t+1)}$ de modo que $P^{(t+1)}$ tenha exatamente N indivíduos. Uma vez que a população na próxima geração $P^{(t+1)}$ foi definida, utiliza-se a seleção por torneio binária para o processo de seleção para variação. A seleção por torneio no NSGA II funciona do seguinte modo. Duas soluções são escolhidas aleatoriamente da população e comparadas entre si de acordo com a aptidão obtida pela classificação por não-dominação. A solução com menor aptidão é escolhida. Caso elas tenham a mesma aptidão, isto é, caso elas estejam na mesma frente, a solução com maior *crowding distance* é escolhida.

Como o CD é um conceito que será utilizado mais adiante nessa dissertação, seu cálculo será detalhado aqui. Para se calcular o CD de um conjunto de soluções não-dominadas A, é necessário classificar as soluções desse conjunto de acordo com o valor de cada função objetivo em ordem crescente de magnitude. Em seguida, para um dado objetivo, as soluções com o menor e com o maior valor são identificadas e a elas se atribuem um CD infinito. Para as soluções intermediárias são atribuídas uma distância igual a diferença normalizada dos valores das duas soluções imediatamente adjacentes. Esse procedimento é executado para todos os objetivos, e o CD de uma solução é calculado como a soma de todas as distâncias atribuídas a ela para cada objetivo. Formalmente, o CD de uma solução i, denotado por CD_i, de um conjunto não-dominado de soluções A é dado pela equação (2.26):

$$CD_i = \begin{cases} \sum_{m=1}^{M} CD_i^m, & \text{se } i \text{ não é uma solução de fronteira superior ou inferior do conjunto } A, \\ \infty, & \text{caso contrário.} \end{cases} \tag{2.26}$$

Uma solução do conjunto A é considerada uma solução de fronteira inferior desse conjunto se a Definição 2.15 for satisfeita. Ou uma solução de fronteira superior do conjunto A se a Definição 2.16 for satisfeita.

Definição 2.15 (Solução de fronteira inferior)**.** Uma solução i pertencente a um conjunto não-dominado N é uma solução de fronteira inferior desse conjunto, se $\exists m$, tal que $f_m(\mathbf{x}_i) \leq f_m(\mathbf{x}_j), \quad \forall j \in N$.

Definição 2.16 (Solução de fronteira superior)**.** Uma solução i pertencente a um conjunto não-dominado N é uma solução de fronteira superior desse conjunto, se $\exists m$, tal que $f_m(\mathbf{x}_i) \geq f_m(\mathbf{x}_j), \quad \forall j \in N$

O termo CD_i^m da equação (2.26) é a parcela do *crowding distance* da solução i corresponde ao objetivo m, e ele é calculado de acordo com a equação (2.27):

$$CD_i^m = \frac{f_m(\mathbf{x}_a) - f_m(\mathbf{x}_b)}{f_m^{max} - f_m^{min}}, \tag{2.27}$$

em que a e b são soluções adjacentes à solução i em relação ao objetivo m. A solução a é uma solução adjacente inferior à i com respeito à função objetivo m se a seguinte relação é verdade:

$$f_m(\mathbf{x}_a) \geq f_m(\mathbf{x}_j), \quad \forall j \in \{l \,|\, f_m(\mathbf{x}_l) < f_m(\mathbf{x}_i)\}. \tag{2.28}$$

A solução b é uma solução adjacente superior à solução i com respeito à função objetivo m se a seguinte relação é verdade:

$$f_m(\mathbf{x}_a) \leq f_m(\mathbf{x}_j), \quad \forall j \in \{l \,|\, f_m(\mathbf{x}_l) > f_m(\mathbf{x}_i)\}. \tag{2.29}$$

O termo f_m^{max} é o maior dos valores da função objetivo m dentro do conjunto A, isto é, $f_m^{max} = f_m(\mathbf{x}_j)$ tal que $f_m(\mathbf{x}_j) \geq f_m(\mathbf{x}_k), \quad \forall k \in A$. Enquanto o termo f_m^{min} é o menor dos valores da função objetivo m dentro do conjunto A, isto é, $f_m^{min} = f_m(\mathbf{x}_j)$ tal que $f_m(\mathbf{x}_j) \leq f_m(\mathbf{x}_k), \quad \forall k \in A$. Os valores f_m^{max} e f_m^{min} servem para normalizar as distâncias referentes ao objetivo m. A normalização é necessária pois os objetivos podem ser incomensuráveis, isto é, podem está em escalas diferentes. Desse modo, a normalização pode tornar as distâncias dos objetivos adimensionais. O pseudo-código em Algoritmo 5 resume o cálculo do *crowding distance* na forma de algoritmo.

Algoritmo 5: Pseudo-código do cálculo do *crowding distance* das soluções de um conjunto não-dominado A.

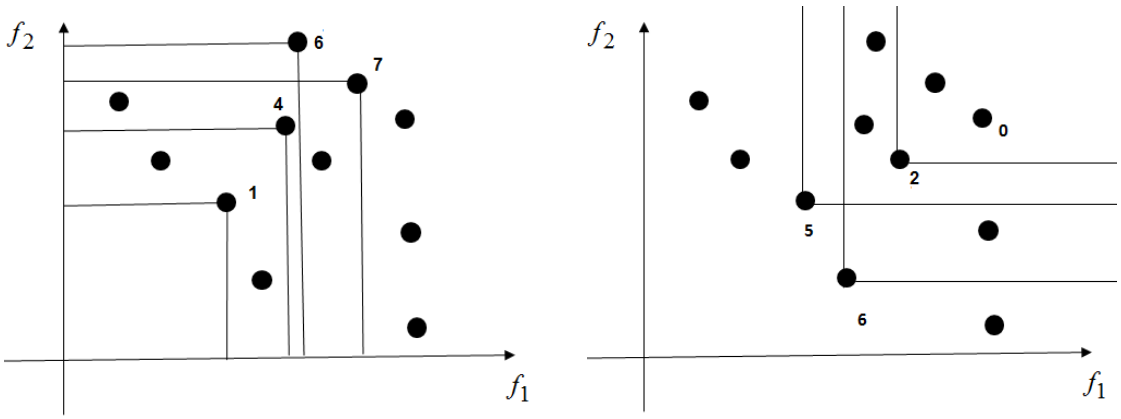
```
1  Seja A um conjunto de soluções não-dominadas no qual se deseja calcular os crowding
   distances das soluções;
2  l = |A| /* l é o número de soluções de A                                    */
3  para i ∈ A faça
4      A[i].CD = 0;
5  para j ∈ {1,2,...,m} faça
       /* Classifique os indivíduos em A de acordo com o
          objetivo j                                                          */
6      A = classifique(A, j);
7      A[1].CD = A[l].CD = ∞;
8      para i = 2 a (l − 1) faça
9          A[i].CD = A[i].CD + (A[i+1].j − A[i−1].j)/(f_j^max − f_j^min);
```

2.3.3 Métodos de Atribuição de Aptidão Baseados em Dominância

Após essa rápida revisão sobre os principais MOEAs que surgiram durante o desenvolvimento da área, é possível perceber que os métodos de atribuição baseados em dominância de Pareto

estão no cerne de alguns dos principais e modernos MOEAs. Esses métodos buscam favorecer soluções localmente não-dominadas sobre as soluções dominadas e desse modo eles tendem a guiar a população em direção da Frente de Pareto. Alguns exemplos de métodos que fazem uso de dominância de Pareto para atribuir um valor de aptidão para cada solução da população são [CLV07]:

- **Ranqueamento de dominância:** Esse método atribui como aptidão para uma dada solução, a quantidade de soluções que a dominam (mais um). A Figura 2.10(a) ilustra o processo de atribuição de aptidão do ranqueamento dominância. Cada solução (círculos) está rotulada com o valor da sua aptidão.
- **Contagem de dominância:** Esse método atribui como aptidão para uma dada solução, o número de soluções que ela domina. A Figura 2.10(b) ilustra o processo de atribuição de aptidão desse método.
- **Profundidade de dominância:** Nesse método, a aptidão de cada solução é determinada pelo ordenamento por não-dominação. Ou seja, após as frentes serem formadas, as soluções da primeira frente recebem aptidão 1, as da segunda frente recebem aptidão 2 e assim sucessivamente até a última frente.



(a) Ranqueamento de dominância. (b) Contagem de dominância.

Figura 2.10 Exemplos de métodos de atribuição de aptidão.

2.4 Considerações Finais

Nesse capítulo, foram apresentados os principais conceitos de MOEAs. Em particular, deu-se ênfase aos métodos de atribuição de aptidão baseados em dominância de Pareto que foram responsáveis pelo desenvolvimento dos MOEAs. Esses métodos estão presentes nos MOEAs mais promissores. No entanto, o bom desempenho desses MOEAs, que utilizam dominância de Pareto, é restrito a problemas com apenas dois ou três objetivos. No próximo capítulo,

será mostrado que esses MOEAs têm o seu desempenho comprometido quando o número de objetivos é maior que quatro.

CAPÍTULO 3

Otimização de Muitos Objetivos Por Algoritmos Evolucionários

" O homem é uma corda esticada
entre o animal e o super-homem,
uma corda por cima do abismo. "
– Friedrich Nietzsche

OS MOEAs baseados em dominância de Pareto conseguiram obter bastante sucesso em problemas reais com dois e três objetivos [CLV07, BFFB+11]. Contudo, muitos problemas reais envolve um número maior de objetivos [Gar09, PF03, Hug05, FPL05]. Isso motivou a aplicação desses algoritmos em problemas com essas características. Problemas multiobjetivos com mais de três objetivos são denominados na literatura de problemas com muitos objetivos (**MaOPs**, *Many Objective Problems*) [ITN08, PF03]. Assim, a partir de 2002 iniciou-se estudos no sentido de investigar primeiro a escalabilidade dos MOEAs baseados em dominância de Pareto em MaOPs [Kha02]. A partir desses estudos, constatou-se, de um modo geral, que esses algoritmos não conseguem escalar devido a uma dificuldade intrínseca da dominância de Pareto. Essa dificuldade consiste na incapacidade da dominância de Pareto de discriminar, isto é, de impor aptidões diferentes às soluções em MaOPs. Isto porque todas as soluções do espaço objetivo tendem a se tornar incomparáveis com o aumento do número de objetivos. Esse problema será exposto em mais detalhes na Seção 3.1. A dificuldade inerente dos MOEAs baseados em dominância de Pareto em MaOPs, levou os pesquisadores da área a substituir a dominância de Pareto por outros métodos com uma maior capacidade discriminativa das soluções. Alguns desses métodos são apresentados em detalhes na Seção 3.2 e 3.3.

3.1 Dificuldades Impostas por Problemas com Muitos Objetivos

Como já foi mencionado, MOEAs baseados em dominância de Pareto são ineficazes em problemas com muitos objetivos, devido à incapacidade da dominância de Pareto de impor diferenças entre as soluções nesses problemas. Mais ainda, os MaOPs colocam dificuldades de outras ordens tais como de custo computacional e de visualização das soluções [ITN08, Gar09]. Na Seção 3.1.1 é explicado a razão pela qual a dominância de Pareto falha em discriminar soluções em MaOPs. A Seção 3.1.2 apresenta dificuldades de outra natureza que são comuns a todos os algoritmos multiobjetivos de um modo geral.

3.1.1 Ineficácia da Dominância de Pareto em Discriminar Soluções em Problemas com Muitos Objetivos

O primeiro estudo sobre escalabilidade de MOEAs em problemas com muitos objetivos foi provavelmente realizado por Khare em sua dissertação [Kha02, PF03]. Nesse estudo, Khare comparou três MOEAs baseados em dominância de Pareto, a saber, o NSGA II, o SPEA2, e o PESA. Em seus experimentos, ele constatou que o PESA apresentou uma boa escalabilidade em termos de convergência em relação ao NSGA II e ao SPEA2. Contudo, o PESA não apresentou uma boa escalabilidade em termos de diversidade. Contrariamente, o SPEA2 e o NSGA II escalaram similarmente bem em termos de diversidade mas não conseguiram escalar em termos de convergência em relação ao PESA.

A partir desse estudo, outros trabalhos se seguiram, discutindo a causa da ineficácia de convergência dos MOEAs baseados em dominância de Pareto em problemas com muitos objetivos. Nesse sentido, Purshouse e Fleming [PF03] investigaram o comportamento do NSGA II em MaOPs. Seus resultados mostraram que, assim como no estudo de Khare, a convergência do NSGA II é mitigada com o aumento do número de objetivos [PF03]. Nesse estudo, eles explicaram esse fenômeno argumentando a ineficácia da dominância de Pareto em discriminar soluções uma vez que a proporção de soluções localmente não-dominadas na população cresce rapidamente com o número de objetivos [PF03].

Em [Hug05], Hughes confirmou as observações de Purshouse e Fleming. No entanto, ele também observou que métodos de atribuição de aptidão que não são baseados em dominância de Pareto se mostraram promissores tanto em convergência como diversidade em MaOPs. Assim, ele foi provavelmente o primeiro a verificar que métodos alternativos à dominância de Pareto podem ser usados para lidar com MaOPs.

Em 2004, Farina e Amato [FA04] apresentaram uma explicação teórica para a ineficácia da dominância de Pareto em discriminar soluções em problemas com muitos objetivos. Basicamente, eles fizeram uma análise probabilística e concluíram que a probabilidade de uma solução dominar outra diminui rapidamente com o aumento do número de objetivos. Portanto, à medida que o número de objetivos cresce, existe uma maior tendência de encontrar soluções que sejam incomparáveis. O argumento de Farina e Amato para essa afirmação é reproduzido aqui. Para efeitos de simplificação, considere que o espaço objetivo seja contínuo. Defina agora, o mapeamento $e : \mathscr{O} \mapsto [0,1]$ de modo que, para qualquer $\mathbf{y}^* \in \mathscr{O}$, $e(\mathbf{y}^*)$ é a razão entre a *medida* do *conjunto* de soluções incomparáveis ao vetor $\mathbf{y}^*$ e a *medida* da região factível $\mathscr{O}$. A *medida* de um conjunto é, a grosso modo, o seu tamanho. Por exemplo, se um o conjunto é geometricamente um quadrado, sua medida é a área desse quadrado. Caso o conjunto seja geometricamente um cubo, sua medida é o volume de desse cubo. Com isso, $e(\mathbf{y}^*)$ é definido conforme a equação (3.1)

$$e(\mathbf{y}^*) = \frac{\mu\{\mathbf{y} : \mathbf{y} || \mathbf{y}^*\}}{\mu(\mathscr{O})}, \tag{3.1}$$

em que μ é qualquer medida aditiva, como a medida de Lebesgue, [1] por exemplo.

[1] Em Teoria da Medida, o conceito matemático de *medida* formaliza a noção intuitiva de tamanho de um conjunto, tais com seu comprimento, área e volume. Assim, por exemplo, a medida de Lebesgue do conjunto $[0,1]^2$ é igual a um que corresponde a área do retângulo corresponde no sistema de coordenadas cartesiano

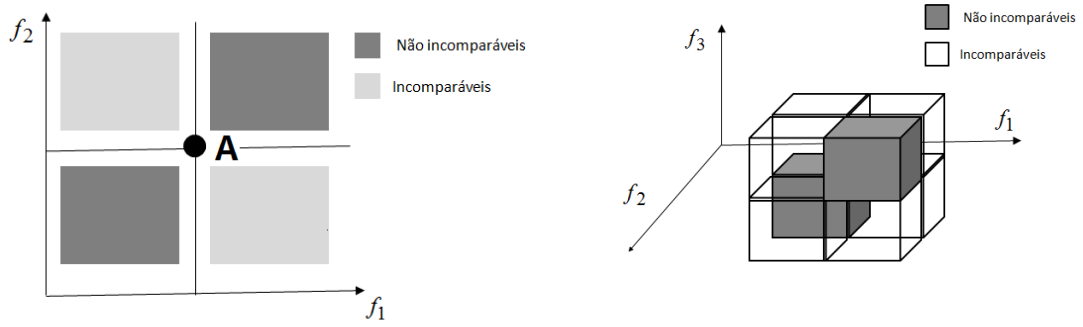
Considere agora que o espaço objetivo é um hipercubo de medida 1. Então, o valor esperado de e denotado por $\widetilde{e}$ é dado pela equação (3.2)

$$\widetilde{e}(m) = \int_{\mathscr{O}} 1 - (y_1 y_2 \dots y_m) - ((1-y_1)(1-y_2)\dots(1-y_m))d\mathbf{y}, \tag{3.2}$$

em que $\mathbf{y} = [y_1, y_2, \dots, y_m]$. Essa integral pode ser expressa, de acordo com [FA04], de um modo mais simples como

$$\widetilde{e}(m) = \frac{2^m - 2}{2^m}. \tag{3.3}$$

A equação (3.3) fica mais fácil de ser interpretada observando a Figura 3.1(a), para o caso em que o espaço objetivo é bidimensional, e observando a Figura 3.1(b), para o caso em que o espaço objetivo é tridimensional. Para a Figura 3.1(a), observa-se que uma solução A divide o espaço objetivo em quatro regiões retangulares, duas das quais são de soluções incomparáveis à solução em consideração. Desse modo, $\widetilde{e}(2) = \frac{2^2-2}{2^2} = \frac{2}{4}$. Para a Figura 3.1(b), observa-se que essa mesma solução divide o espaço objetivo em oito regiões retangulares, seis das quais são de soluções incomparáveis à solução em consideração. Desse modo, $\widetilde{e}(3) = \frac{2^3-2}{2^3} = \frac{6}{8}$.



(a) Espaço objetivo bidimensional. (b) Espaço objetivo tridimensional.

Figura 3.1 Regiões de dominância para uma dada solução.

Analisando a equação (3.3), observa-se que, quando o número de objetivos m tende ao infinito, o valor esperado $\widetilde{e}(m)$ tende a 1. Ou seja, à medida que o número de objetivos aumenta, todas as soluções do espaço objetivo tendem a ser incomparáveis a uma dada solução. Tal comportamento é geral e independente do problema [FA04].

Resumindo, quando o número de objetivos aumenta todas as soluções tendem a ser incomparáveis. Portanto, todos os indivíduos da população de um MOEA apresentam a mesma aptidão, caso os métodos de atribuição de aptidão baseados em dominância sejam usados. Dessa forma, não é mais possível criar uma pressão seletiva em direção à Frente de Pareto do problema e o algoritmo acaba se comportando como uma busca aleatória [Gar09, GPC10, DS05].

3.1.2 Dificuldades Intrínsecas de Problemas com Muitos Objetivos

Além da mitigação da capacidade de busca, outras dificuldades surgem quando se lida com MaOPs. Essas dificuldades são descritas a seguir:

- Aumento exponencial do número de soluções necessárias para aproximar toda Frente de Pareto: a Frente de Pareto de um problema consiste de uma hipersuperfície imersa no espaço objetivo hiperdimensional. Quando o número de objetivos aumenta, a quantidade de soluções não-dominadas necessária para cobrir toda a extensão da Frente de Pareto, com uma dada resolução, cresce exponencialmente [ITN08, LJC09];

- Dificuldade de visualização das soluções: essa dificuldade não representa um problema em si para os algoritmos, no entanto, representa uma dificuldade para o tomador de decisão no momento em que é necessário escolher uma ou mais soluções fornecidas pelo algoritmo [LJC09]. Além disso, essa dificuldade também tem um impacto no projeto e na avaliação de algoritmos, uma vez que se torna muito mais difícil analisar o comportamento deles;

- Aumento do custo computacional: dependendo das estruturas de dados utilizadas e dos mecanismos utilizados para atribuição de aptidão, o custo computacional dos algoritmos pode crescer exponencialmente com o número de objetivos [Gar09].

3.2 Métodos Para Lidar com Problemas com Muitos Objetivos

A partir do trabalho de Hughes [Hug05], as pesquisas começaram a focar no desenvolvimento de métodos que pudessem tornar os MOEAs efetivos em problemas com muitos objetivos. Nesta seção, sumarizam-se algumas dessas técnicas. A classificação das técnicas adotada neste trabalho é baseada na classificação realizada por Ishibuchi *et al.* [ITN08].

3.2.1 Modificação da Relação Dominância de Pareto

Como já foi observado, a região no espaço objetivo dominada por uma dada solução tende a ficar cada vez menor com o aumento do número de objetivos. Então, uma estratégia para resolver esse problema é modificar a relação de dominância de Pareto com o objetivo de aumentar a região de dominância das soluções. Aumentando-se a região de dominância das soluções, diminui-se a quantidade de soluções incomparáveis.

Seguindo essa ideia, Sato *et al.* [SAT07] propuseram uma técnica chamada Controle da Área de Dominância de Soluções (**CDAS**, *Controlling Dominance Area of Solutions*). O CDAS modifica a geometria da relação de dominância de Pareto expandindo ou contraindo a região de dominância de um vetor objetivo $\mathbf{u}$. Isso é feito alterando-se os valores de suas componentes e produzindo um novo vetor $\widetilde{\mathbf{u}}$ conforme a equação (3.4)

$$\widetilde{u_i} = \frac{r \cdot \operatorname{sen}(\omega_i + S_i \cdot \pi)}{\operatorname{sen}(S_i \cdot \pi)}, \quad \forall i \in \{1, 2, \ldots, m\}. \tag{3.4}$$

Essa equação é obtida a partir da geometria da Figura 3.2 aplicando-se a Lei dos Senos. Assim, o termo S_i da equação (3.4) é obtido a partir da equação $\varphi_i = S_i \cdot \pi$ (veja Figura 3.2). O termo r é a norma Euclidiana do vetor $\mathbf{u}$, e o ângulo ω_i é a inclinação entre o vetor $\mathbf{u}$ e a sua componente u_i.

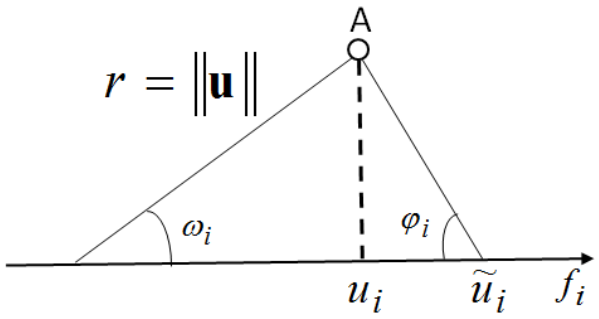


Figura 3.2 Ilustração do cálculo do CDAS.

Para apresentar um exemplo do funcionamento do CDAS será considerado a maximização de todos os objetivos ao invés da minimização, como foi feito até então. Isso será feito com o intuito de explicar o CDAS conforme apresentado por Sato *et al.* em [SAT07]. Tendo isso em vista, a Figura 3.3 apresenta a área de dominância de três soluções a, b e c em um problema em que se deseja maximizar dois objetivos. Como se trata de um problema de maximização, as áreas de dominância dessas soluções correspondem aos retângulos abaixo dessas soluções. Por exemplo, para a solução a, a sua área de dominância corresponde ao retângulo que tem a solução a no canto superior direito.

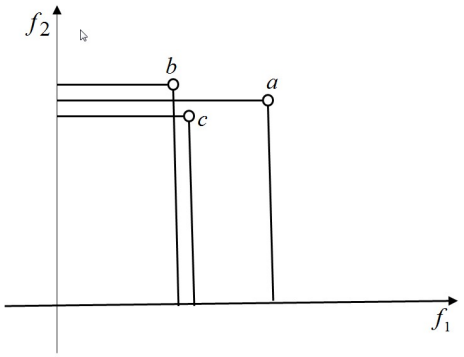


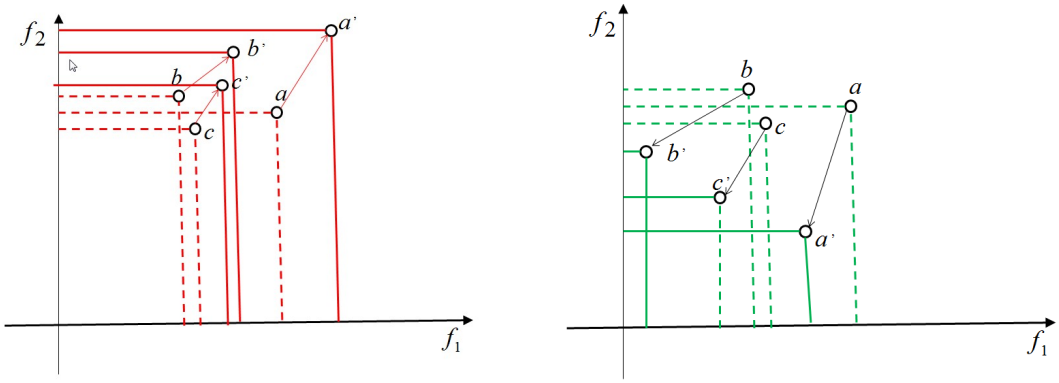
Figura 3.3 Exemplo de três soluções e suas respectivas regiões de dominância.

A partir da Figura 3.4, pode-se descrever as duas possíveis articulações do CDAS. A Figura 3.4(a) apresenta o caso da *expansão* das áreas de dominância. Nessa figura, as soluções a,

b, c representam as soluções originais, enquanto as soluções a', b', c' representam as posições das soluções após a aplicação do CDAS. As linhas sólidas demarcam os limites das áreas de dominância das soluções a', b', c', enquanto as linhas tracejadas demarcam os limites das áreas de dominância originais. Como se pode notar, a área de dominância se expandiu e com isso a solução b passou a dominar a solução c. E a solução a passou a dominar a solução b. Portanto, após a aplicação do CDAS, só existe apenas uma solução não-dominada que é a solução a. Enquanto antes haviam duas soluções não-dominadas. Esse exemplo, mostra como o CDAS diminui o número de soluções incomparáveis favorecendo a discriminação das soluções.

Por outro lado, a Figura 3.4(b) apresenta o caso da *contração* das áreas de dominância das soluções. Novamente, as linhas sólidas demarcam as áreas de dominância das soluções a', b', c', que são as posições das soluções após a aplicação do CDAS. As linhas tracejadas representam as áreas de dominância das soluções originais. Como se pode perceber por essa figura, após a contração das áreas de dominância das soluções, as soluções a, b e c se tornaram incomparáveis. Desse modo, com a contração da área de dominância, diminui-se a discriminação das soluções.

O tamanho da expansão ou contração das áreas de dominância é controlado pelo parâmetro $\mathbf{S} = [S_1, S_2, \ldots, S_m]$ especificado pelo usuário. Como se observa, é necessário atribuir um parâmetro para cada objetivo de modo que se tem um vetor de parâmetros. A escolha adequada do parâmetro $\mathbf{S}$ é dependente do problema e, portanto, seu ajuste requer do usuário conhecimento sobre a forma da Frente de Pareto, o que geralmente não acontece.



(a) Expansão do CDAS. (b) Contração do CDAS.

Figura 3.4 Articulações possíveis do CDAS.

3.2.2 Métodos de Atribuição de Aptidão Não Baseados em Dominância de Pareto

A partir da constatação da ineficácia de dominância de Pareto em lidar com MaOPs, surgiram na literatura diversos métodos de atribuição de aptidão que não usavam a dominância de Pareto diretamente e que tinham o objetivo de impor uma discriminação mais fina entre as soluções e assim aumentar a pressão em direção à Frente de Pareto. Por mais fina ou granular, entende-se que a probabilidade de duas soluções dentro de um conjunto de soluções assumirem um mesmo valor de aptidão é bastante pequena.

Esses métodos em geral atribuem um único valor a uma solução com base na comparação dessa solução com uma ou mais soluções. É importante destacar que esses métodos atuam basicamente nos estágios de seleção para sobrevivência e para variação nos MOEAs. E que o grande potencial desses métodos está em sua capacidade de discriminar ao máximo as soluções. Nesse sentido, alguns estudos na literatura passaram a comparar os métodos de atribuição propostos de acordo com a sua capacidade de gerar valores diversos de aptidão. Corne e Knowles [CK07], por exemplo, analisaram a capacidade discriminativa de alguns métodos de atribuição de aptidão. Desse estudo, eles concluíram que o método de ranqueamento médio (veja Seção 3.2.2.1) foi o que se mostrou mais efetivo entre as técnicas comparadas.

Nas próximas subseções, alguns desses métodos são explicados. Para isso, durante as explicações, será assumido que se deseja atribuir valores de aptidão às soluções de um conjunto P. Esse conjunto pode ter diferentes denominações dependendo do contexto no algoritmo em que o método é usado. Ele pode representar, por exemplo, a população interna de indivíduos ou a população externa no caso de algoritmos evolucionários ou um enxame ou arquivo externo no caso de algoritmos baseados em enxame de partículas (veja Capítulo 4).

Por fim, é importante destacar que os algoritmos apresentados nessa seção são apenas uma pequena amostra de técnicas de atribuição de aptidão. Na literatura é possível encontrar muitas outras técnicas [Gar09], tais como a relação de *Favour* [DDB01] e o ε-*Ranking* adaptativo [AT09]. Em [Gar09], Garza Fabre apresenta várias outras técnicas de atribuição de aptidão e também um estudo comparativo entre elas.

3.2.2.1 Ranqueamento Médio e Ranqueamento Máximo

O ranqueamento médio e o ranqueamento máximo foram propostos por Bentley e Wakefield [BW97, LJC09, Gar09]. O ranqueamento médio apresenta um funcionamento descrito como segue. Para um objetivo k qualquer, o conjunto de soluções P é classificado em ordem crescente segundo os valores correspondentes a esse objetivo. É importante enfatizar que a minimização dos objetivos está sendo considerada. Assim, para cada solução $\mathbf{x}_i$ de P se atribui um valor $r_k(\mathbf{x}_i)$ que corresponde a posição ocupada por ela na classificação segundo o objetivo k. Por exemplo, a solução com menor valor para o objetivo k recebe um valor igual a 1, a segunda recebe um valor igual a 2 e assim sucessivamente. Esse procedimento é repetido para todos os objetivos de modo que no final cada solução apresenta uma lista de m valores $r_k(\mathbf{x}_i)$. Finalmente, a aptidão da solução $\mathbf{x}_i$ é a média desses valores, ou seja, a aptidão final $AR(\mathbf{x}_i)$ da solução $\mathbf{x}_i$ é dada pela equação (3.5)

$$AR(\mathbf{x}_i) = \sum_{k=1}^{m} r_k(\mathbf{x}_i). \tag{3.5}$$

O ranqueamento máximo, por sua vez, consiste em atribuir para cada solução $\mathbf{x}_i$ de P o melhor (menor) valor $r_k(\mathbf{x}_i)$ entre todos m valores possíveis. Formalmente, o ranqueamento máximo é dado pela equação $MR(\mathbf{x}_i) = \min_{k=1}^{m} r_k(\mathbf{x}_i)$. É importante observar que soluções com valores menores de AR e MR são preferíveis.

3.2.2.2 Ranqueamento por ordem de eficiência

Di Pierro *et al.* [DKS07] propôs em 2007 uma estratégia de atribuição de aptidão que considera a ordem de *eficiência* das soluções. Uma solução é eficiente de ordem k se ela é não-dominada (de acordo com a dominância de Pareto) em todos os $\binom{m}{k}$ sub-espaços possíveis em que $k \leq m$. Por exemplo, considere o conjunto não-dominado $a = (1,3,6)$, $b = (2,2,9)$ e $c = (5,4,5)$ em um espaço objetivo de três dimensões e considere também a minimização dos objetivos [CK07]. Se apenas o primeiro e o segundo objetivo forem considerados, o ponto c é dominado pelos outros dois. O ponto b por sua vez é dominado por a considerado apenas o primeiro e o terceiro objetivo [CK07]. Entretanto, a permanece não-dominado para qualquer par de objetivos escolhidos. Assim, o ponto a é dito ser eficiente de ordem 2, enquanto b e c são eficientes de ordem 3. Como se pode perceber, a eficiência de ordem m ($k = m$) corresponde a dominância de Pareto.

Uma propriedade do conceito de eficiência é que se um vetor objetivo $\mathbf{x}_i$ é eficiente de ordem k então ele também é eficiente de ordem $k+1$. Assim, a ordem de eficiência de uma solução $\mathbf{x}_i$ é o valor mínimo de k para o qual $\mathbf{x}_i$ é eficiente. Nesse sentido, quanto menor for a ordem de eficiência de uma solução, melhor é a solução, pois, a grosso modo, ela impõe uma maior resistência em se tornar dominada.

Mais especificamente, Di Pierro *et al.* [DKS07] utilizaram a ordem de eficiência em conjunto com o ordenamento por não-dominação de Goldberg no NSGA II. Assim, o método de atribuição de aptidão proposto por Di Pierro *et al.* consiste em definir primeiramente as várias frentes do conjunto P usando o ordenamento por não-dominação. Posteriormente, as soluções da primeira frente recebem uma aptidão que corresponde à sua ordem de eficiência. Depois, as soluções da segunda frente em diante têm as suas aptidões (número da frente) acrescidas com a pior aptidão da primeira frente [DKS07, LJC09].

A principal desvantagem deste método é o seu custo computacional que cresce rapidamente com o número de objetivos, pois ele considera todos as combinações possíveis entre os objetivos. Dessa forma, sua aplicação é restrita a um número limitado de objetivos [Gar09].

3.2.2.3 Distância para a Melhor Solução Conhecida

A Distância para a Melhor Solução Conhecida (**DBKS**, *Distance to the Best Known Solution*) [Gar09, GPC09] consiste em primeiramente definir um ponto de referência chamado *GBEST*. O *GBEST* é formado pelos melhores valores conhecidos de cada objetivo dentro do conjunto P de soluções. Um vez que o ponto *GBEST* é determinado, a aptidão de cada solução consiste em sua distância em relação ao ponto *GBEST* no espaço objetivo. A distância de uma solução para o *GBEST* é calculada usando a distância Euclidiana.

3.2.3 Abordagens Alternativas Para Lidar com Problemas com Muitos Objetivos

Nesta seção serão apresentados outras abordagens que também foram aplicadas para o caso de problemas com muitos objetivos. Como estas técnicas estão além do escopo dessa dissertação, elas serão apresentadas superficialmente. O principal objetivo dessa seção é destacar as razões pelas quais tais abordagens não foram escolhidas para o desenvolvimento dessa dissertação.

Entre essas técnicas se destacam: (1) os algoritmos baseados em indicadores, (2) métodos de redução de dimensionalidade, e (3) os MOEAs baseados em informação de preferência.

3.2.3.1 Algoritmos Baseados em Indicadores

Os MOEAs baseados em indicadores são algoritmos que consideram a otimização de algum indicador (hipervolume, o ε-indicador aditivo [ZK04]). Um indicador é uma forma de avaliar a qualidade de um conjunto não-dominado de soluções [ITN08]. Alguns exemplos de MOEAs baseados em indicadores são o *Indicator Based Evolutionary Algorithm* (IBEA) [ZK04] e o *S-metric Selection-EMOA* (SMS-EMOA) [BNE07].

O IBEA é um algoritmo genérico e existem várias variantes dele [ITN08]. Para usá-lo, é necessário definir um indicador I que associa a cada par ordenado de soluções um valor escalar. O indicador mais utilizado no IBEA é o hipervolume [ITN08] por ser uma métrica que avalia convergência e diversidade simultaneamente [BNE07]. O hipervolume de um conjunto não-dominado A consiste, a grosso modo, na medida de Lebesgue de um conjunto formado pela união de todas as regiões dominadas pelas soluções do conjunto A. A Figura 3.5 ilustra o cálculo desse indicador entre duas soluções A e B. Como se observa por essa figura, o indicador hipervolume $I_{HV}(A,B)$ corresponde a área que é dominada por B mas não é dominada por A. Por outro lado, $I_{HV}(B,A)$ corresponde a área que é dominada por A mas não é dominada por B.

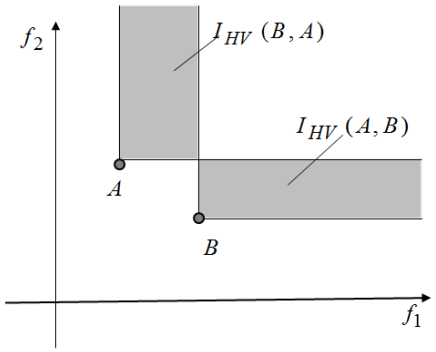


Figura 3.5 Exemplo de cálculo do I_{HV} no IBEA (adaptado de [ZK04]).

Uma solução A pode apresentar vários valores para um indicador I, pois o indicador é sempre calculado com relação a uma outra solução da população. Assim, a aptidão final $F(\mathbf{x}_i)$ de uma solução $\mathbf{x}_i$ pertencente a uma população P é dada pela equação (3.6)

$$F(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in (P - \{\mathbf{x}_i\})} -e^{\frac{-I_{HV}(\{\mathbf{x}_2\},\{\mathbf{x}_1\})}{\kappa}}, \tag{3.6}$$

em que κ é um parâmetro normalmente ajustado para $0,05$ [ZK04]. O valor $F(\mathbf{x}_i)$ corresponde a uma medida da perda de qualidade da aproximação da Frente de Pareto se a solução $\mathbf{x}_i$ é removida da população. Assim, a aptidão $F(\mathbf{x}_i)$ de um indivíduo $\mathbf{x}_i$ tem que ser maximizado, ou seja, quanto maior for o seu valor, melhor é o indivíduo.

Diferentemente, o SMS-EMOA é um algoritmo que objetiva maximizar a métrica *S* da população [BNE07]. A métrica *S* é uma outra denominação para o hipervolume [NBE05]. O SMS-EMOA é similar ao NSGA II pois ele utiliza o ordenamento por não-dominação na população para atribuir um valor de aptidão para cada solução no processo de seleção para sobrevivência. No entanto, ele utiliza como critério secundário o hipervolume da região dominada por uma solução ao invés do *crowding distance* [DPAM02]. Mais detalhadamente, esse algoritmo produz apenas uma única solução por geração e, em seguida, elimina a solução que contribui menos para o hipervolume da última frente [BNE07].

Em [WBN07], Wagner *et al.* relatou que o IBEA e o SMS-EMOA apresentaram bons resultados em MaOPs em termos de convergência e diversidade. Mais especificamente, o SMS-EMOA mostrou o melhor desempenho tanto em convergência como em diversidade com relação ao IBEA e a outros algoritmos baseados em dominância de Pareto. Entretanto, os MOEAs baseados em indicadores têm a desvantagem de não escalarem computacionalmente pois o custo elevado do cálculo do hipervolume pode tornar inviável o seu uso em problemas com muitos objetivos [ITN08].

3.2.3.2 Redução do Número de Objetivos

Uma outra solução alternativa para lidar com MaOPs é a redução de dimensionalidade [ITN08], que consiste em eliminar funções objetivos que sejam redundantes de modo a obter um número menor de objetivos. Após a eliminação dos objetivos, MOEAs baseados em dominância de Pareto podem ser utilizados para resolver o problema reduzido. Singh *et al.* [SIR11] classificou os métodos de redução de dimensionalidade para MaOPs em três grupos, a saber: (1) Métodos de Redução de Objetivos Baseados em Correlação, (2) Métodos de Redução Baseados em Estrutura de Dominância e (3) Seleção Baseada em Característica. Nesta dissertação, será discutido superficialmente a primeira abordagem. Mais detalhes sobre os outros métodos podem ser encontrados em [SIR11].

Os Métodos de Redução de Objetivos Baseados em Correlação são métodos que realizam a redução do número de objetivos com base na correlação entre os objetivos. Umas das primeiras técnicas que utilizam esse conceito é o PCA-NSGA II proposto por Deb e Saxena [DS05, SIR11]. O PCA-NSGA II utiliza uma técnica estatística de redução de dimensionalidade conhecida como Análise dos Componentes Principais (**PCA**, *Principal Component Analysis*) para redução de dimensionalidade em dados de alta dimensionalidade. No PCA-NSGA II, um conjunto representativo de soluções é obtido executando o NSGA II por um número elevado de gerações. Esse conjunto de soluções é utilizado para criar uma matriz de correlação. Os autovalores e autovetores dessa matriz são calculados e os componentes principais são determinados. A partir disso, ocorre o processo de eliminação dos objetivos. O PCA-NSGA II começa com o conjunto original de objetivos, e esses objetivos são eliminados iterativamente com base em sua contribuição para os componentes principais e com base no grau de conflitos entre eles [SIR11]. Uma vez que o conjunto de objetivos não pode ser mais reduzido, o processo de redução é encerrado. Em estudos posteriores, Saxena e Deb [SD07] substituíram o PCA, que é uma técnica de redução de dimensionalidade linear, por técnicas de redução de dimensionalidade não-lineares, tais como o *correntropy PCA* (C-PCA) e o *Maximum Variance Unfolding* (MVU). Isso levou ao desenvolvimento de dois novos algoritmos: o

C-PCA-NSGA II e o MVU-PCA-NSGA II. Todas essas técnicas mencionadas têm basicamente duas desvantagens. Elas envolvem parâmetros que são de difícil definição e, além disso, elas são de difícil implementação.

Em geral, um problema comum a todas as técnicas de redução de dimensionalidade é que o problema pode não admitir uma redução de sua dimensionalidade. Isto é, todos os seus objetivos podem ser conflitantes [ITN08], e portanto não é possível obter uma redução do número de objetivos. Além disso, não há nenhuma garantia de que o número reduzido de objetivos seja suficiente. Isso que dizer que, após a redução do número de objetivos, o problema reduzido pode apresentar ainda um número elevado de objetivos.

3.2.3.3 Incorporação de Informação de Preferência

Uma outra classe de algoritmos que tem sido aplicados em MaOPs são os algoritmos baseados em informação de preferência. Esses algoritmos utilizam algum tipo de informação de preferência fornecido pelo tomador de decisão para focar a busca em direção a uma região específica da Frente de Pareto [ITN08, FPL05]. Esses métodos são aplicados principalmente para resolver o problema da aproximação da Frente de Pareto em MaOPs.

Fleming *et al.* em [FPL05] aplicou um método de articulação de preferência [FF98] em problemas com muitos objetivos em que a região de interesse se torna cada vez menor, à medida que a busca progride e o tomador de decisão impõe suas preferências, até finalmente isolar uma solução desejada. Diferentemente, Deb e Sundar realizaram uma adaptação no NSGA II para incorporar informação de preferência baseado em pontos de referência [DS06]. Nessa abordagem, o tomador de decisão fornece um conjunto de pontos de referência e o NSGA II é modificado por meio da substituição do *crowding distance* por um critério secundário que tende a enfatizar soluções que sejam próximas de tais pontos de referência [DS06]. Assim, o objetivo desse algoritmo é encontrar soluções da Frente de Pareto que estejam próximas dos pontos de referência.

A principal crítica quanto aos algoritmos baseados em informação de preferência é o fato de que a incorporação de preferência requer algum tipo de conhecimento sobre o problema sendo resolvido, o que geralmente não é disponível [ITN08, RS06]. Para um *survey* sobre incorporação de preferência em MOEAs, o leitor é convidado a consultar [RS06].

3.3 Estado da Arte

Apesar da grande variedade de técnicas propostas para lidar com problemas com muitos objetivos, nenhum delas consegue resolver efetivamente esse tipo problema. Isto é, elas não conseguem produzir uma aproximação razoável da Frente de Pareto em termos de convergência e diversidade. De um modo geral, observou-se que, nos algoritmos estudados, existe um conflito entre convergência e diversidade. Isto significa que, se um algoritmo consegue convergir para a Frente de Pareto, ele converge apenas para algumas soluções sobre ela. Às vezes para um único ponto, como destacado por Corne e Knowles [CK07]. Por outro lado, algoritmos que produzem um maior número de soluções, geralmente não convergem para a Frente de Pareto. Como a convergência é o primeiro requisito que deve ser atendido por qualquer algoritmo mul-

tiobjetivo, nessa dissertação deu-se prioridade aos algoritmos que conseguiram bons níveis de convergência.

Entre as abordagens presentes na literatura, dois algoritmos recentes se destacam pela sua boa capacidade de convergência, mesmo em problemas difíceis e com mais de 50 objetivos conflitantes [GPC10]. Esses algoritmos são: o Algoritmo Genético Elitista baseado em Agrupamento (**CEGA**, *Clustering-based Elitist Genetic Algorithm*) e o Atribuição de Aptidão em Múltiplas Direções (**MDFA**, *Multi-Directional Fitness Assignment*). Esses algoritmos são o estado da arte em se tratando de algoritmos evolucionários para problemas com muitos objetivos. Tanto o CEGA como o MDFA são algoritmos que utilizam métodos de atribuição de aptidão especialmente projetados para o caso de problemas com muitos objetivos. O CEGA utiliza um método de atribuição de aptidão chamado Detrimento Global (**GD**, *Global Detriment*), enquanto o MDFA utiliza uma técnica de atribuição de aptidão em múltiplas direções.

O CEGA e o MDFA apresentam muitas características em comum. Basicamente, a única diferença entre eles está na forma como atribuem aptidão aos indivíduos da população. Assim, eles apresentam um fluxo de execução em comum, que é apresentado na Figura 3.6. No início de ambos os algoritmos, cria-se uma população inicial de N indivíduos no espaço de decisão segundo uma função de distribuição de probabilidade uniforme, como é prática comum em algoritmos evolucionários. Depois, esses indivíduos são avaliados utilizando o método de atribuição de aptidão específico de cada algoritmo. Em seguida, enquanto um critério de parada não é alcançado, os dois algoritmos realizam iterativamente os processos de seleção para variação, variação, e de seleção para sobrevivência, produzindo, no final, uma nova população de indivíduos que participarão da geração seguinte.

A seleção para variação, tanto no CEGA como no MDFA, ocorre por meio da seleção por torneio binário com substituição. Assim, o seguinte procedimento é realizado. Escolhem-se duas soluções da população da geração atual aleatoriamente. Essas duas soluções são comparadas entre si quanto às suas aptidões. O indivíduo com melhor aptidão é escolhido para participar do processo de variação. Uma cópia desse indivíduo é inserida em uma população em separado que é o *mating pool*, e o indivíduo original é devolvido para a população a qual ele pertencia. É importante enfatizar que o operador de seleção utilizado tem como resultado apenas um único indivíduo. O processo de seleção por torneio é repetido N vezes, de modo que no final a população de pais possui N indivíduos. Após a seleção dos pais, inicia-se o processo de variação, no qual novos indivíduos são produzidos.

Após a seleção para variação, inicia-se a etapa de **variação**. Sendo assim, o operador de cruzamento é aplicado em cada par de indivíduos do *mating pool* (população com os pais). Os pares de pais podem ser formados aleatoriamente. No caso específico da implementação utilizada nessa dissertação, os pares de pais foram produzidos seguindo a ordem da seleção por torneio da etapa anterior. Desta forma, assim que o operador de seleção por torneio é aplicado por duas vezes consecutivas, produzindo dois pais, o operador de cruzamento é imediatamente aplicado sobre esses dois pais. Nessa dissertação, utilizou-se o operador SBX como operador de cruzamento (veja Seção 2.2.3.1). Assim, para cada par de pais, dois filhos são produzidos. Assim que esses dois filhos são produzidos, o operador de mutação é imediatamente aplicado sobre cada um deles, produzindo os filhos finais que são então armazenados em uma população denominada de população de filhos. O operador de mutação utilizado nesse trabalho foi a

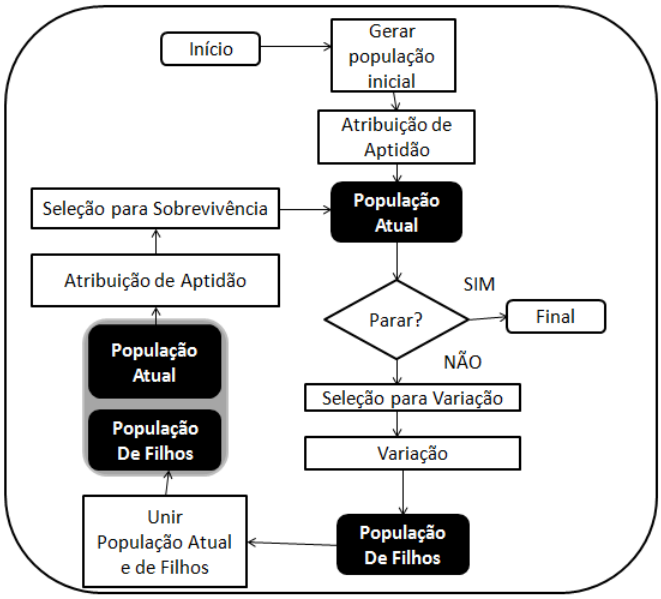


Figura 3.6 Fluxograma dos algoritmos CEGA e MDFA (adaptado de [GPC10]).

mutação polinomial (veja Seção 2.2.3.2). Desse modo, a população de filhos terá N indivíduos. Por fim, o *mating pool* é descartado.

Após o processo de variação, os indivíduos da população de filhos são avaliados e valores de aptidão são atribuídos a eles. O estágio seguinte é a seleção para sobrevivência, na qual a população da geração atual é combinada com a população de filhos, formando uma população com $2N$ indivíduos. A seleção para sobrevivência tem que ser, então, aplicado para criar uma nova população com exatamente N indivíduos para a geração seguinte. A seleção para sobrevivência ocorre de modo diferente entre o CEGA e o MDFA. Essa etapa será explicada posteriormente para cada uma deles na Seção 3.3.2. Independente disso, esse operador de seleção eliminará sempre N indivíduos de modo que a população da geração seguinte tenha o mesmo tamanho que a anterior. Finalmente, verifica-se se o critério de parada foi satisfeito ou não. O critério de parada utilizado foi o número máximo de gerações denotado por t_{max}.

A próxima seção (Seção 3.3.1) apresentará os métodos de atribuição de aptidão do CEGA e do MDFA. A Seção 3.3.2 apresentará os detalhes sobre o processo de seleção para sobrevivência de cada um deles.

3.3.1 Atribuição de Aptidão

Para melhor apresentar os métodos de atribuição do CEGA e do MDFA, essa seção será divida em duas partes. O método de atribuição de aptidão do CEGA que é mais simples que o do MDFA é apresentado primeiro na Seção 3.3.1.1. A explicação do método de atribuição do MDFA que é um pouco mais longo é deixado por último na Seção 3.3.1.2.

3.3.1.1 CEGA

O CEGA utiliza um método de atribuição de aptidão de alta granularidade denominada de Detrimento Global (**GD**, *Global Detritment*) que foi proposto em [GPC09]. Por alta granularidade entende-se que a probabilidade de dois indivíduos possuírem o mesmo valor de aptidão é bastante pequena. Desse modo, o GD impõe uma alta capacidade discriminativa entre as soluções. O GD consiste em penalizar cada indivíduo pela magnitude da diferença com que ele é superado na comparação objetivo a objetivo com respeito ao resto dos indivíduos da população. Matematicamente, o GD de uma solução $\mathbf{x}_i$ é dada pela equação (3.7).

$$GD(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in P} \sum_{k=1}^{m} \max\left[f_k(\mathbf{x}_i) - f_k(\mathbf{x}_j), 0\right] \; \forall i \in P, \tag{3.7}$$

em que m é o número de objetivos do problema a ser resolvido e P é a população de indivíduos. Um indivíduo $\mathbf{x}_i$ é dito ser melhor que outro indivíduo $\mathbf{x}_j$ se $GD(\mathbf{x}_i) < GD(\mathbf{x}_j)$.

3.3.1.2 MDFA

O MDFA utiliza um método de atribuição de aptidão cujo objetivo é guiar o processo de busca em diferentes direções no espaço objetivo simultaneamente [GPC10]. Para estabelecer diferentes direções de busca, o MDFA utiliza um conjunto de vetores de pesos uniformemente distribuídos no espaço objetivo, em que cada vector de peso corresponde a uma determinada direção. Desta forma, assume-se que diferentes vetores de peso permitem dirigir a busca em diferentes regiões no espaço objetivo, promovendo assim o espalhamento (diversidade) das soluções ao longo da Frente de Pareto. Ou seja, a assunção básica do método de atribuição de aptidão do MDFA é que um conjunto bem distribuído de pesos pode levar a uma distribuição mais uniforme das soluções na Frente de Pareto.

No inicio do MDFA, um conjunto de vetores de peso V é criado no espaço objetivo aleatoriamente. A quantidade de vetores de peso $|V|$ é especificado pelo usuário. Esse conjunto de pesos é mantido constante ao longo de todas as gerações do MDFA e é utilizado no processo de atribuição de aptidão nos indivíduos. Em [GPC10], Garza Fabre *et al.* sugeriram inicializar cada componente dos vetores de peso no intervalo de $[0,25;1]$ de acordo com uma função de distribuição de probabilidade uniforme.

Após o conjunto de pesos V ser definido, o processo de atribuição de aptidão dos indivíduo é realizado conforme o Algoritmo 6. Como pode-se observar nesse algoritmo, o processo de atribuição de aptidão do MDFA ocorre em dois estágio consecutivos. No primeiro estágio, todos os indivíduos são avaliados para todos os vetores de peso $\mathbf{v} \in V$ utilizando o método da soma ponderada tendo $\mathbf{v}$ como vetor de peso. Deve ser observado que o número de componentes do vetor de peso $\mathbf{v}$ é igual ao número de objetivos m. Assim, a soma ponderada de uma solução $\mathbf{x}_i$ com respeito a um vetor de peso $\mathbf{v} = [v_1, v_2, \ldots, v_m]$ é dado por $soma(\mathbf{x}_i, \mathbf{v}) = \sum_{k=1}^{m} v_k f_k(\mathbf{x}_i)$. Fazendo isso, observa-se que cada indivíduo da população tem exatamente $|V|$ valores de soma ponderada.

O próximo passo consiste em encontrar para cada vetor de peso $\mathbf{v}$, o indivíduo que apresenta melhor desempenho, isto, que apresenta menor valor de soma ponderada para esse vetor de peso. Seja $\mathbf{x}_i$ esse indivíduo. Então a aptidão desse indivíduo $\mathbf{x}_i$ é exatamente o valor da

soma ponderada. Entretanto, pode ocorrer de um indivíduo apresentar o menor valor de soma ponderada para mais de um vetor de peso em V. Desse modo, a aptidão final do indivíduo $\mathbf{x}_i$ é o menor valor dessas somas ponderadas. Fazendo-se isso, o Estágio 1 é encerrado. Também pode ocorrer de um indivíduo não apresentar o melhor desempenho para nenhum dos vetores de pesos. Nesse caso, ele não tem nenhum valor de aptidão (sua aptidão é infinita). Esse caso é resolvido no Estágio 2. Para um indivíduo sem aptidão, sua aptidão corresponde ao maior valor de soma ponderada entre todas as $|V|$ somas possíveis determinadas no Estágio 1. Além disso, sua aptidão é acrescida (penalizada) pelo maior valor de soma ponderada possível encontrado no Estágio 1. Isto permitirá que esses indivíduos sejam menos preferidos que os indivíduos que receberam um valor de aptidão logo no Estágio 1. Finalmente, como ocorre no CEGA, um indivíduo com menor aptidão é preferido.

Algoritmo 6: Processo de atribuição de aptidão do MDFA.

```
   /* V é o conjunto de vetor de pesos                                  */
   /* Estágio 1                                                         */
1  aptidao[x_i] ← ∞   ∀x_i ∈ P;
2  para cada v ∈ V faça
3  |  Encontre x_i tal que soma(x_i, v) < soma(x_j, v)   ∀x_j ∈ P e x_j ≠ x_i ;
4  |  se soma[x_i] < aptidao[x_i] então
5  |  |  aptidao[x_i] ← soma(x_i, v);
   /* Estágio 2                                                         */
6  pior_valor_do_estagio1 ← max(aptidao[x_i])   ∀x_i ∈ P e aptidao[x_i] ≠ ∞;
7  para cada x_i ∈ P faça
8  |  se aptidao[x_i] = ∞ então
9  |  |  aptidao[x_i] ← max_{v∈V} soma(x_i, v) + pior_valor_do_estagio1.
```

3.3.2 Seleção para Sobrevivência

Como a seleção para sobrevivência do MDFA é mais simples que a do CEGA. Ela será explicada primeiro. Basicamente, a seleção para sobrevivência do MDFA consiste em um simples esquema do tipo $(\mu + \lambda)$. Ou seja a população combinada com $2N$ indivíduos é classificada e os N indivíduos com melhores aptidões são escolhidos para próxima geração.

Enquanto a promoção de diversidade no MDFA é realizado no processo de atribuição de aptidão, a promoção de diversidade no CEGA ocorre na seleção para sobrevivência. Isso é alcançado por meio de uma técnica de agrupamento [GPC10]. A técnica de agrupamento utilizada é o agrupamento hierárquico aglomerativo; e ele é aplicado no espaço de variáveis de decisão, ou seja, o vetor de variáveis de decisão de cada solução corresponde a um ponto de dado no algoritmo de agrupamento.

O algoritmo hierárquico aglomerativo funciona do seguinte modo: inicialmente, cada ponto de dado representa um grupo diferente; em seguida, a cada iteração do algoritmo de agrupamento, os dois grupos mais próximos, isto é, com maior similaridade segundo alguma métrica de distância entre grupos, são combinados em um único grupo; esse processo é repetido

até que o número de grupos N_{grupos} ($N_{grupos} \in [2, N]$) especificado pelo usuário seja atingido. A Figura 3.7 apresenta três iterações da execução do algoritmo de agrupamento aglomerativo hierárquico. Para se avaliar a distância entre dois grupos se utilizou o método da ligação média (*average linkage*). Ou seja, a distância entre dois grupos C_1 e C_2 é a média das distâncias entre cada par de pontos $\mathbf{p}_i$ e $\mathbf{p}_j$ tal que $\mathbf{p}_i \in C_1$ e $\mathbf{p}_j \in C_2$. A métrica de distância utilizada para avaliar a distância entre dois pontos foi a distância Euclidiana.

Uma vez que o método de agrupamento hierárquico aglomerativo foi explicado. O processo de seleção para sobrevivência do CEGA é descrito pelas seguintes etapas:

1. Calcular a aptidão de cada indivíduo da população combinada (população da geração atual mais população de filhos) utilizando o método *GD*;

2. Aplicar o algoritmo de agrupamento hierárquico aglomerativo sobre a população combinada para formar N_{grupos} grupos;

3. Selecionar o melhor indivíduo de cada grupo para a população seguinte. O melhor indivíduo em um grupo é aquele que apresenta o melhor DBKS (veja Seção 3.2.2.3) considerando apenas os indivíduos desse grupo;

4. Após a etapa anterior, N_{grupos} indivíduos já estão selecionados para formarem a população da geração seguinte. Se $N_{grupos} < N$, então os $N - N_{grupos}$ indivíduos necessários para completar a população da próxima geração são selecionados com base no GD, ou seja, os $N - N_{grupos}$ indivíduos dos $2N - N_{grupos}$ indivíduos remanescentes com menor GD são selecionados para completar a população.

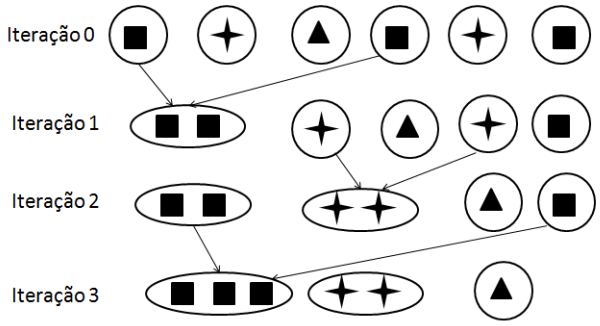


Figura 3.7 Execução do algoritmo de agrupamento aglomerativo hierárquico (adaptado de [GPC10]).

3.4 Considerações Finais

Neste capítulo, foram discutidos os principais conceitos e técnicas sobre otimização de muitos objetivos. Foi apresentando as dificuldades e deficiências dos métodos baseados em dominân-

cia de Pareto, bem como as principais soluções presentes na literatura para lidar com esses problemas. Entretanto, muito pouco tem sido feito no sentido de adaptar os métodos baseados em enxame de partículas para lidar com esses problemas. O próximo capítulo apresenta algumas dessas adaptações bem como o algoritmo desenvolvido nesta dissertação.

CAPÍTULO 4

Enxame de Partículas para Problemas com Muitos Objetivos

" *A única maneira de teres sensações novas*
é construíres-te uma alma nova. "
– Fernando Pessoa

ESTE capítulo tem como objetivo apresentar a fundamentação teórica sobre algoritmos multiobjetivos baseados em enxame de partículas. Na Seção 4.1, o algoritmo de otimização por enxame de partículas (**PSO**, *Particle Swarm Optimization*) é descrito e brevemente analisado. A Seção 4.2 descreve os algoritmos de otimização multiobjetivos baseados em enxame de partículas utilizados neste trabalho.

Como foi discutido no Capítulo 3, problemas com muitos objetivos oferecem muitos desafios aos algoritmos, em particular aos algoritmos baseados em dominância de Pareto. Nesse capítulo, também foi apresentado os principais métodos adotados na literatura para lidar com essa classe de problemas. Esses métodos têm sido aplicados de forma bem sucedida em algoritmos evolucionários. No entanto, existem poucos trabalhos na literatura que investigam a aplicação desses métodos em algoritmos baseados em enxame de partículas. A Seção 4.3 revisa alguns algoritmos baseados em enxame de partículas criados para lidar com problemas com muitos objetivos. Essa seção também apresenta as principais desvantagens dessas técnicas. Por fim, esse capítulo é encerrado com as considerações finais na Seção 4.4.

4.1 Otimização por Enxame de Partículas

O algoritmo de Otimização por Enxame de Partículas (**PSO**, *Particle Swarm Optimization*) é um algoritmo estocástico para otimização de funções não-lineares de alta dimensionalidade e com variáveis contínuas [BK07, AEH09]. O PSO foi proposto por Kennedy e Eberhart em 1995 [KE95], e foi inspirado no comportamento social de organismos biológicos, mais especificamente na habilidade de algumas espécies de animais de trabalhar em conjunto para localizar boas regiões com fontes de alimento, assim como ocorre em cardumes e em bandos de pássaros [BK07]. Desde de sua publicação, o PSO vem sendo aplicado com sucesso na solução de problemas da ciência e da engenharia devido à sua simplicidade, eficácia e robustez [AEH09, Eng07, VE06].

Para realizar a otimização de uma função objetivo, o PSO utiliza um enxame de partículas, em que cada partícula representa uma solução candidata. A partícula no PSO equivale a um

indivíduo em um algoritmo evolucionário. Essas partículas se movem em um espaço de busca de alta dimensionalidade buscando por boas soluções usando uma combinação de atração para a melhor solução encontrada até o momento pela própria partícula; e de uma atração para a melhor solução encontrada até o momento pela vizinhança da partícula. A vizinhança de uma partícula é definida como o conjunto de partículas com as quais ela é capaz de se comunicar [BK07]. No PSO, cada partícula i do enxame S tem uma posição $\mathbf{x}_i$, que consiste de um vetor de n dimensões cujos componentes representam os parâmetros da função objetivo, uma velocidade $\mathbf{v}_i$ e uma posição $\mathbf{y}_i$ chamada de *melhor posição pessoal* ou *pbest*, que representa a melhor posição encontrada pela partícula até o momento. Nesta dissertação, o termo líder cognitivo será usado no lugar desses termos (melhor posição pessoal e *pbest*) para se referir a posição $\mathbf{y}_i$ [1].

No início do PSO, as partículas do enxame são inicializadas em posições aleatórias no espaço de busca segundo uma distribuição de probabilidade uniforme, assim como ocorre em muitos algoritmos evolucionários. Em seguida, a posição $\mathbf{x}_i(t)$ de cada partícula i na iteração t é modificada por uma velocidade estocástica $\mathbf{v}_i(t)$ que depende da distância que a partícula está da sua melhor solução conhecida e da distância para melhor solução conhecida dentro de sua vizinhança. Cada partícula $i \in S$ se movimenta em cada dimensão $j \in \{1,2,\ldots,n\}$ do espaço de busca em um instante discreto de tempo t segundo a equação (4.1) e equação (4.2),

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)], \tag{4.1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{4.2}$$

em que o termo $\hat{\mathbf{y}}_i(t)$ consiste na melhor posição encontrada até o momento t pelas partículas vizinhas da partícula i no enxame. Nesta dissertação, essa posição será denominada de líder social por questões de coerência com a Seção 4.2. Quando a vizinhança das partículas consiste no enxame inteiro, a posição $\hat{\mathbf{y}}_i(t)$ é denominada de *gbest*. Os termos c_1 e c_2 são constantes reais positivas conhecidas como coeficientes de aceleração. Os termos $r_{1j}(t)$ e $r_{2j}(t)$ são valores aleatórios retirados de uma distribuição de probabilidade uniforme no intervalo $[0,1]$, ou seja, $r_{1j}(t), r_{2j}(t) \sim U[0,1]$.

O vetor velocidade é o responsável por guiar o processo de otimização, e para isso ele utiliza tanto o conhecimento adquirido individualmente pela partícula quanto o conhecimento adquirido pela partícula a partir da comunicação com sua vizinhança. O termo $c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)]$ da equação de atualização da velocidade é a componente cognitiva e representa a experiência da partícula. Essa componente é a responsável pela tendência que a partícula tem de voltar para a melhor solução encontrada por ela no passado. O termo $c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)]$, por sua vez, é conhecido como a componente social da equação da velocidade, e representa o conhecimento coletivo do enxame, sendo a responsável por atrair cada partícula para a melhor solução encontrada por alguma partícula de sua vizinhança. Um exemplo de movimento de uma partícula em um espaço de busca bidimensional está apresentado na Figura 4.1. Como se observa, o vetor velocidade consiste basicamente de uma soma vetorial das componentes cognitiva e social com a inércia da partícula. A **inércia** (velocidade anterior da partícula) funciona

[1] O termo líder cognitivo é utilizado no contexto em que o PSO é aplicado em problemas multiobjetivos como será visto na Seção 4.2.

como uma memória da direção da velocidade anterior da partícula. Esse termo é importante pois ele evita que a partícula sofra alterações bruscas na direção de sua velocidade [Eng07]. Desta forma, se a partícula se movimentava em direção a uma boa região, essa direção não é completamente modificada em virtude da descoberta de um novo líder social. A **componente cognitiva** é importante pois ela atrai a partícula na direção da melhor posição encontrada por ela desde o início da busca. Com isso, espera-se que a partícula encontre uma boa posição que possivelmente esteja próximo do seu líder cognitivo atual. A **componente social**, por sua vez, é a principal componente na equação da velocidade das partículas, pois é por meio dela que as partículas compartilham informação a respeito das melhores posições encontradas por elas desde o início do processo de busca.

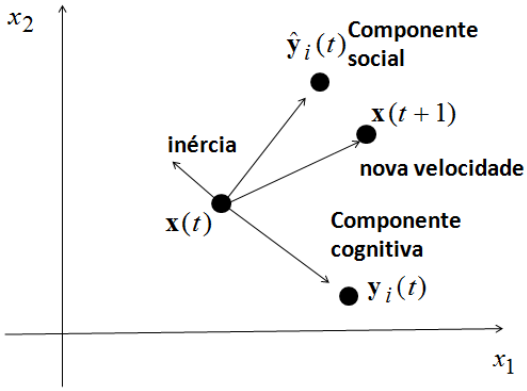


Figura 4.1 Exemplo de movimento de uma partícula em um espaço de busca com 2 dimensões.

O pseudo-código do PSO é apresentado no Algoritmo 7. No início do algoritmo, as posições das partículas, velocidades e seus líderes cognitivos são inicializados. Geralmente as posições das partículas são inicializadas segundo uma distribuição uniforme, entre o limite inferior $x_{min,j}$ e superior $x_{max,j}$, para toda dimensão j do espaço de busca. Normalmente, as partículas iniciam com velocidade igual a zero [Eng07]. Os líderes cognitivos são inicializados com as posições iniciais das partículas. É importante destacar que essa estratégia de inicialização foi seguida por todas as abordagens baseadas em PSO utilizadas nesse trabalho. Em seguida, enquanto um critério de parada não é atingido as seguintes etapas são realizadas. Primeiramente, os líderes cognitivos das partículas são atualizados. Depois, o líder social dos enxame é determinado. Por fim, a velocidade e a posição de cada partícula são atualizadas.

O PSO é um algoritmo de busca, e como tal sua eficácia depende do bom balanceamento entre *exploração* e *explotação*. A exploração (conhecida também como busca em largura ou busca global) consiste na capacidade de um algoritmo de busca de explorar diferentes regiões no espaço de busca com a finalidade de encontrar um bom ótimo [Eng07, Tre03]. A explotação (também conhecida como busca em profundidade ou busca local), por sua vez, é a capacidade do algoritmo de concentrar a busca em torno de uma região promissora do espaço de busca [Eng07, Tre03]. Em outras palavras, na exploração o algoritmo realiza uma busca "grosseira" por todo espaço de busca atrás de uma região que, possivelmente, possa conter a solução ótima do problema. Na fase de explotação, por outro lado, o algoritmo se concentra na região

Algoritmo 7: Pseudo-código do PSO.

```
Inicialize o enxame de partículas no espaço de busca;
enquanto condição de parada não for alcançado faça
    Atualize os líder cognitivo das partículas;
    Atualize o líder social das partículas;
    para cada i ∈ S faça
        para cada j ∈ {1,2,...n} faça
            Atualize velocidade usando equação (4.1);
            Atualize posição usando equação (4.2);
Retorne melhor solução encontrada.
```

promissora e realiza uma busca mais "fina" nessa região a fim de encontrar a solução ótima do problema [Tre03]. No PSO, o balanceamento entre exploração e explotação é tratado pela equação de velocidade [Eng07].

No PSO original, proposto por Kennedy e Eberhart, foi verificado que a velocidade das partículas alcançavam rapidamente valores muito altos, especialmente para partículas longe de seu líder cognitivo e do seu líder social. Consequentemente, devido às componentes cognitivas e sociais, essas partículas apresentam uma mudança acentuada em sua velocidade, o que fazia com que as partículas ultrapassem as fronteiras do espaço de busca [Eng07]. Para evitar esse aumento abrupto da velocidade, passou-se a limitá-la por um valor máximo [Eng07]. Assim, se a velocidade de uma partícula em uma dimensão j excede um valor máximo denotado por $V_{max,j}$, a sua velocidade é definida como sendo $V_{max,j}$. Sendo assim, a velocidade de cada partícula $i \in S$ para cada dimensão $j = \{1,2,\ldots,j\}$ é ajustada, antes de modificar a posição da partícula, usando a equação (4.3):

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1), & \text{se } |v'_{ij}(t+1)| < V_{max,j}, \\ \text{sinal}(v'_{ij}(t+1)) \times V_{max,j}, & \text{se } |v'_{ij}(t+1)| \geq V_{max,j}, \end{cases} \tag{4.3}$$

em que $v'_{ij}(t+1)$ é calculada usando equação (4.1) e $v_{ij}(t+1)$ é a velocidade da partícula usada na equação (4.2). A função sinal(x) é definida como

$$\text{sinal}(x) = \begin{cases} 1, & \text{se } x > 1, \\ 0, & \text{se } x = 0, \\ -1, & \text{se } x < -1. \end{cases} \tag{4.4}$$

O valor da velocidade máxima permitida para cada partícula $V_{max,j}$ é dependente do problema. A estratégia geralmente adotada para escolher cada componente $V_{max,j}$ do vetor velocidade máxima é defini-lo como uma fração do espaço de busca [Eng07]. Ou seja,

$$V_{max,j}(t+1) = \delta(x_{max,j} - x_{min,j}), \tag{4.5}$$

em que $x_{max,j}$ e $x_{min,j}$ são respectivamente os valores máximos e mínimos do espaço de busca na dimensão j, e δ é um parâmetro tal que $\delta \in (0,1]$. É importante enfatizar que o fato de usar a

velocidade máxima não evita que as partículas escapem dos limites do espaço de busca. Nesse caso, existem várias abordagens na literatura para tratar esse caso. Contudo, nessa dissertação, para todos os algoritmos baseados em enxame partículas, a seguinte estratégia foi utilizada. Caso uma partícula i ultrapasse os limites do espaço de busca na dimensão j, utiliza-se a equação (4.6):

$$x_{ij}(t) = \begin{cases} x_{max,j} & \text{se } x'_{ij}(t) > x_{max,j}, \\ x_{min,j} & \text{se } x'_{ij}(t) < x_{min,j}, \end{cases} \tag{4.6}$$

em seguida a velocidade da partícula é invertida usando equação (4.7):

$$v_{ij}(t) = \begin{cases} -v'_{ij}(t) & \text{se } x'_{ij}(t) > x_{max,j} \text{ ou se } x'_{ij}(t) < x_{min,j}, \\ v'_{ij}(t) & \text{caso contrário,} \end{cases} \tag{4.7}$$

em que $x'_{ij}(t)$ e $v'_{ij}(t)$ são, respectivamente, a posição e a velocidade da partícula i antes de se aplicar a equação (4.6) e equação (4.7).

Para permitir um melhor balanceamento e controle entre exploração e explotação no PSO, Shi e Eberhart [SE98a] introduziram um termo denominado peso de inércia ω na equação de velocidade do PSO. O termo ω controla a contribuição da velocidade anterior da partícula em sua velocidade atual. Desse modo, a equação da velocidade do PSO com o termo ω é dado pela equação (4.8):

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)]. \tag{4.8}$$

O parâmetro ω tem uma grande influência sobre o desempenho do PSO. Uma escolha adequada da inércia é muito importante para assegurar a convergência do PSO e um bom balanceamento entre exploração e explotação [Eng07]. Para $\omega \geq 1$, a velocidade das partículas aumenta ao longo do tempo e o enxame diverge [Eng07]. Caso $\omega < 1$, as velocidades das partículas diminuem ao longo do tempo até suas velocidades atingirem o valor zero, dependendo dos valores dos coeficientes de aceleração [Eng07]. Grandes valores de ω favorecem a exploração e, portanto, a diversidade, enquanto um valor pequeno de ω, por sua vez, favorece a explotação. O valor adequado para ω, assim como a velocidade máxima, é dependente do problema [SE98b]. Van de Bergh e Engelbrecht [VE06] mostraram que uma estratégia para a seleção de ω é escolhê-lo em conjunção com os coeficientes c_1 e c_2 de modo que a seguinte condição seja verdade:

$$\omega > \frac{1}{2}(c_1 + c_2) - 1. \tag{4.9}$$

Satisfazendo-se a equação (4.9), as trajetórias das partículas podem ser convergentes, caso contrário, elas podem assumir um comportamento cíclico e divergente [VE06].

Uma outra estratégia para lidar com a dificuldade para a seleção adequada do parâmetro ω, é modificá-lo dinamicamente com o tempo ao invés de usar um valor estático. Essa abordagem geralmente começa com um valor grande para ω que diminui linearmente ao longo do tempo. Nesse caso, tem-se um valor $\omega(t)$ em cada instante t que é dado pela equação (4.10):

$$\omega(t) = (\omega(0) - \omega(t_{max}))\frac{(t_{max} - t)}{t_{max}} + \omega(t_{max}), \tag{4.10}$$

em que t_{max} é o número total de iterações, $\omega(0)$ (geralmente 0,9) é o valor inicial do peso de inércia e $\omega(t_{max})$ (geralmente 0,4) é o valor final do peso de inércia [Eng07].

Clerc e Kennedy [CK02] desenvolveram uma abordagem muito similar ao peso de inércia para equilibrar o compromisso entre exploração e explotação. Eles introduziram um termo de constrição denotado por χ. Com esse termo, a equação da velocidade é modificada para:

$$v_{ij}(t+1) = \chi \left\{ v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \right\}, \tag{4.11}$$

em que

$$\chi = \frac{2}{\left| 2 - \phi - \sqrt{\phi(\phi - 4)} \right|}, \tag{4.12}$$

com $\phi = c_1 + c_2$, sendo $\phi > 4$ [BK07, Van06].

Além das variações apresentadas para o PSO, existem muitas outras na literatura. Em particular, assim como ocorreu com EA, existem adaptações do PSO para problemas multiobjetivos. Essa adaptação não é imediata, sendo que algumas questões precisam ser tratadas adequadamente para que o PSO possa ser aplicado efetivamente em problemas com múltiplos objetivos. Na Seção 4.2, o processo de adaptação do PSO em problemas multiobjetivos é tratado em detalhes.

4.2 Algoritmos Multiobjetivos Baseados em Enxame de Partículas

O sucesso do PSO como otimizador mono-objetivo em muitas aplicações, principalmente em problemas com espaço de busca contínuo, motivou a sua extensão para problemas multiobjetivos [RSC06]. Além disso, o PSO apresenta outras características que o tornam um bom candidato para resolução de problemas dessa natureza, tais como o fato dele ser baseado em população e sua simplicidade relativa [RSC06]. A primeira tentativa de estender o PSO para problemas multiobjetivo foi proposta por Moore e Chapman em um manuscrito não publicado em 1999 [RSC06]. Desde de então, outros trabalhos foram publicados na literatura. De modo que qualquer uma dessas extensões, em suas mais diversas variações, é denominada genericamente de Algoritmo Multiobjetivo Baseado em Enxame de Partícula (**MOPSO**, *Multiobjective Particle Swarm Optimization*) [RSC06]. Além das características já mencionadas, os MOPSOs apresentam vantagens com relação aos MOEAs, seus principais competidores, tais como rápida capacidade de convergência [RSC06, CPL04], simplicidade, e a presença de menos parâmetros de controle que as abordagens evolucionárias [MS08].

Assim como ocorre com MOEAs, qualquer MOPSO deve atender a dois requisitos básicos, a saber: convergência e diversidade. O desenvolvimento de um MOPSO para atender esses dois requisitos não é uma tarefa trivial, de modo que a concepção de um MOPSO envolve várias decisões de projeto que, quando não bem definidas, podem levá-lo a não convergir na maioria de suas execuções, ou fornecer um conjunto de soluções com pouca diversidade. Nesse último caso, isto pode significar que o MOPSO converge para um único ponto da Frente de Pareto ou que ele cobre apenas um pequena região da Frente de Pareto [RSC06].

Desse modo, para conceber um MOPSO efetivo, basicamente duas questões devem ser resolvidas [ZQL+11, RSC06]. A primeira delas é como manter as soluções não-dominadas encontradas ao longo das iterações do MOPSO. Como já foi exposto, em problemas multiobjetivos não existe um conceito de otimalidade bem definido. Entretanto, nos MOPSOs mais promissores a otimalidade de Pareto é comumente utilizada como critério de otimalidade. Desse modo, em cada iteração do algoritmo, é necessário armazenar as soluções não-dominadas encontradas no decorrer da busca do MOPSO. Para esse fim, os MOPSOs mais promissores geralmente utilizam um arquivo externo (AE) responsável por manter essas soluções no decorrer do processo de busca, de modo que as soluções presentes nessa população são sempre não-dominadas com respeito à todas as soluções encontradas desde do inicio do algoritmo [MS08, ZQL+11]. Explicando melhor, considere que $W^{(t)}$ seja um conjunto formado por todas as soluções encontradas pelo MOPSO desde o início da busca até o instante t. Então o AE no instante t denotado por $AE^{(t)}$ contém apenas as soluções localmente não-dominadas com respeito a $W^{(t)}$. Por meio desse procedimento, a busca em um MOPSO pode progredir em direção à Frente de Pareto.

A segunda questão a ser resolvida é como selecionar o líder social e o líder cognitivo de cada partícula, uma vez que existem várias soluções consideradas ótimas [ZQL+11, RSC06]. A abordagem frequentemente usada para determinar esses líderes é, primeiramente, dar prioridade às soluções não-dominadas em relação àquelas que são dominadas, assim como ocorre em alguns MOEAs. Nesse sentido, o AE pode ser utilizado como fonte de potenciais líderes para as partículas do enxame. Entretanto, no AE existe apenas soluções incomparáveis, de modo que é necessário definir outros critérios e mecanismos para se impor algum tipo de preferência sobre essas soluções e assim atribuir um líder social para cada partícula do enxame. Uma das formas de alcançar esse objetivo é definir uma medida de qualidade que de algum modo quantifique a qualidade de uma solução. Como ao se projetar um MOPSO, um dos objetivos é a preservação da diversidade das soluções, um modo natural de se impor uma preferência entre duas soluções incomparáveis é usar um estimador de densidade. Nesse caso, uma solução é superior a outra se ela se localiza em uma região de baixa densidade no espaço objetivo. Na literatura, o *crowding distance* é utilizado em alguns MOPSOs como estimador de densidades, tais como o MOPSO-CDR [SPBF09], o SMPSO [DGNN+09], e o MOPSO-CD [RN05].

No que se refere a definição do líder cognitivo, a dominância de Pareto geralmente é utilizada [RSC06]. Assim, se a nova posição da partícula domina seu líder cognitivo, essa posição da partícula se torna seu novo líder cognitivo. Caso o líder cognitivo domine a atual posição, o líder cognitivo permanece inalterado. Entretanto, caso eles sejam incomparáveis, algum outro mecanismo deve ser utilizado para decidir quem é o novo líder cognitivo.

Um ponto importante a ser enfatizado a respeito do AE, é que devido à restrições computacionais ele tem um tamanho limitado [RSC06]. Desse modo, é importante eliminar soluções do AE quando este excede um número máximo de soluções. Como todas as soluções no AE são incomparáveis, uma questão que surge é quais soluções devem ser mantidas e quais devem ser eliminadas. Novamente, em alguns algoritmos, o requisito de diversidade é levando em consideração. Com isso, soluções em regiões de baixa densidade são preferidas em relação àquelas em áreas densas. Em alguns algoritmos, o mesmo operador de densidade utilizado na escolha dos líderes sociais é utilizado para decidir quais soluções serão preservadas no AE e

quais serão descartadas. O processo de eliminação de soluções no AE quando um determinado número de soluções é excedido é denominado de poda do AE.

Como já foi mencionado, um dos problemas inerentes aos MOPSOs é a convergência prematura [RSC06]. A convergência prematura é decorrente da rápida perda de diversidade do enxame. Desse modo a promoção da diversidade em um MOPSO é uma questão importante do projeto de um MOPSO efetivo [RSC06]. Isso é alcançado de dois modos. Um deles já foi mencionado e consiste em preservar soluções no espaço objetivo que estejam em áreas esparsas. O outro modo é por meio da preservação da diversidade das soluções no espaço de decisão. Isso é alcançado por meio de um operador de turbulência, também denominado de mutação [RSC06]. Esse operador tem o objetivo de fazer com que regiões inexploradas do espaço de busca sejam alcançadas por meio da modificação aleatória da posição das partículas do enxame.

Por fim, como já foi exposto, existem muitas variações possíveis para o MOPSO, cada uma com suas vantagens e desvantagens. Nesse trabalho, dois MOPSOs estado da arte foram escolhidos como base para o desenvolvimento desta dissertação: o SMPSO [NDG$^+$09, DGNN$^+$09] e o MOPSO-CDR [SPBF09]. Esses algoritmos apresentam muitas características em comum, de modo que os dois apresentam o mesmo funcionamento geral apresentado no pseudo-código do Algoritmo 8. A diferença entre eles está no modo como eles definem os líderes sociais e cognitivos, na forma como a poda é realizada e no operador de turbulência utilizado. Tendo isso em vista, ambos os algoritmos começam inicializando as posições das partículas no espaço de busca de acordo com uma função de distribuição de probabilidade uniforme, como é usual em algoritmos evolucionários. Todas as partículas começam com velocidade igual a zero, e os líderes cognitivos das partículas são iguais as suas posições atuais. Em seguida, após a inicialização, o SMPSO e o MOPSO-CDR entram em um ciclo em que as seguintes operações são realizadas até que seja atingido um número máximo de iterações (t_{max}). Primeiro, os líderes cognitivos são determinados. Novamente, o SMPSO e o MOPSO-CDR apresentam maneiras diferentes de fazer essa escolha. Depois da seleção dos líderes cognitivos, os líderes sociais são determinados. O SMPSO e o MOPSO-CDR utilizam esquemas diferentes para isso. O SMPSO utiliza torneio binário para selecionar os líderes, enquanto o MOPSO-CDR utiliza seleção por roleta. Após a escolha dos líderes sociais e cognitivos, ambos atualizam a velocidade e a posição das partículas e logo em seguida aplicam um operador de turbulência sobre elas. Posteriormente, as partículas são avaliadas usando o problema multiobjetivo sendo resolvido. As novas soluções não-dominadas encontradas pelo enxame são então inseridas no AE e o mecanismo de poda é acionado se necessário. Os detalhes sobre cada um dos algoritmos são dados na Seção 4.2.1 para o SMPSO a na Seção 4.2.2 para o MOPSO-CDR.

4.2.1 SMPSO: Algoritmo Baseado em Enxame de Partículas com Constrição de Velocidade

Em 2009, Durillo *et al.* [DGNN$^+$09] realizaram um estudo comparativo de seis MOPSOs, a saber: NSPSO, SigmaMOPSO, OMOPSO, AMOPSO, MOPSOpd e o MOCLPSO. A partir dessa análise comparativa, eles constataram que todos os MOPSOs eram incapazes de resolver satisfatoriamente alguns problemas com multimodalidade, ou seja, problemas com múltiplas Frentes de Pareto locais nos quais um algoritmo pode estagnar. A partir dessa observação, Durillo *et al.* [DGNN$^+$09] perceberam que esse problema ocorria devido ao fato de que, nos

Algoritmo 8: Pseudo-código do SMPSO e MOPSO-CDR.

```
1  Inicializa partículas no espaço de busca;
2  Inicialize o arquivo externo AE;
3  Qualifique soluções em AE;
4  para t de 1 a t_max faça
5      Atualize líderes sociais das partículas;
6      Atualize velocidade das partículas;
7      Atualize posição das partículas;
8      Aplique operador de turbulência;
9      Avalie as partículas usando o problema multiobjetivo;
10     Atualize líderes cognitivos das partículas;
11     Atualize arquivo externo AE;
12     Qualifique soluções em AE;
13     Realize poda do arquivo externo AE se necessário;
```

algoritmos estudados, as velocidades das partículas atingiam valores muito altos, resultando em um fenômeno conhecido como "explosão do enxame". Esse fenômeno consiste em movimentos errôneos das partículas para além dos limites do espaço de busca. Ou seja, as partículas escapavam do espaço de busca. Como foi mencionado anteriormente sobre o PSO, uma das abordagens utilizadas para tratar o caso em que as partículas escapam do espaço de busca é reverter a suas velocidades. Com isso, algumas partículas do enxame oscilam entre esses limites do espaço de busca ao longo das iterações, e, portanto, regiões do espaço de busca deixavam de ser exploradas.

Para contornar essa situação, Durillo *et al.* [DGNN$^+$09] realizaram uma adaptação no OMOPSO, o algoritmo que apresentou melhores resultados em termos de convergência e diversidade dentre os algoritmos comparados. A adaptação realizada consistiu em utilizar um mecanismo de constrição de velocidade para evitar que a velocidade das partículas assumissem valores muito altos ao longo das iterações. Esse mecanismo levou o surgimento de um novo algoritmo chamado Algoritmo Baseado em Enxame de Partículas com Constrição de Velocidade (**SMPSO**, *Speed-constrained Multi-objective PSO*) [NDG$^+$09].

Como já foi mencionado, o SMPSO segue o mesmo fluxo de execução, que o Algoritmo 8. No SMPSO, a atualização dos líderes cognitivos ocorre de um modo bastante simples. Primeiramente, cada partícula compara, em termos de dominância de Pareto, seu líder cognitivo presente em sua memória com a sua posição atual. Se a posição atual domina o líder cognitivo, então a posição atual se torna o novo líder cognitivo da partícula. Caso a posição atual seja dominada pelo líder cognitivo da partícula, nada acontece. Por fim, caso essas soluções sejam incomparáveis, a posição atual se torna o novo líder cognitivo (regra da solução mais recente).

No SMPSO, a atualização do líder social é também muito simples. Basicamente, a seleção do líder social para cada partícula consiste na realização de um torneio binário entre as soluções do arquivo externo. Isto é, para se determinar o líder social de uma partícula i a seguinte operação é realizada. Duas soluções são escolhidas aleatoriamente do arquivo externo e são então comparadas entre si usando o *crowding distance* como aptidão. A solução com maior

crowding distance (menor densidade) é então escolhida como o líder social da partícula i. Esse procedimento é repetido para todas as partículas do enxame.

A atualização da velocidade, por sua vez, é definida como segue. Primeiramente, a velocidade de cada partícula é determinada conforme a equação (4.13):

$$\begin{aligned} v_{ij}(t+1) = \chi \ \{ & \omega \cdot v_{ij}(t) + c_1 \cdot r_1(t) \cdot [y_{ij}(t) - x_{ij}(t)] \\ & + c_2 \cdot r_2(t) \cdot [\hat{y}_{ij}(t) - x_{ij}(t)] \}. \end{aligned} \tag{4.13}$$

Nessa equação, χ é o coeficiente de constrição obtido do fator de constrição originalmente proposto em [CK02]. No SMPSO esse termo é calculado como:

$$\chi = \begin{cases} \frac{2}{2-\varphi-\sqrt{\varphi^2-4\varphi}}, & \text{se } \varphi > 4, \\ 1, & \text{se } \varphi \leq 4, \end{cases} \tag{4.14}$$

em que φ é dado pela equação (4.15)

$$\varphi = c_1 + c_2. \tag{4.15}$$

Como se pode observar por essa equação, o SMPSO utiliza tanto o peso de inércia ω como o fator de constrição χ para determinar a velocidade das partículas. Isso é uma diferença em particular com outros MOPSOs. Além disso, no SMPSO os coeficientes de aceleração c_1 e c_2 não são fixos. Em cada iteração do SMPSO, os valores de c_1 e c_2 são amostrados de uma função de distribuição de probabilidade uniforme. Ou seja, $c_1 \sim U(c_1^{min}, c_1^{max})$ e $c_2 \sim U(c_2^{min}, c_2^{max})$ em que c_1^{min} e c_1^{max} são os valores mínimos e máximos para o coeficiente c_1, enquanto c_2^{min} e c_2^{max} são os valores mínimos e máximos para o coeficiente c_2. Após o calculo da velocidade, o SMPSO limita a velocidade v_{ij} de cada partícula i em cada dimensão $j \in \{1, 2, \ldots n\}$ por meio da seguinte equação (4.16):

$$v_{ij}(t) = \begin{cases} \Delta_j, & \text{se } v_{ij}(t) > \Delta_j \\ -\Delta_j, & \text{se } v_{ij}(t) \leq -\Delta_j \\ v_{ij}(t), & \text{caso contrário,} \end{cases}, \tag{4.16}$$

com

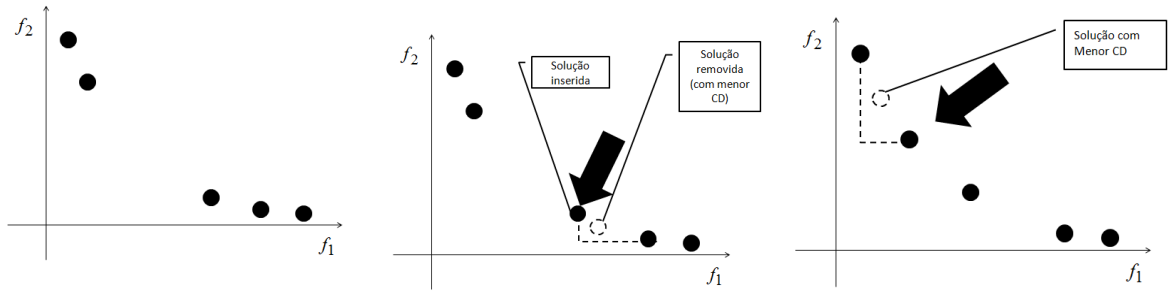
$$\Delta_j = \frac{(x_{max,j} - x_{min,j})}{2}, \quad \forall j \in \{1, 2, \ldots, n\}, \tag{4.17}$$

em que $x_{min,j}$ é o limite inferior do espaço de busca na dimensão j e $x_{max,j}$ é o limite superior.

O operador de turbulência no SMPSO consiste em aplicar a mutação polinomial sobre um percentagem pré-definida α de partículas com uma probabilidade p_m.

Por fim, o mecanismo de poda do SMPSO é realizado do seguinte modo. As soluções do enxame são inseridas uma de cada vez no arquivo externo (AE). Ao se tentar inserir uma solução, as soluções do AE dominadas por essa solução são removidas. Ou, caso alguma solução do AE domine a solução em questão, essa solução (a solução que está sendo inserida) é descartada e a próxima solução do enxame tenta ser inserida. Supondo que a solução seja inserida no AE, verifica-se se o número máximo de soluções permitidas no AE foi ultrapassado

ou não com a inclusão da solução. Em caso afirmativo, o *crowding distance* (CD) de todas as soluções presentes no AE é calculado (incluindo a solução recentemente inserida). Em seguida, a solução com menor CD é eliminada do AE. A Figura 4.2 apresenta um exemplo de execução do mecanismo de poda. A Figura 4.2(a) apresenta o AE com quatro soluções, que nesse exemplo corresponde ao número máximo de soluções. A Figura 4.2(b) ilustra o processo em que uma nova solução é inserida no AE. A solução inserida está indicada pela seta. Ao se inserir essa solução, o CD de cada uma das soluções é calculado e a solução com menor CD é removida. Na Figura 4.2(b), a solução removida possui contornos tracejados. Ao se incluir outra solução, o mesmo procedimento é repetido. Isso é repetido até que todas as soluções do enxame tenham tentado ser inseridas no AE.



(a) Estado inicial do arquivo externo. (b) Estado do arquivo externo após um solução ser inserida. (c) Estado do arquivo externo após a inserção de uma outra solução.

Figura 4.2 Exemplo de execução do mecanismo de poda no SMPSO.

4.2.2 Otimização Multiobjetiva por Enxame de Partículas por Análise de Densidade e Seleção por Roleta

Também em 2009, Santana *et al.* [SPBF09] propuseram um algoritmo que apresentou resultados promissores em relação à quatro abordagens de enxames, a saber: CSS-MOPSO, MOPSO-CDLS, MOPSO e m-DNPSO. O algoritmo foi denominado de Otimização Multiobjetiva por Enxame de Partículas utilizando Análise de Densidade e Seleção por Roleta (**MOPSO-CDR**, *Multiple Objective Particle Swarm Optimization Approach using Crowding Distance and Roulette Wheel*). Basicamente, o diferencial desse algoritmo com respeito às abordagens desenvolvidas até em então foi o uso de um mecanismo de seleção por roleta para seleção dos líderes sociais das partículas do enxame. Além disso, essa abordagem apresenta uma nova forma da atualização do líder cognitivo de cada partícula em que o arquivo externo é utilizado. Cada uma das etapas de MOPSO-CDR é descrita nas próximas subseções.

4.2.2.1 Seleção dos Líderes Sociais

O MOPSO-CDR também utiliza um arquivo externo para armazenar soluções não-dominadas com relação as todas as soluções encontradas do início do processo de busca até o instante atual, e os líderes sociais são escolhidos desse arquivo. Diferentemente do SMPSO, que utiliza o torneio binário para selecionar uma solução do arquivo como líder para cada partícula, o

MOSPO-CDR utiliza a seleção por roleta. Nesse caso, a aptidão de cada solução do arquivo externo, que é utilizado na seleção por roleta, é o *crowding distance* da solução calculado com base em todas soluções do arquivo externo.

4.2.2.2 Seleção dos Líder Cognitivo

O mecanismo de seleção dos líderes cognitivos no MOPSO-CDR é um pouco mais complexo que no SMPSO. No MOPSO-CDR, se a posição (solução) atual da partícula domina o seu líder cognitivo, então a posição atual é definida como o novo líder cognitivo. Por outro lado, quando as soluções são incomparáveis, o arquivo externo é utilizado para decidir entre as duas soluções. Assim, primeiramente se localiza as soluções no arquivo externo que são mais próximas da posição atual da partícula e do líder cognitivo atual em termos de distância Euclidiana. Se a solução mais próxima da posição atual da partícula estiver em um região menos densa que a solução mais próxima do atual líder cognitivo então a posição atual da partícula é definida como o novo líder cognitivo da partícula. Caso contrário, o líder cognitivo permanece inalterado.

A Figura 4.3 apresenta um exemplo de execução do procedimento de atualização do líder cognitivo de uma partícula em particular. Essa figura ilustra apenas o caso em que o líder cognitivo e a posição atual da partícula são incomparáveis. Nesse figura, as soluções do arquivo externo estão envolvidas por uma curva tracejada. Como a partícula e seu atual líder cognitivo são soluções incomparáveis, o arquivo externo é usado para decidir entre as duas. As soluções do arquivo externo mais próximas da partícula e do líder cognitivo atual estão envolvidas por elipses. Nesse exemplo, a solução mais próxima da partícula está em uma região menos densa, isto é, ela possui um maior *crowding distance* que a solução mais próxima do líder cognitivo atual. Dessa forma, essa partícula (posição atual) se torna o novo líder cognitivo da partícula. Como se pode notar por esse exemplo, o objetivo desse mecanismo se torna claro: atrair as partículas para regiões menos densas do arquivo externo.

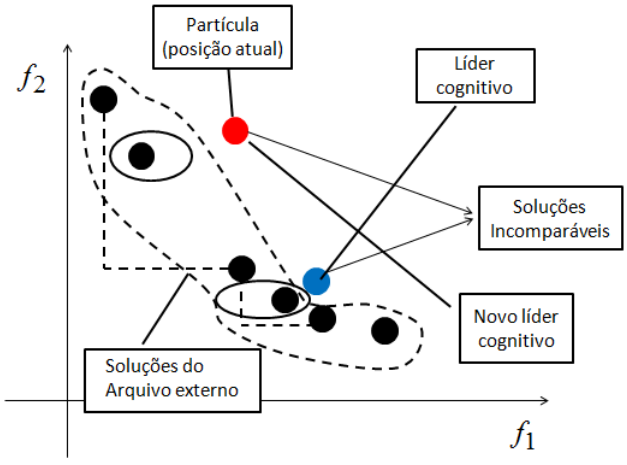


Figura 4.3 Seleção do líder cognitivo de um partícula para o caso em que o líder cognitivo e a posição atual da partícula são incomparáveis.

4.2.2.3 Atualização da Velocidade das Partículas

O MOPSO-CDR atualiza a posição das partículas conforme a equação (4.8) usando um peso de inércia adaptativo que diminui linearmente com o tempo como apresentado na equação (4.10). A velocidade máxima no MOPSO-CDR é dada pela equação (4.5) com $\delta = 1$. A equação (4.3) é usada para limitar a velocidade das partículas. Após a atualização da velocidade das partículas, suas posições são modificadas conforme equação (4.2). As partículas que ultrapassam os limites do espaço de busca são tratadas pela equação (4.6) e pela equação (4.7).

4.2.2.4 Operador de Turbulência

O operador de turbulência do MOPSO-CDR é o mesmo que o utilizado no MOPSO [CPL04]. O pseudo-código do operador de turbulência do MOPSO é apresentado no Algoritmo 9.

Algoritmo 9: Operador de turbulência.

/* **x:** partícula na qual o operador de mutação está sendo aplicado */
/* n é o número de variáveis de decisão */
/* t é a iteração atual */
/* t_{max} número total de iterações */
/* p_m é a taxa de mutação */
/* $\mathbf{x}_{max}$ é o vetor com os limites superiores do espaço de decisão. Cada componente do vetor é denotado por $x_{max,j}$ para $\forall j \in \{1,2,\ldots,n\}$ */
/* $\mathbf{x}_{min}$ é o vetor com os limites inferiores do espaço de decisão. Cada componente do vetor é denotado por $x_{min,j}$ para $\forall j \in \{1,2,\ldots,n\}$ */
se $U(0,1) < (1-t/t_{max})^{5/p_m}$ **então**
    /* Seleciona aleatoriamente uma dimensão do espaço de decisão em que a mutação será aplicada */
    $j = U[1,n]$;
    $intervalo\_mutacao = (x_{max,j} - x_{min,j})(1-t/t_{max})^{5/p_m}$;
    $ub = x_j + intervalo\_mutacao$;
    $lb = x_j - intervalo\_mutacao$;
    /* Controla os limites do intervalo de mutação para evitar que as partículas deixem o espaço de busca */
    se $lb < x_{min,j}$ **então**
        $lb = x_{min,j}$;
    se $ub > x_{max,j}$ **então**
        $ub = x_{max,j}$;
    /* Retira um número aleatório entre lb e ub de uma distribuição de probabilidade uniforme */
    $x_j \sim U(lb,ub)$.

O operador de turbulência é aplicado em cada iteração do algoritmo. No início do algoritmo, esse operador age sobre todas as partículas e em seguida o número de partículas afetadas por ele é rapidamente diminuído. A Figura 4.4 apresenta o gráfico da percentagem do número de partículas afetadas pelo operador de mutação pela percentagem do número de iterações. Como se pode observar por essa figura, o efeito do operador de mutação é inexistente antes mesmo do algoritmo atingir metade do número de iterações.

O operador de mutação basicamente realiza uma pertubação em somente uma dimensão em cada partícula. A dimensão da variável de decisão que será afetada pela mutação é escolhida aleatoriamente ($U[1,n]$). Como se pode observar pelo Algoritmo 9, o intervalo de pertubação também diminui com o passar das iterações. Desse modo, no início do algoritmo, o operador de turbulência permite que as partículas atinjam regiões inexploradas do espaço de busca e, à medida que as iterações passam, as alterações provocadas são cada vez menores até que, no fim, elas são inexistentes.

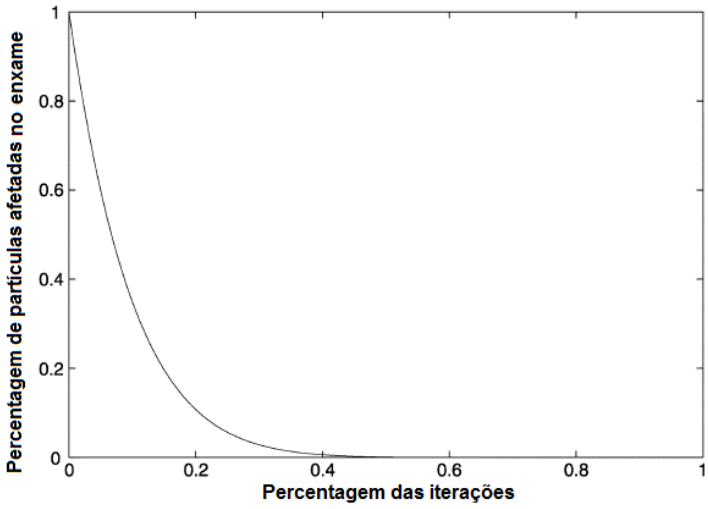
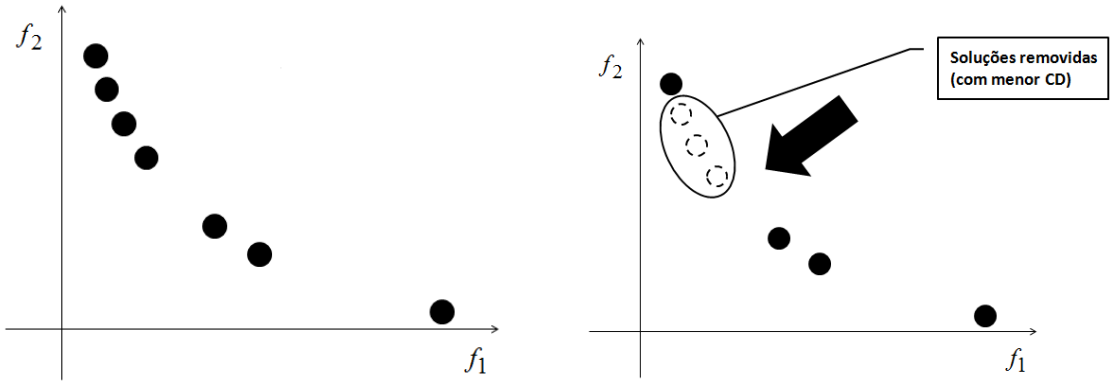


Figura 4.4 Percentagem de partículas nas quais o operador de turbulência é aplicado ao longo das iterações (adaptado de [CPL04]).

4.2.2.5 Atualização do Arquivo Externo

No MOPSO-CDR, todas as soluções não-dominadas do enxame e que são não-dominadas pelas soluções do arquivo externo são inseridas nele. Em seguida, as soluções no arquivo externo dominadas por essas soluções inseridas são removidas do arquivo externo. Todas essas soluções são incluídas simultaneamente. O arquivo externo no MOPSO-CDR tem um número máximo de soluções. Se esse número é excedido, o mecanismo de poda é utilizado. O mecanismo de poda funciona do seguinte modo. Inicialmente, o *crowding distance* de todas as soluções é calculado. Em seguida, as soluções do arquivo externo são classificadas de acordo com o *crowding distance*. Por fim, se o tamanho atual do arquivo é $|AE|$ e o número máximo de soluções permitido no arquivo externo é AE_{max}, as $|AE| - AE_{max}$ soluções do arquivo externo

com menor *crowding distance* são eliminadas. A Figura 4.5 apresenta um exemplo de execução do mecanismo de poda. Nesse exemplo o número máximo de partículas no arquivo externo é quatro. A Figura 4.5(a) apresenta um aquivo externo com sete soluções, três a mais que o permitido. O *crowding distance* das soluções é então calculado, e as soluções em regiões densas são removidas. A Figura 4.5(b) ilustra esse processo de eliminação no qual três soluções são removidas simultaneamente. Como se pode observar pela Figura 4.5, essa abordagem de poda no MOPSO-CDR pode fazer com que o arquivo externo tenha regiões muito esparsas. Isso se deve ao fato de que o *crowding distance* pode falhar na estimativa das densidades das soluções como observado em [PSG10]. Mais ainda, o mecanismo de poda do SMPSO aparenta ser superior ao do MOPSO-CDR. Ou seja, a abordagem do SMPSO de remoção de soluções do arquivo externo uma por uma parece ser mais efetiva que a remoção *em bloco* do MOPSO-CDR. Essa hipótese pode ser melhor investigada em um trabalho futuro.



(a) Estado inicial do arquivo externo.

(b) Estado do arquivo externo após um solução ser inserida.

Figura 4.5 Exemplo de execução do mecanismo de poda do MOPSO-CDR.

4.3 Enxame de Partículas Para Problemas com Muitos Objetivos

Assim como ocorre com os MOEAs, os MOPSOs baseados em dominância de Pareto também apresentam dificuldades quando tratam problemas com muitos objetivos pelas mesmas razões já mencionadas na Seção 3.1.1. Como já explicado, já existem MOEAs adaptados para lidarem com problemas com muitos objetivos; e dois MOEAs estado da arte para lidar com esses problemas foram apresentados. Apesar da adaptação bem sucedida de MOEAs para MaOPs, ainda não existem MOPSOs, até o momento, que possam lidar efetivamente com esses problemas. Além disso, MOPSOs têm algumas características tais como rápida convergência [CPL04], simplicidade, e um pequeno número de parâmetros de controle comparados a MOEAs [MS08] que o tornam bons candidatos para lidarem com problemas com muitos objetivos [KY07]. Na literatura, apenas recentemente os MOPSOs foram estendidos para problemas com muitos objetivos, de modo que as pesquisas nessa área ainda são incipientes. Algumas propostas de MOPSOs para problemas com muitos objetivos são apresentados nos próximos parágrafos.

Em 2007, Köppen e Yoshida [KY07] propuseram um MOPSO para lidar com problemas com muitos objetivos. Esse MOPSO utiliza uma relação de dominância difusa para atribuir qualidade a uma solução dentro de um conjunto de soluções A. Essa dominância foi denominada de *dominância gradual* [KY04]. O termo gradual se refere ao fato de que a dominância entre duas soluções é determinado por um grau de dominância entre $[0,1]$. O grau de dominância entre dois vetores $\mathbf{u}$ e $\mathbf{v}$ no espaço objetivo é dado pela equação (4.18):

$$\mu(\mathbf{u},\mathbf{v}) = \prod_{i=1}^{m} \left[\frac{u_i}{v_i}\right], \tag{4.18}$$

em que a seguinte noção de divisão foi utilizada:

$$\left[\frac{x}{y}\right] = \begin{cases} 1 & \text{se } y \leq x, \\ \frac{x}{y} & \text{se } x < y. \end{cases} \tag{4.19}$$

Após o cálculo dos graus de dominância, a aptidão $r(\mathbf{u})$ de uma solução $\mathbf{u}$ no espaço objetivo pertencente a um conjunto de soluções A é dado pela equação (4.20):

$$r(\mathbf{u}) = \max_{\mathbf{v} \in A - \{\mathbf{u}\}} \mu(\mathbf{u},\mathbf{v}). \tag{4.20}$$

O valor de aptidão de uma solução dominada é 1, caso contrário a aptidão está entre 0 e 1. Nessa abordagem, soluções com aptidões menores são preferidas pois elas são consideradas menos dominadas que outras soluções do conjunto. Nessa abordagem, o conjunto A representa o enxame de partículas [KY07, WL09].

Em 2008, Mostaghim e Schmeck [MS08] propuseram um método de atribuição de aptidão chamado de *DR* baseado em distâncias e usou esse método em um MOPSO que eles denominaram de DMOPSO. O DR de um vetor no espaço de decisão $\mathbf{x}_i$ denotado por $DR(\mathbf{x}_i)$ é dado por

$$DR(\mathbf{x}_i) = \sum_{k=1}^{m} \sum_{j \in AE} |f_k(\mathbf{x}_i) - f_k(\mathbf{x}_j)|. \tag{4.21}$$

Aqui, *AE* representa o arquivo externo; e quanto maior for valor de *DR* de uma solução melhor ela é. Eles também usaram o ranqueamento médio em um MOPSO denominando a combinação resultante de RMOPSO. Em [MS08], eles compararam o DMOPSO, RMOPSO e o NSGA II e mostraram que o DMOPSO se mostrou superior a essas técnicas para o problema de teste utilizado.

Em 2009, Wickramasinghe e Li [WL09] propuseram usar métricas de distância baseadas em preferência do usuário ao invés de dominância de Pareto para atribuir aptidões as soluções e assim encontrar eficientemente soluções em problemas com muitos objetivos. As métricas utilizadas foram o método de ponto de referência e a busca por feixe de luz (*light beam search*). Eles usaram esses métodos de atribuição de aptidão em um MOPSO chamado MDEPSO [WL08]. Os resultados mostraram que o MOPSO é capaz de convergir para regiões especificadas pelo usuário, mas a necessidade de se especificar pontos de referências, muitas vezes não disponíveis, torna essa abordagem ineficaz em problemas práticos nos quais pouco se sabe sobre a forma e localização da Frente de Pareto.

Em 2010, Kachroudi e Grossard [KG10] substituíram a dominância de Pareto pelo ranqueamento médio na seleção dos líderes no SMPSO e também no NSGA II. Essas duas modificações foram comparados sobre os problemas DTLZ{1,2,3,4} com até 80 objetivos. Os resultados experimentais mostraram que esta relação não foi capaz de melhorar o desempenho do SMPSO em termos de convergência e diversidade. Mais interessante ainda é o fato de que o SMPSO obteve um desempenho pior que o original SMPSO, enquanto o NSGA II modificado foi melhor que o original NSGA II. Esse exemplo mostra que somente a modificação de relação de dominância em um MOPSO é incapaz de melhorá-lo.

Por fim, em 2012, De Carvalho e Pozo [DP12] aplicaram o CDAS no SMPSO. Eles analisaram diferentes valores do parâmetro S_i e discutiram sua influência sobre o desempenho do SMPSO em termos de convergência e diversidade, de acordo com a contração e expansão da área de dominância das soluções. Entretanto como já foi exposto, o parâmetro S_i do CDAS é dependente do problema e é muito difícil de ser especificado.

4.4 Considerações Finais

Neste capítulo foi apresentado dois MOPSOs e também foi apresentado as principais adaptações de MOPSOs para problemas com muitos objetivos. Como se pôde perceber neste capítulo, a literatura envolvendo a aplicação de MOPSOs em problemas com muitos objetivos é ainda incipiente [DP12]. Além disso, as abordagens têm sérias desvantagens e carecem de serem melhoradas. Logo, a adaptação de MOPSOs para problemas com muitos objetivos é um problema em aberto e que apresenta vários desafios. Esse fato motivou o desenvolvimento dessa dissertação que consistiu em desenvolver uma técnicas baseada em enxame de partículas que fosse eficaz em problemas com muitos objetivos. A esse algoritmo deu-se o nome de MOPSO-GD. No próximo capítulo (Capítulo 5), o MOPSO-GD será apresentado em detalhes.

CAPÍTULO 5

MOPSO-GD

" *Dez mil dificuldades não constituem uma dúvida.* "
– Isaac Newton

NSTE capítulo, o algoritmo desenvolvido nessa dissertação chamado de MOPSO-GD é apresentado. O principal diferencial do MOPSO-GD com respeito ao SMPSO e ao MOPSO-CDR, por exemplo, é o uso de um método de atribuição de aptidão de alta granularidade denominado de Detrimento Global (GD). Esse é o mesmo método que é usado no CEGA. No entanto, o MOPSO-GD também apresenta um diferencial com relação ao CEGA e também com respeito ao MDFA. Esse diferencial consiste no uso de um arquivo externo para armazenar as soluções não-dominadas encontradas pelo algoritmo ao longo do processo de busca. Desse modo, o MOPSO-GD combina dominância de Pareto juntamente com um método de alta discriminação para conduzir a busca. O arquivo externo é utilizado também para assegurar o elitismo no MOPSO-GD. Na próxima seção, o algoritmo é explicado em detalhes.

5.1 Descrição do MOPSO-GD

O MOPSO-GD é baseado no SMPSO, apresentando diferenças quanto à forma como os líderes sociais e cognitivos são selecionados; e também na forma como o mecanismo de poda é conduzido. O SMPSO foi escolhido como base para o desenvolvimento do MOPSO-GD porque ele apresentou bons resultados em relação a outras abordagens quanto à convergência e à diversidade devido ao seu mecanismo de controle de velocidade. O pseudo-código do MOPSO-GD é apresentado em Algoritmo 10.

O MOPSO-GD começa inicializando as partículas no espaço de busca segundo uma distribuição uniforme. As partículas iniciam com velocidade igual a zero. Em seguida, as partículas são avaliadas usando o problema multiobjetivo. O MOPSO-GD assim como boa parte dos principais MOEAs e MOPSOs na literatura utiliza um arquivo externo AE para armazenar as soluções não-dominadas encontradas pelo algoritmo. Desse modo, após as partículas serem avaliadas, as partículas não-dominadas são inseridas no arquivo externo AE. O uso do arquivo externo é uma característica peculiar do MOPSO-GD com respeito ao CEGA e ao MDFA. De fato, como foi visto, o CEGA e o MDFA utilizam apenas uma única população para realizar o processo de busca enquanto o MOPSO-GD utiliza duas populações: uma interna (enxame) e outra externa (arquivo externo). Nesse sentido, o CEGA e o MDFA não similares ao NSGA II

Algoritmo 10: Pseudo-código do MOPSO-GD.

```
Inicialize o enxame S no espaço de busca;
Avalie as partículas usando o problema multiobjetivo;
Inicialize o arquivo externo AE com soluções não-dominadas;
para t de 1 a t_max faça
    Atualize líder cognitivo;
    Atualize líderes sociais usando seleção por torneio binário baseado no GD;
    Atualize velocidade das partículas;
    Atualize posição das partículas;
    Aplique o operador de mutação polinomial;
    Avalie as partículas usando o problema multiobjetivo;
    Atualize arquivo externo baseado no GD.
```

pois o elitismo é baseado em um esquema de seleção para sobrevivência do tipo $(\mu + \lambda)$; enquanto no MOPSO-GD o elitismo é assegurado por meio do arquivo externo assim como ocorre no SPEA2, MOPSO-CDR e SMPSO.

Após o estágio de inicialização, o MOPSO-GD entra em um ciclo até que um número máximo de iterações t_{max} seja atingido. Outros critérios de parada poderiam ser utilizados, mas, por questões de simplicidade e para facilitar a comparação com outros algoritmos, o critério de parada adotado foi o número máximo de iterações. Em cada iteração do MOPSO-GD, as seguintes operações são realizadas sucessivamente: primeiramente, os líderes cognitivos das partículas são determinados; em seguida, os líderes sociais são selecionados; depois as velocidades das partículas são calculadas e as suas posições são atualizadas; posteriormente, as partículas são perturbadas usando o operador de mutação polinomial; e, finalmente, as partículas do enxame são avaliadas pelo problema multiobjetivo e o arquivo externo é atualizado. Cada uma dessas etapas é detalhada nas próximas seções.

5.1.1 Seleção dos líderes cognitivos

No MOPSO-GD, a seleção dos líderes cognitivos é realizada comparando o líder cognitivo atual de cada partícula com a sua posição atual com respeito à dominância. Se a posição atual da partícula **domina** o seu líder cognitivo então esse líder cognitivo é substituído pela posição atual da partícula. Por outro lado, se a posição atual é dominada pelo líder cognitivo então o líder cognitivo da partícula é mantido. Por fim, se o líder cognitivo da partícula e a sua posição atual são incomparáveis então elas são comparadas entre si quanto a uma métrica denominada *ganho* [GPC09]. O ganho de uma solução $\mathbf{x}_i$ com relação a uma solução $\mathbf{x}_j$ é dado pela equação (5.1):

$$ganho(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{m} \max\left[f_k(\mathbf{x}_j) - f_k(\mathbf{x}_i), 0\right]. \tag{5.1}$$

Se o ganho da partícula na geração atual for maior que o ganho do líder cognitivo dessa partícula, então a posição dessa partícula se torna o novo líder cognitivo. Como se observa, o

mecanismo de atualização do líder cognitivo no MOPSO-GD é bastante simples e direto. É importante ressaltar aqui que o uso da métrica ganho como critério de seleção do líder cognitivo é exclusivo do MOPSO-GD. No SMPSO, por exemplo, quando a posição atual da partícula e o seu líder cognitivo atual são incomparáveis, utiliza-se a regra da solução mais recente para decidir entre as duas soluções. Ou seja, a posição atual da partícula é sempre preferida com relação ao líder cognitivo. Como é fácil notar, essa estratégia de seleção do líder cognitivo do SMPSO é uma potencial desvantagem desse algoritmo, principalmente em problemas com muitos objetivos. Como nesses problemas existe uma grande tendência das soluções serem incomparáveis, pode ocorrer do SMPSO dar preferência a soluções que estejam longe da Frente de Pareto. Esse fato motivou o uso do ganho no MOPSO-GD. O ganho é uma métrica que privilegia soluções que estejam, de certo modo, mais próximas da Frente de Pareto. A Figura 5.1 ilustra um exemplo do cálculo do ganho entre duas soluções A e B em um espaço objetivo bidimensional. Como se pode observar a partir dessa figura, o ganho de A é maior que o ganho de B. Essa figura ilustra também a motivação para o uso do ganho no processo de seleção dos líderes cognitivos no MOPSO-GD. A solução A e B são soluções incomparáveis e apesar disso a solução A (maior ganho) está mais próxima da Frente de Pareto do que a solução B.

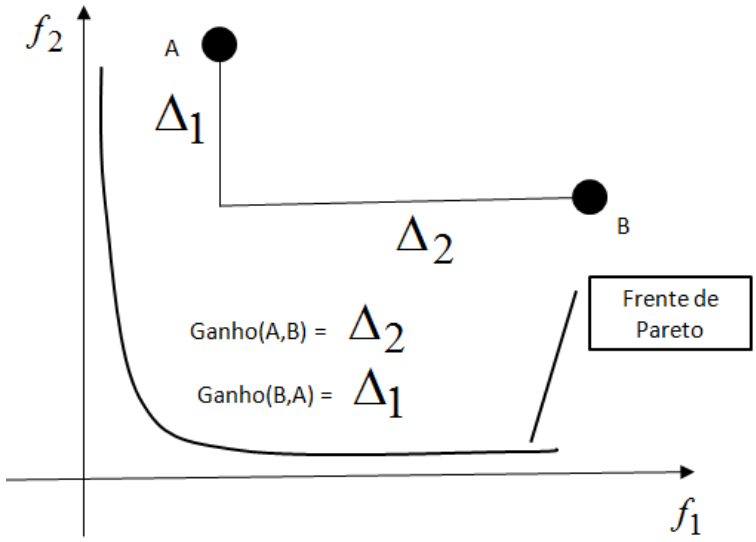


Figura 5.1 Ilustração do cálculo do ganho entre duas soluções A e B.

O pseudo-código do mecanismo de atualização dos líderes cognitivos do MOPSO-GD é apresentado em Algoritmo 11. Como se pode notar, o esquema de atualização dos líderes cognitivos no MOPSO-GD é similar ao do SMPSO e do MOPSO-CDR. A principal diferença ocorre com relação à escolha do novo líder cognitivo quando a posição atual de uma partícula é incomparável ao seu líder cognitivo. No SMPSO, essa escolha é baseada na regra da solução mais recente. No MOPSO-CDR, a escolha é baseada em uma análise de densidade de soluções no arquivo externo. Por outro lado, o MOPSO-GD utiliza informação sobre a distância da posição atual com relação ao líder cognitivo e vice-versa para determinar o novo líder cognitivo.

Algoritmo 11: Pseudo-código da seleção dos líderes cognitivos no MOPSO-GD.

```
1 para i ∈ S faça
2     se f(x_i) ≺ f(y_i) então
3         y_i = x_i;
4     senão
5         se f(x_i) || f(y_i) então
6             se ganho(x_i, y_i) > ganho(y_i, x_i) então
7                 y_i = x_i;
```

5.1.2 Seleção dos líderes sociais

A seleção do líder social no MOPSO-GD é também bastante simples sendo um dos principais deferenciais do MOPSO-GD com relação aos MOPSOs anteriormente mencionados (SMPSO e MOPSO-CDR). Em particular, esse método de seleção substitui o *crowding distance* pelo Detrimento Global do CEGA. Antes de comentar sobre esse método de seleção, é importante entender porque substituir o *crowding distance* pelo Detrimento Global. A razão disso é porque o *crowding distance* favorece soluções que estão muito afastadas de seus vizinhos imediatos. Essa característica é interessante quando aliada a um mecanismo que favoreça fortemente a convergência, como é o caso da dominância de Pareto em problemas multiobjetivos com dois ou três objetivos. Contudo, como já foi explicado, a dominância de Pareto perde a sua efetividade como mecanismo de convergência em problemas com muitos objetivos, pois todas as soluções se tornam praticamente não-dominadas; e encontrar uma solução que domine outra se torna cada vez mais raro. Dessa forma, apenas o critério secundário utilizado para seleção dos líderes é privilegiado. No caso de alguns algoritmos, esse critério é o *crowding distance*. Assim, soluções bastante afastadas das outras são escolhidas como líderes e preservadas pelo mecanismo de poda. Esse comportamento faz com que as soluções da população comecem a divergir uma das outras em problemas com muitos objetivos. Portanto, o *crowding distance* não é um bom critério para estabelecer a aptidão das soluções em problemas dessa natureza. Assim, outras técnicas devem ser utilizadas como critério secundário em substituição ao *crowding distance* (CD). Para uma melhor explicação sobre as desvantagens do CD em MaOPs o leitor pode consultar [PF03].

Desta forma, o Detrimento Global (GD) foi utilizado no MOPSO-GD com o objetivo de favorecer a convergência. O GD já foi exposto na Seção 3.3. Entretanto, esse método foi aplicado de um modo ligeiramente diferente que no CEGA. Em particular, o GD foi aplicado na população externa no MOPSO-GD, enquanto no CEGA, o GD é aplicado na população interna. Desse modo, o GD no MOPSO-GD assume a forma apresentada na equação (5.2):

$$GD(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in AE} \sum_{k=1}^{m} \max\left[f_k(\mathbf{x}_i) - f_k(\mathbf{x}_j), 0\right], \ \forall i \in AE. \tag{5.2}$$

É importante destacar que o GD é apenas atribuído às soluções presentes no arquivo externo. Além disso, as soluções envolvidas no cálculo pertencem apenas ao arquivo externo.

Essa decisão foi tomada porque as soluções presentes na população interna (enxame) são sempre incluídas (desde que não-dominadas) no arquivo externo. E, assim, elas são envolvidas indiretamente no cálculo do GD.

Uma vez que o GD no MOPSO-GD foi explicado, o processo de seleção dos líderes sociais pode ser descrito. A seleção dos líderes sociais no MOPSO-GD é similar ao SMPSO. Desse modo, para cada partícula do enxame, duas soluções aleatórias são escolhidas do arquivo externo e comparadas entre si com respeito ao GD. A solução com melhor GD (menor GD em um problema de minimização) é então escolhida como líder social da partícula. O pseudo-código em Algoritmo 12 apresenta o processo de seleção dos líderes sociais em detalhes. No pseudo-código, a função $random(1,|AE|)$ é uma função que escolhe um número inteiro aleatório entre $[1,|AE|]$, e $AE[b]$ representa a solução na posição b do arquivo externo AE.

Algoritmo 12: Pseudo-código da seleção dos líderes sociais no MOPSO-GD.

```
para i ∈ EA faça
    GD(x_i) = Σ_{x_j ∈ AE} Σ_{k=1}^{m} max(f_k(x_i) − f_k(x_j), 0);
para i ∈ S faça
    a = random(1, |AE|) ;
    b = random(1, |AE|) ;
    se GD(AE[a]) < GD(AE[b]) então
        ŷ_i = AE[a] ;
```

5.1.3 Atualização da posição das partículas

Após a atualização dos líderes cognitivos e sociais, as novas posições das partículas podem ser calculadas. O cálculo da velocidade das partículas no MOPSO-GD é similar ao do SMPSO. Apesar disso, as equações serão novamente expostas aqui por conveniência. Sendo assim, no MOPSO-GD as partículas têm a sua velocidade atualizada de acordo com a equação (5.3):

$$\begin{aligned} v_{ij}(t+1) = \chi \ \{ & \omega \cdot v_{ij}(t) + c_1 \cdot r_1(t) \cdot [y_{ij}(t) - x_{ij}(t)] \\ & + c_2 \cdot r_2(t) \cdot [\hat{y}_{ij}(t) - x_{ij}(t)] \}. \end{aligned} \tag{5.3}$$

Novamente, os coeficientes de aceleração c_1 e c_2 são amostrados de uma função de probabilidade com distribuição uniforme $c_1 \sim U(c_1^{min}, c_1^{max})$ e $c_2 \sim U(c_2^{min}, c_2^{max})$, respectivamente. Os termos c_1^{min} e c_1^{max} são os valores mínimos e máximos para o coeficiente c_1 enquanto c_2^{min} e c_2^{max} são os valores mínimos e máximos para o coeficiente c_2. O termo χ é o coeficiente de constrição e é calculado como:

$$\chi = \begin{cases} \frac{2}{2-\varphi-\sqrt{\varphi^2-4\varphi}}, & \text{se } \varphi > 4, \\ 1, & \text{se } \varphi \leq 4, \end{cases} \tag{5.4}$$

em que φ é dado pela equação (5.5):

$$\varphi = c_1 + c_2. \tag{5.5}$$

O MOPSO-GD também limita a velocidade das partículas de forma similar ao SMPSO. Assim, ele limita a velocidade v_{ij} de cada partícula i em cada dimensão $j \in \{1,2,\ldots n\}$ de acordo com a equação (5.6):

$$v_{ij}(t) = \begin{cases} \Delta_j, & \text{se } v_{ij}(t) > \Delta_j, \\ -\Delta_j, & \text{se } v_{ij}(t) \leq -\Delta_j, \\ v_{ij}(t), & \text{caso contrário,} \end{cases} \tag{5.6}$$

com

$$\Delta_j = \frac{(x_{max,j} - x_{min,j})}{2}, \quad \forall j \in \{1,2,\ldots,n\}, \tag{5.7}$$

em que $x_{min,j}$ é o limite inferior do espaço de busca na dimensão j e $x_{max,j}$ é o limite superior.

Uma vez que as velocidades das partículas foram determinadas, a suas posições são atualizadas simplesmente usando a equação (5.8):

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{5.8}$$

Ao se atualizar a posição das partículas, pode ocorrer delas escaparem dos limites do espaço de busca. Quando isso ocorre, ou seja, quando uma partícula i deixa o espaço de busca na dimensão j, a equação (5.9) é utilizada para recolocá-la novamente no espaço de busca nessa dimensão:

$$x_{ij}(t) = \begin{cases} x_{max,j} & \text{se } x'_{ij}(t) > x_{max,j}, \\ x_{min,j} & \text{se } x'_{ij}(t) < x_{min,j}, \end{cases} \tag{5.9}$$

em seguida a velocidade da partícula é invertida usando equação (5.10):

$$v_{ij}(t) = \begin{cases} -v'_{ij}(t) & \text{se } x'_{ij}(t) > x_{max,j} \text{ ou se } x'_{ij}(t) < x_{min,j}, \\ v'_{ij}(t) & \text{caso contrário,} \end{cases} \tag{5.10}$$

em que $x'_{ij}(t)$ e $v'_{ij}(t)$ são, respectivamente, a posição e a velocidade da partícula i antes de se aplicar a equação (5.9) e equação (5.10).

Após as partículas atualizarem suas posições, o operador de turbulência é aplicado. O operador de turbulência no MOPSO-GD é a mutação polinomial, que é aplicado sobre uma percentagem pré-definida α de partículas com uma probabilidade p_m. É importante enfatizar aqui que quando esse operador é aplicado, as partículas sempre se mantém no espaço de busca. Além disso, é importante ressaltar a diferença entre os termos α e p_m. A percentagem α se refere ao número de partículas do enxame que sofrerão a mutação. Por outro lado, o termo p_m é a probabilidade de cada uma das componentes de uma determinada partícula sofrer uma mutação. Em suma, α é uma probabilidade a nível de população, enquanto p_m é a probabilidade de aplicar a mutação em cada componente da partícula.

5.2 Atualização do arquivo externo

No MOPSO-GD, a poda do arquivo externo também é similar ao procedimento exposto para o SMPSO. A única diferença é a substituição do *crowding distance* pelo GD como critério de poda. O pseudo-código do processo de atualização do arquivo externo *AE* é apresentado em Algoritmo 13. Assim, as partículas do enxame *S* são inseridas uma de cada vez no arquivo externo. Se, ao inserir uma partícula no arquivo externo, o tamanho do arquivo externo ultrapassar o limite máximo AE_{max} permitido, então o *GD* de todas as soluções do arquivo externo é calculado e a solução com maior *GD* é eliminada do arquivo. Como se pode observar, o método de atualização do arquivo externo tende a preservar soluções próximas da Frente de Pareto. Por fim, é importante enfatizar que a estratégia de poda assume um papel fundamental no desempenho do MOPSO-GD. Como o MOPSO-GD encontra muitas soluções não-dominadas, o arquivo frequentemente tem o seu tamanho máximo ultrapassado. Sendo assim, a escolha de quais soluções devem ser preservadas no arquivo externo é fundamental.

Algoritmo 13: Pseudo-código do processo de poda do MOPSO-GD.

```
1 para s ∈ S faça
2     Insira partícula s no AE;
3     se |AE| > AE_max então
4         para i ∈ EA faça
5             GD(x_i) = Σ_{x_j ∈ AE} Σ_{k=1}^{m} max(f_k(x_i) − f_k(x_j), 0);
6         Remova solução com maior GD do arquivo externo AE;
```

5.3 Considerações Finais

Neste capítulo, foi apresentado o algoritmo desenvolvido nesta dissertação: o MOPSO-GD. O MOPSO-GD foi criado com o propósito de lidar com problemas com muitos objetivos. Para isso, ele adota um método de atribuição de aptidão granular denominado de GD. O GD foi utilizando tanto no processo de seleção dos líderes sociais das partículas como no processo de poda do arquivo externo. Assim, diferentemente dos métodos baseados em dominância de Pareto que não conseguem discriminar soluções, o MOPSO-GD por meio do GD tem um alto poder discriminativo e com isso ele consegue conduzir uma busca em direção à Frente de Pareto com maior efetividade. No próximo capítulo, o arranjo experimental para avaliar o MOPSO-GD é descrito.

CAPÍTULO 6

Experimentos

" *Chega sempre a hora em que não basta*
apenas protestar: após a filosofia,
a acção é indispensável. "
– Victor Hugo

NSTE capítulo, o arranjo experimental utilizado para validar o MOPSO-GD é apresentado. Os problemas de teste utilizados nos experimentos são apresentados Seção 6.1. Na Seção 6.2, as métricas utilizadas para avaliar os algoritmos são descritas. Por fim, na seção 6.3, o delineamento experimental é explicado.

6.1 Problemas de Teste

Para avaliação de algoritmos em problemas com muitos objetivos, Deb, Thiele, Laumanns e Zitzler em [DTLZ05], propuseram a família de problemas de teste conhecida pela sigla DTLZ, que correspondem as iniciais dos nomes dos autores. Além de serem escaláveis para qualquer número de objetivos, os problemas da família DTLZ apresentam as seguintes característica: (1) podem ser escalados para qualquer número de variáveis de decisão; (2) a dificuldade dos problemas podem ser controladas por meio do ajuste de um único parâmetro; (3) as Frentes de Pareto desses problemas são conhecidas e (4) a Frente de Pareto desses problemas pode ser descrita por meio de equações que permitem que a distância do conjunto de soluções encontrado por um algoritmo para a Frente de Pareto global possa ser determinada analiticamente.

Nos experimentos realizados nesse trabalho, foram utilizados quatro problemas de teste: os problemas DTlZ1, DTlZ3, DTlZ4, e DTLZ6. Esses problemas foram escolhidos, pois eles apresentam a Frente de Pareto contínua, e não apresentam restrições com relação às variáveis de decisão. Outra característica desses problemas é que o espaço de variáveis de decisão é contínuo. Os problemas DTLZ2 e DTLZ5 não foram inclusos nos experimentos, pois são versões simples dos problemas DTLZ{3,4} e DTLZ6, respectivamente. Ou seja, esses problemas não impõe grandes desafios aos algoritmos em termos de convergência e diversidade.

Nos próximos parágrafos, cada um dos problemas de teste será descrito em detalhes. Para todos os problemas, as seguinte notação é utilizada: o número de variáveis de decisão é representado por $n = m + k - 1$ em que m é o número de objetivos e k é um parâmetro que controla a dificuldade do problema, $\mathbf{x}$ é um vetor de variáveis de decisão, e $\mathbf{x}_m$ representa as k últimas variáveis de decisão do vetor $\mathbf{x}$. O termo x_i corresponde a i-ésima variável de decisão do vetor $\mathbf{x}$

e está sujeito a $0 \leq x_i \leq 1$ para $i = 1, 2, \ldots, n$. Por conveniência, os problemas DTLZ2 e DTLZ5 foram inseridos na descrição uma vez que os problemas DTLZ3, DTLZ4 e DTLZ6 são versões desses problemas.

6.1.1 Problema DTLZ1

O problema de teste DTLZ1 é um problema com a Frente de Pareto linear e é descrito matematicamente pela equação (6.1):

$$\begin{cases} \text{Minimize} f_1(\mathbf{x}) = \frac{1}{2} x_1 x_2 \ldots x_{m-1}(1 + g(\mathbf{x}_m)), \\ \text{Minimize} f_2(\mathbf{x}) = \frac{1}{2} x_1 x_2 \ldots (1 - x_{m-1})(1 + g(\mathbf{x}_m)), \\ \vdots \\ \text{Minimize} f_{m-1}(\mathbf{x}) = \frac{1}{2} x_1 (1 - x_2)(1 + g(\mathbf{x}_m)), \\ \text{Minimize} f_m(\mathbf{x}) = \frac{1}{2}(1 - x_1)(1 + g(\mathbf{x}_m)), \end{cases} \tag{6.1}$$

em que a função $g(\mathbf{x}_m)$ é definida pela equação (6.2):

$$g(\mathbf{x}_m) = 100 \left\{ |\mathbf{x}_m| + \sum_{x_i \in \mathbf{x}_m} (x_i - 0,5)^2 - \cos\left[20\pi(x_i - 0,5)\right] \right\}. \tag{6.2}$$

O termo $|\mathbf{x}_M|$ na função $g(\mathbf{x}_m)$ representa o tamanho do vetor $\mathbf{x}_m$, ou seja, o número de componentes que ele possui. As soluções do conjunto Pareto-ótimo correspondem a $x_i = 0,5 \quad \forall x_i \in \mathbf{x}_m$. A Frente de Pareto corresponde ao hiperplano $\sum_{j=1}^{m} f_j^* = 0,5$. A Figura 6.1 apresenta uma aproximação da Frente de Pareto global para o problema DTLZ1 com três objetivos $m = 3$.

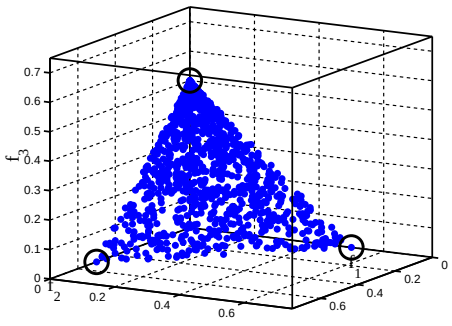


Figura 6.1 Aproximação da Frente de Pareto para o problema DTLZ1.

A principal dificuldade imposta pelo problema DTLZ1 é a dificuldade em termos de convergência para a Frente de Pareto. O problema DTLZ1 apresenta $11^k - 1$ Frentes de Pareto locais em que um algoritmo pode estagnar antes de atingir a Frente de Pareto global. Como se pode notar, o número de Frentes de Pareto locais depende do parâmetro k. Quanto maior for o valor do parâmetro k, mais Frentes de Pareto locais o problema DTLZ1 apresenta.

6.1.2 Problema DTLZ2

O problema DTLZ2 é um problema com Frente de Pareto esférica e é definido pela equação (6.3):

$$\begin{cases} \text{Minimize} f_1(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(x_1\pi/2)\cdots\cos(x_{m-2}\pi/2)\cos(x_{m-1}\pi/2), \\ \text{Minimize} f_2(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(x_1\pi/2)\cdots\cos(x_{m-2}\pi/2)\,\text{sen}(x_{m-1}\pi/2), \\ \text{Minimize} f_3(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(x_1\pi/2)\cdots\text{sen}(x_{m-2}\pi/2), \\ \qquad\vdots \\ \text{Minimize} f_m(\mathbf{x}) = (1+g(\mathbf{x}_m))\,\text{sen}(x_1\pi/2) \end{cases} \tag{6.3}$$

em que a função $g(\mathbf{x}_m)$ é definida pela equação (6.2):

$$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2. \tag{6.4}$$

O conjunto Pareto ótimo desse problema corresponde a $x_i = 0,5$ para todo $x_i \in \mathbf{x}_m$. A Frente de Pareto para esse problema é contínua, côncava, e ela é descrita pela equação $\sum_{j=1}^{m} f_j^{*2} = 1$, que corresponde a uma hiperesfera de raio 1 com centro na origem do espaço objetivo. A Figura 6.2 representa apenas uma aproximação da Frente de Pareto do problema DTLZ2. Mais precisamente, como se pode notar na Figura 6.2, a Frente de Pareto desse problema representa apenas o octante positivo da hiperesfera, ou seja, ao octante em que os pontos que formam a Frente de Pareto tem apenas valores positivos em suas coordenadas. O problema DTLZ2 não impõe nenhuma dificuldade em termos de convergência e nem de diversidade. Como se pode observar pela equação (6.4), existe apenas uma Frente de Pareto global que ocorre quando $g(\mathbf{x}) = 0$.

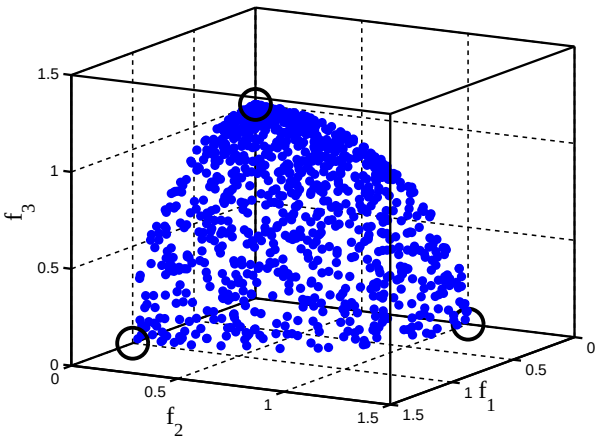


Figura 6.2 Aproximação da Frente de Pareto para o problema DTLZ2.

6.1.3 Problema DTLZ3

O problema DTLZ3 consiste em uma modificação do problema DTLZ2 descrito anteriormente. Essa modificação faz do problema DTLZ3 um dos problemas mais difíceis em termos de convergência. Esse problema introduz $3^k - 1$ Frentes de Pareto locais que são paralelos à Frente de Pareto global. O problema DTLZ3 é descrito matematicamente pela equação (6.5):

$$\begin{cases} \text{Minimize} f_1(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(x_1\pi/2)\cdots\cos(x_{m-2}\pi/2)\cos(x_{m-1}\pi/2), \\ \text{Minimize} f_2(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(x_1\pi/2)\cdots\cos(x_{m-2}\pi/2)\,\text{sen}(x_{m-1}\pi/2), \\ \text{Minimize} f_3(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(x_1\pi/2)\cdots\text{sen}(x_{m-2}\pi/2), \\ \qquad\vdots\vdots \\ \text{Minimize} f_m(\mathbf{x}) = (1+g(\mathbf{x}_m))\,\text{sen}(x_1\pi/2), \end{cases} \tag{6.5}$$

em que a função $g(\mathbf{x}_m)$ é definida pela equação (6.6):

$$g(\mathbf{x}_m) = 100\left\{|\mathbf{x}_m| + \sum_{x_i\in\mathbf{x}_m}(x_i-0,5)^2 - \cos[20\pi(x_i-0,5)]\right\}. \tag{6.6}$$

O conjunto Pareto ótimo corresponde a $x_i = 0,5$ para $x_i \in \mathbf{x}_m$.

6.1.4 Problema DTLZ4

O problema DTLZ4 também é uma modificação do problema DTLZ2 e assim como o problema DTLZ2 ele tem uma Frente de Pareto côncava (esférica). Esse problema foi criado para avaliar a capacidade dos algoritmos de gerar um conjunto bem distribuído de soluções sobre a Frente de Pareto global. Esse problema tem uma tendência natural em atrair as soluções para uma região específica da Frente de Pareto. Por exemplo, esse problema forma um conjunto denso de soluções próximo ao plano $f_1 - f_3$ em um problema com 3 objetivos [DTLZ05]. O problema DTLZ4 é descrito matematicamente pela equação (6.7). Como se observar por essa equação, o problema DTLZ4 consiste em criar um novo mapeamento diferente para as variáveis.

$$\begin{cases} \text{Minimize} f_1(x) = (1+g(\mathbf{x}_m))\cos(x_1^\alpha\pi/2)\cdots\cos(x_{m-2}^\alpha\pi/2)\cos(x_{m-1}^\alpha\pi/2), \\ \text{Minimize} f_2(x) = (1+g(\mathbf{x}_m))\cos(x_1^\alpha\pi/2)\cdots\cos(x_{m-2}^\alpha\pi/2)\,\text{sen}(x_{m-1}^\alpha\pi/2), \\ \text{Minimize} f_3(x) = (1+g(\mathbf{x}_m))\cos(x_1^\alpha\pi/2)\cdots\text{sen}(x_{m-2}^\alpha\pi/2), \\ \qquad\vdots\vdots \\ \text{Minimize} f_m(x) = (1+g(\mathbf{x}_m))\,\text{sen}(x_1^\alpha\pi/2), \end{cases} \tag{6.7}$$

em que a função $g(\mathbf{x}_m)$ é definida pela equação (6.8):

$$g(\mathbf{x}_m) = \sum_{x_i\in\mathbf{x}_m}(x_i-0.5)^2. \tag{6.8}$$

Para esse problema, Deb *et al.* [DTLZ05] sugeriu usar o valor $\alpha = 100$.

6.1.5 Problema DTLZ5

O problema DTLZ5 é descrito matematicamente pela equação (6.9) [SIR11]:

$$\begin{cases} \text{Minimize } f_1(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2})\cos(\theta_{m-1}), \\ \text{Minimize } f_2(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2}\,\text{sen}(\theta_{m-1}), \\ \text{Minimize } f_3(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\text{sen}(\theta_{m-2}), \\ \qquad \vdots\vdots \\ \text{Minimize } f_m(\mathbf{x}) = (1+g(\mathbf{x}_m))\,\text{sen}(\theta_1), \\ \text{em que} \quad \theta_i = \begin{cases} \dfrac{\pi}{2}x_i, \quad se\ i=1 \\ \dfrac{\pi}{4[1+g(\mathbf{x}_m)]}[1+2g(\mathbf{x}_m)x_i], \quad \forall i \in \{2,3,\ldots,(m-1)\}, \end{cases} \end{cases} \tag{6.9}$$

em que a função $g(\mathbf{x}_m)$ é definida pela equação (6.10):

$$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2. \tag{6.10}$$

O conjunto Pareto-ótimo desse problema corresponde aos pontos em que $x_i = 0,5$ para todo $x_i \in \mathbf{x}_m$ e os valores dos vetores objetivos satisfazem $\sum_{j=1}^{m} f_j^2 = 1$. Este problema avalia a habilidade de um MOEA de convergir para uma curva, ou seja, uma Frente de Pareto de duas dimensões imersa em um espaço objetivo de m dimensões. A Figura 6.3 ilustra a Frente de Pareto do problema DTLZ5 em três dimensões. O problema DTLZ5 tem apenas duas dimensões independentemente do número de objetivos pois devido a sua formulação matemática os objetivos de f_1 a f_{m-1} não são conflitantes entre si.

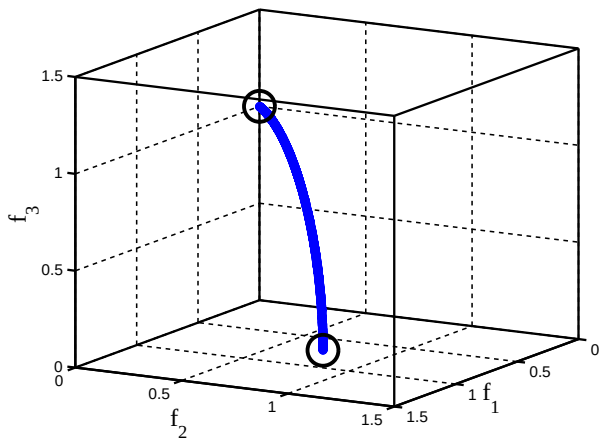


Figura 6.3 Aproximação da Frente de Pareto para o problema DTLZ5.

6.1.6 Problema DTLZ6

O problema DTLZ6 é similar ao problema DTLZ5. Entretanto, o problema DTLZ6 incorpora maiores dificuldades de convergência. Esse problema é definido matematicamente pela equação (6.11):

$$\begin{cases} \text{Minimize } f_1(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2})\cos(\theta_{m-1}), \\ \text{Minimize } f_2(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2}\,\text{sen}(\theta_{m-1}), \\ \text{Minimize } f_3(\mathbf{x}) = (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\text{sen}(\theta_{m-2}), \\ \qquad \vdots\!\vdots \\ \text{Minimize } f_m(\mathbf{x}) = (1+g(\mathbf{x}_m))\,\text{sen}(\theta_1), \\ \text{em que} \quad \theta_i = \begin{cases} \dfrac{\pi}{2}x_i, \quad se\ i=1 \\ \dfrac{\pi}{4(1+g(\mathbf{x}_M))}(1+2g(\mathbf{x}_M)x_i), \quad \forall i \in \{2,3,\ldots,(m-1)\}, \end{cases} \end{cases} \tag{6.11}$$

em que a função $g(\mathbf{x}_m)$ é definida pela equação (6.12):

$$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} x_i^{0,1} \tag{6.12}$$

Diferentemente dos problemas descritos anteriormente, as soluções do conjunto Pareto-ótimo correspondem aos pontos em que $x_i = 0$, para todo $x_i \in \mathbf{x}_m$, e os valores dos vetores objetivos satisfazem $\sum_{j=1}^{m} f_j^2 = 1$.

6.2 Métricas de Avaliação

Nesta seção, as duas métricas utilizadas para a realização dos experimentos são apresentadas. Uma para avaliar a convergência dos algoritmos para a Frente de Pareto chamada *Convergência*. E a outra métrica conhecida como Distância Geracional Invertida para avaliar a diversidade das soluções geradas pelo algoritmo.

6.2.1 Métrica de Convergência

A métrica Convergência [GPC10] consiste em calcular a distância média entre as soluções do conjunto aproximação PF (aproximação da Frente de Pareto) obtido pelo algoritmo e o ponto mais próximo de cada uma delas na Frente de Pareto. Inicialmente se calcula para cada ponto $\mathbf{u} \in PF$ a distância Euclidiana normalizada para o ponto mais próximo em $\mathscr{P}\mathscr{F}^*$ conforme a equação (6.13):

$$d(\mathbf{u}) = \min_{\mathbf{v} \in \mathscr{P}\mathscr{F}^*} \sqrt{\sum_{j=1}^{m} \left(\frac{u_j - v_j}{f_j^{max} - f_j^{min}} \right)^2}, \tag{6.13}$$

em que f_j^{max} e f_j^{min} denotam respectivamente o valor máximo e mínimo para o j-ésimo objetivo da Frente de Pareto. Finalmente, a métrica de convergência C é calculada por meio das distâncias normalizadas dos diferentes $\mathbf{u} \in PF$ usando a equação (6.14):

$$C(PF) = \frac{\sum_{\mathbf{u} \in PF} d(\mathbf{u})}{|PF|}. \tag{6.14}$$

Como para os problemas utilizados nos experimentos (problemas DTLZ{1,3,4,6}) as equações que descrevem a Frente de Pareto são conhecidas, o valor $d(\mathbf{u})$ para cada $\mathbf{u} \in PF$ foi determinado analiticamente. Por fim, quanto menor for o valor de $C(PF)$, melhor é o desempenho do algoritmo avaliado em termos de convergência.

6.2.2 Distância Geracional Invertida

A Distância Geracional Invertida (**IGD**, *Inverted Generational Distance*) [GPC10] é uma variação de uma métrica conhecida como distancia geracional [CLV07]. Para se calcular o IGD é necessário um conjunto de referência de pontos pertencentes à Frente de Pareto global, esse conjunto é denotado por Z. Com isso, o IGD é definido pela equação (6.15):

$$IGD(PF) = \frac{\sqrt{\sum_{\mathbf{v} \in Z} d(\mathbf{v})^2}}{|Z|}, \tag{6.15}$$

em que $d(\mathbf{v})$ é a distância Euclidiana entre um vetor objetivo $\mathbf{v} \in Z$ e o vetor objetivo mais próximo em PF, ou seja

$$d(\mathbf{v}) = \min_{\mathbf{u} \in PF} \sqrt{\sum_{j=1}^{m} (u_j - v_j)^2}. \tag{6.16}$$

Nesta dissertação, o conjunto Z é formado pelas soluções extremas da Frente de Pareto e pelo ponto conhecido como joelho [GPC10, Gar09]. As soluções extremas foram escolhidas pois elas representam as soluções de máxima diversidade [SIR11]. O joelho foi inserido pois ele é geralmente a solução escolhida pelo tomador de decisão [LJC09].

Para o problema DTLZ1 os extremos da Frente de Pareto são dados por m vetores da forma $\mathbf{e}_i = [f_1^i, f_2^i \ldots f_j^i \ldots f_m^i]$, em que $f_j^i = 0,5$, se $j = i$, e $f_j^i = 0$, caso contrário, para $\forall i \in 1,2,\ldots,m$. O joelho da Frente de Pareto desse problema corresponde ao baricentro do triângulo conectando os pontos extremos. Assim, o joelho corresponde ao ponto $\mathbf{j} = [v_1, v_2, \ldots, v_m]$ em que $v_i = 0,5/m$, para $\forall i \in 1,2,\ldots,m$.

Para os problemas DTLZ3, DTLZ4 e DTLZ6, o conjunto de referência Z foi formado do mesmo modo como Garza-Fabre *et al.* [GPC10] Sendo assim, o conjunto Z é formado por m vetores da forma $\mathbf{e}_i = [f_1^i, f_2^i \ldots f_j^i \ldots f_m^i]$, em que $f_j^i = 1$, se $j = i$, e $f_j^i = 0$, caso contrário, para $\forall i \in 1,2,\ldots,m$. Para esses problemas, o joelho corresponde ao ponto $\mathbf{j} = [v_1, v_2, \ldots, v_m]$, em que $v_i = \frac{1}{\sqrt{m}}$ para $\forall i \in 1,2,\ldots,m$.

Por fim, é importante mencionar que o IGD como formulado acima fornece informação sobre o espalhamento do PF dos algoritmos sobre a Frente de Pareto global. Desse modo, ele foi utilizado com uma métrica de diversidade nos experimentos. Contudo, deve-se salientar que o IGD também é uma métrica de convergência, de modo que sua análise deve levar em conta

essa duplicidade. Para terminar, quanto menor for o IGD do PF melhor é a sua convergência e diversidade.

6.3 Arranjo Experimental

Nessa dissertação, foram utilizados quatro algoritmos para comparação com o MOPSO-GD. Foram escolhidos dois MOPSOs (SMPSO e MOPSO-CDR) e dois MOEAs (CEGA e MDFA). O SMPSO e MOPSO-CDR foram escolhidos para demonstrar a ineficácia dos MOPSOs baseado em dominância de Pareto em problemas com muitos objetivos. O CEGA e o MDFA foram escolhidos pois eles são dois algoritmos estado da arte para lidar com problemas com muitos objetivos. Esses algoritmos foram implementados usando a plataforma JMetal [DN11]. O JMetal é uma plataforma de desenvolvimento em linguagem Java criada para facilitar e agilizar o desenvolvimento de MOEAs e MOPSOs. Os detalhes dos experimentos são apresentados nas próximas seções.

6.3.1 Definição dos Parâmetros dos Algoritmos

Em todos os algoritmos, foram utilizados 100 indivíduos/partículas ($N = 100$) e um total de 300 iterações ($t_{max} = 300$). Para o CEGA e o MDFA, o operador de cruzamento SBX foi aplicado com uma probabilidade $p_c = 1$ e com o parâmetro $\eta_c = 15$. A mutação polinomial foi aplicada com um parâmetro $\eta_m = 20$ e com uma probabilidade $p_m = \frac{1}{n}$, em que n é número de variáveis de decisão. Para o CEGA, no algoritmo de agrupamento foram formados todas as vezes 50 grupos ($N_{grupos} = 50$). No MDFA o número de pesos utilizado foi $|V| = 100$. Para o MOPSO-CDR, o peso de inércia foi $\omega(0) = 0,4$ e o peso de inércia final foi $\omega(t_{max}) = 0$. Os coeficientes de aceleração foram $c_1 = 1.49445$ e $c_2 = 1.49445$. O tamanho máximo do arquivo foi $AE_{max} = 100$. A taxa de mutação utilizada no operador de turbulência foi 0,5. Por fim, no SMPSO, o peso de inércia do SMPSO foi $\omega = 0,1$, e os valores mínimos e máximos dos coeficientes de aceleração foram $c_1^{min} = 1,5$ $c_1^{max} = 2,5$ $c_2^{min} = 1,5$ e $c_2^{max} = 2,5$. Além disso, o tamanho máximo do arquivo foi $AE_{max} = 100$, e a mutação polinomial foi aplicado em $\alpha = 15\%$ das partículas com probablidade $p_m = \frac{1}{n}$. Esses parâmetros foram escolhidos utilizando as definições originais dos algoritmos. No caso específico do MOPSO-GD, a escolha dos parâmetros foi a mesma que a do SMPSO. A Tabela 6.1 apresenta uma listagem dos principais parâmetros utilizados nos algortimos. O número de indivíduos/partículas e o número máximo de iterações não foram inseridos na tabela pois são comuns a todos os algoritmos. O termo AE_{max}, que foi definido com o mesmo tamanho do enxame, também não foi inserido na tabela. Como se pode observar na Tabela 6.1, o número de parâmetros nos MOPSOs é maior que nos MOEAs. Contudo, o MOPSO-GD, em uma certa medida, pode ser considerado mais flexível que o CEGA e o MDFA, uma vez que ele não apresenta parâmetos que afetam muito o seu desempenho e que, portanto, são difíceis de serem estabelecidos, como o número de grupos N_{grupos} no caso do CEGA e o número de pesos $|V|$ no caso do MDFA. Os parâmetros c_1 e c_2, por exemplo, não precisam ser estabelecidos com precisão pois eles são selecionados aleatoriamente, e, além disso, foi observado que o MOPSO-GD é pouco sensível ao parâmetro ω. Nesse sentido, o MOPSO-GD apresenta dois parâmetros que podem ser considerados cru-

ciais, a saber, η_m e α. Dessa menaira, o MOPSO-GD pode ser considerado, dependendo do problema, mais flexível que o CEGA e o MDFA.

Tabela 6.1 Lista de Parâmetros dos Algoritmos.

Principais Parâmetros				
CEGA	MDFA	SMPSO	MOPSO-CDR	MOPSO-GD
p_c	p_c	c_1^{max}, c_2^{max}	c_1, c_2	c_1^{max}, c_2^{max}
η_m	η_m	c_1^{min}, c_2^{min}	$\omega(t_{max})$	c_1^{min}, c_2^{min}
η_c	η_c	ω	$\omega(0)$	ω
N_{grupos}	$\|V\|$	α	p_m	α
–	–	η_m	–	η_m

6.3.2 Delineamento Experimental

Nos experimentos realizados, cada algoritmo foi executado 31 vezes para cada problema de teste. Escolheu-se o valor 31 porque a estatística utilizada para a comparação dos algoritmos foi a mediana, que também foi utilizada por Garza-Fabre *et al.* em [Gar09, GPC10]. Para o problema DTLZ1, escolheu-se um valor de $k = 5$ e para os outros problemas foi escolhido $k = 10$, conforme recomendado por [GPC10, CLV07, DTLZ05]. Em cada execução, a *Convergência* e o IGD foram calculados. Seguindo o método experimental de Garza Fabre *et al.* [GPC10, Gar09], os resultados foram apresentados em *boxplots*. A vantagem de usar *boxplots* é que os resultados dos algoritmos são mais fáceis de serem visualizados do que se fossem apresentados em tabelas. Esse modo de apresentação torna a interpretação dos resultados muito mais rápida e direta.

6.3.3 Normalização dos objetivos

Como em problemas multiobjetivos os objetivos podem estar em escalas diferentes, é necessário realizar a normalização desses objetivos antes de realizar algum cálculo envolvendo eles. Nesta dissertação, a técnica de normalização proposta por Garza Fabre *et al.* [GPC10] foi usada. Essa técnica foi usada em todos os algoritmos (MOPSOs e MOEAs) usados nesta dissertação. De acordo com [GPC10], o valor normalizado $f'_m(\mathbf{x}_i)$ de uma solução $\mathbf{x}_i$ para o objetivo $m \in \{1,2,3\ldots,m\}$ é dado pela equação (6.17):

$$f'_m(\mathbf{x}_i) = \frac{f_m(\mathbf{x}_i) - f_m^{min}(t)}{f_m^{max}(t) - f_m^{min}(t)}, \tag{6.17}$$

em que $f_m^{max}(t)$ e $f_m^{min}(t)$ são os valores máximo e mínimo conhecidos para o m-ésimo objetivo até a iteração t, respectivamente. Ou seja, os valores máximo e mínimo conhecidos para o objetivo m desde o início do processo de busca até a iteração atual t. Como se sabe a priori que para os problemas de teste utilizados $f_m^{min}(t) = 0, \quad \forall t \in \{1,2,\ldots,t_{max}\}$, a normalização foi realizada como segue: $f'_m(\mathbf{x}_i) = f_m(\mathbf{x}_i)/f_m^{max}(t)$.

CAPÍTULO 7

Resultados

" *A vida só se compreende mediante*
um retorno ao passado, mas
só se vive para diante. "
– Soren Kierkegaard

NESTE capítulo, os resultados dos experimentos são apresentados. Na Seção 7.1 são apresentados os resultados das métricas *Convergência* e IGD para o SMPSO, MOPSO-CDR, CEGA, MDFA e MOPSO-GD segundo o arranjo experimental apresentado no capítulo anterior. Na Seção 7.2, as Frentes de Pareto obtidas pelos algoritmos são analisadas e discutidas em detalhes.

7.1 Análise da Convergência e do IGD dos Algoritmos

Esta seção apresenta os resultados de convergência e IGD dos algoritmos para os problemas DTLZ{1,3,4,6}. Todos os resultados de convergência e IGD são apresentados em *boxplots* em escala logarítmica. A discussão é feita por problema, e o desempenho dos algoritmos é analisado para todas as instâncias do problema.

7.1.1 Análise para o Problema DTLZ1

As Figuras 7.1 e 7.2 apresentam os resultados para a convergência e o IGD, respectivamente, de todos os algoritmos em todas as instâncias do problema DTLZ1. Para esse problema, o comportamento de convergência do MOPSO-GD é claramente melhor que o do MOPSO-CDR e do SMPSO. Com relação ao MDFA, o MOPSO-GD alcançou melhores resultados ou obteve resultados similares à ele. No entanto, o CEGA mostrou uma convergência similar ao MOPSO-GD até 10 objetivos; a partir desse número (de 15 em diante), a convergência do CEGA é superior a convergência do MOPSO-GD e do MDFA. É importante salientar que a convergência do CEGA se torna cada vez melhor com o aumento do número de objetivos. O mesmo não ocorre com os outros algoritmos, incluindo o MOPSO-GD. Ou seja, com exceção do CEGA, todos os algoritmos obtiveram um desempenho aproximadamente similar em todas as instâncias do problema.

Analisando-se o tamanho dos *boxplots* do CEGA, observa-se que sua convergência é menos estável à medida que o número de objetivos aumenta. O MOPSO-GD, por outro lado, apresenta uma maior estabilidade quanto à sua convergência quando comparado ao CEGA e também ao

MDFA.

Com relação aos algoritmos MOPSO-CDR e SMPSO, observa-se que eles estão muito distantes da Frente de Pareto em todas as instâncias do problema. Além disso, observa-se que esse comportamento é estável uma vez que eles apresentam *boxplots* muito estreitos. Esse resultado jé era esperado devido ao fato desses algoritmos serem baseados em dominância de Pareto e utilizarem o *crowding distance* como método de estimação de densidade. A Seção 7.1.5 apresenta uma melhor explicação sobre isso. O SMPSO ainda conseguiu obter um resultado ligeiramente melhor que o MOPSO-CDR para a instância do problema com 5 objetivos. No entanto, observa-se que as diferenças de desempenho do SMPSO com relação ao MOPSO-CDR se tornam cada vez menores à medida que o número de objetivos aumenta.

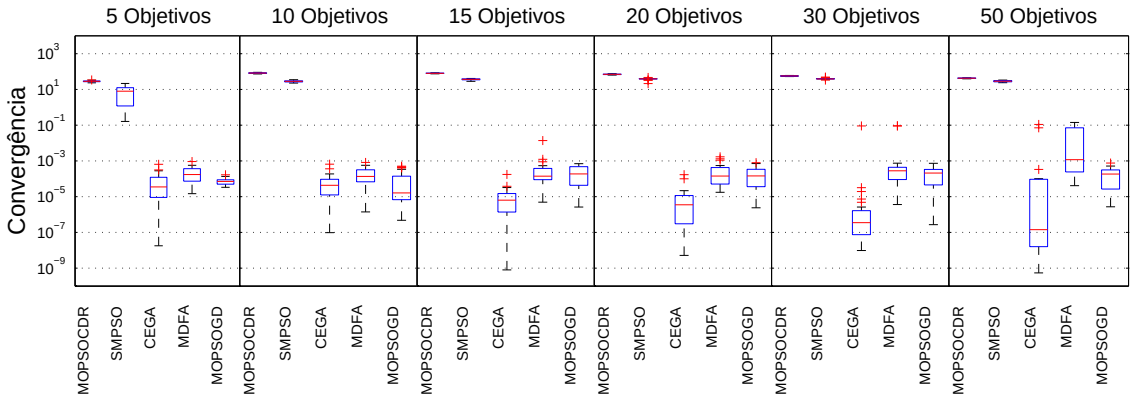


Figura 7.1 Métrica convergência para o problema DTLZ1.

Com respeito ao IGD, o MOPSO-GD apresentou um comportamento similar ao do CEGA e ao do MDFA. e obteve um desempenho superior ao SMPSO e ao MOPSO-CDR em todas as instâncias do problema de um modo geral. Além disso, observa-se, analisando a variância dos *boxplots*, que os resultados de IGD para o CEGA, MDFA e MOPSO-GD são estáveis para todas as instâncias do problema. Ou seja, os valores de IGD para esses algoritmos são praticamente o mesmos ao longo de todas as execuções dos algoritmos. No entanto, o SMPSO apresentou uma grande variância em seu IGD, principalmente para mais de 5 objetivos.

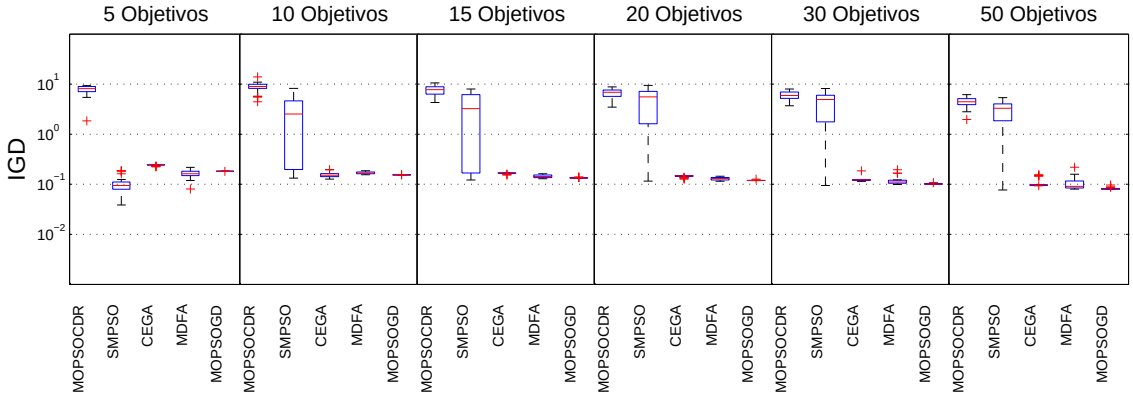


Figura 7.2 Métrica IGD para o problema DTLZ1.

7.1.2 Análise para o Problema DTLZ3

As Figuras 7.3 e 7.4 apresentam os resultados para o problema DTLZ3. Os valores da métrica de convergência para o MOPSO-GD são muito baixos e se aproximam de zero quando o número de objetivos aumenta como é mostrado na Figura 7.3. Analisando a Figura 7.3, observa-se que, em todas as instâncias, a convergência do MOPSO-GD é melhor que a dos outros algoritmos. Com exceção da instância de 5 objetivos, apenas alguns *outliers* aparecem ao invés da caixa típica dos *boxplots*, indicando que em algumas execuções o MOPSO-GD não convergiu precisamente para a Frente de Pareto. Para a instância com 50 objetivos, não existem nem *outliers*. Nessa instância, o valor da convergência do MOPSO-GD foi zero em todas as execuções do algoritmo e por isso seu *boxplot* não aparece na Figura 7.3.

Os algoritmos MOPSO-CDR e SMPSO obtiveram novamente os piores valores de convergência em todas as instâncias do problema. Nesse problema, a convergência alcançou valores na ordem de 10^2 para esses dois algoritmos. Isso ocorreu porque esse problema é bastante difícil em termos de convergência devido à suas múltiplas Frentes de Pareto locais. O CEGA e o MDFA alcançaram valores similares de convergência em toda as instâncias dos problemas. É importante assinalar que à medida que número de objetivos aumenta, os valores de convergência do CEGA e MDFA se tornam cada vez piores, o que não ocorre no MOPSO-GD.

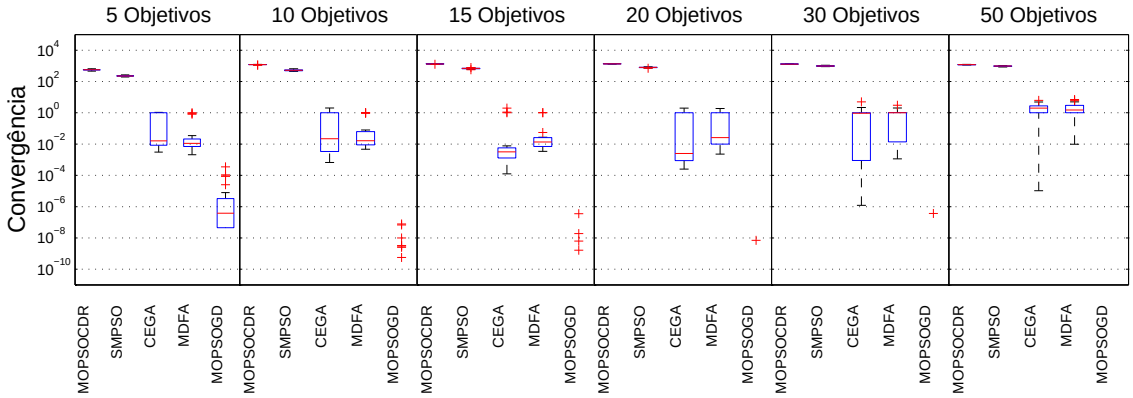


Figura 7.3 Métrica convergência para o problema DTLZ3.

A Figura 7.4 apresenta os valores de IGD dos algoritmos para todas as instâncias do problema. De acordo com essa figura, os valores de IGD do MOPSO-CDR e SMPSO foram novamente os piores entre todos os algoritmos. O IGD do CEGA, MDFA e MOPSO-GD foram similares em todas as instâncias. Contudo, observa-se que o IGD do MOPSO-GD é ligeiramente menor que o do CEGA e do MDFA, principalmente para instâncias com um grande número de objetivos (com mais de 20 objetivos). Em suma, o MOPSO-GD apresenta um melhor desempenho que os outros algoritmos em termos de convergência em várias ordens de magnitude e também mostra a menor métrica de IGD que todos os algoritmos (até mesmo melhor que o CEGA e MDFA).

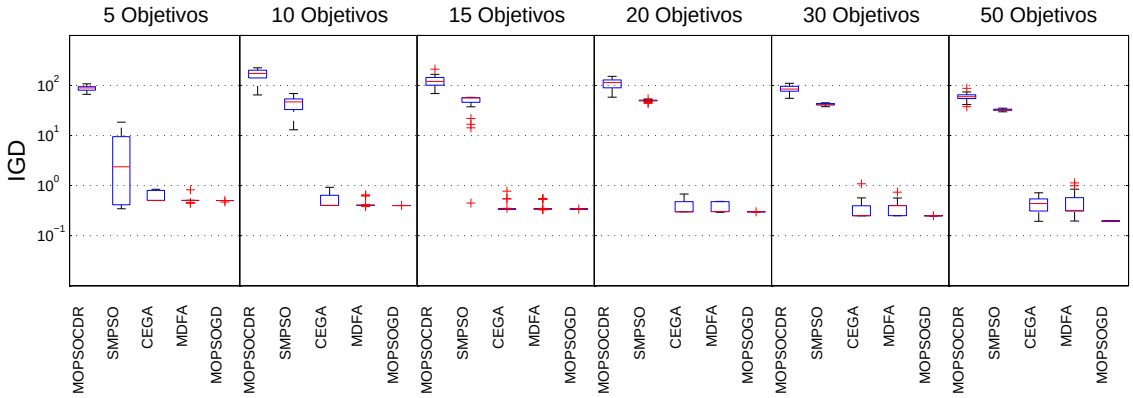


Figura 7.4 Métrica IGD para o problema DTLZ3.

7.1.3 Análise para o Problema DTLZ4

As Figuras 7.5 e 7.6 apresentam os resultados para o problema DTLZ4. Para a instância do problema com 5 objetivos, a convergência do CEGA e do MDFA é ligeiramente melhor que o do MOPSO-GD. Contudo, com o crescimento do número de objetivos, a convergência do MOPSO-GD se torna cada vez melhor. A partir de 10 objetivos, muitos valores de convergência são zero e por isso, com exceção de alguns *outliers*, seus valores não aparecem nas figura. Para acima de 20 objetivos, os *boxplots* do MOPSO-GD nem aparecem na Figura 7.5. Em termos de IGD, todos os algoritmos apresentam resultados muito similares. Entretanto, vale ressaltar que para a instância de 5 objetivos, o SMPSO e o MOPSO-CDR apresentaram valores mais baixos de IGD que o CEGA, MDFA e MOPSO-GD.

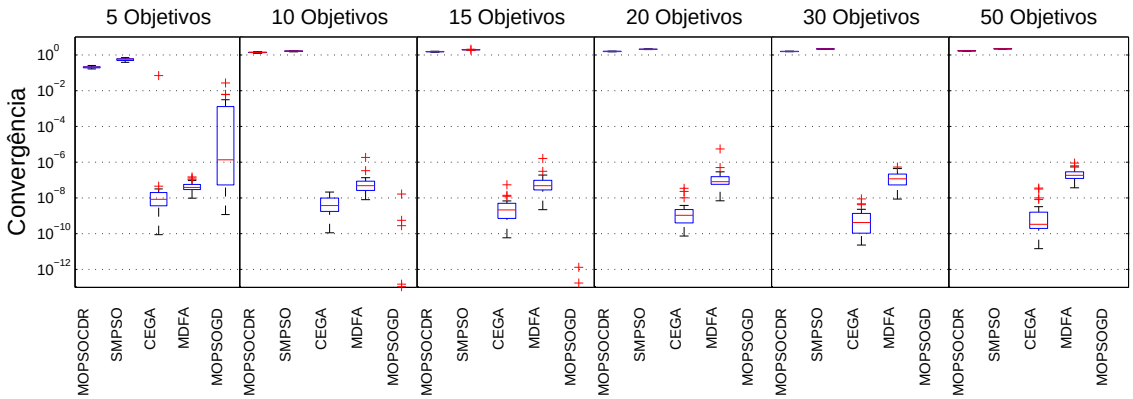


Figura 7.5 Métrica convergência para o problema DTLZ4.

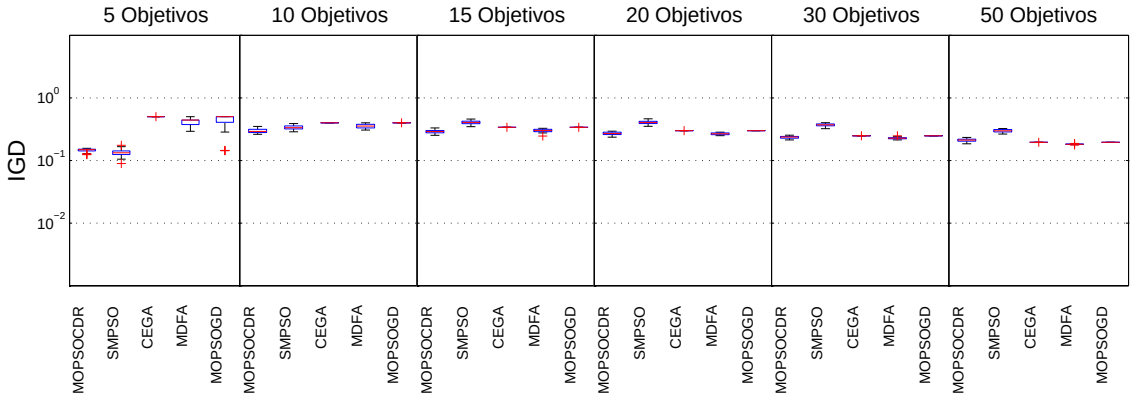


Figura 7.6 Métrica IGD para o problema DTLZ4.

7.1.4 Análise para o Problema DTLZ6

As Figuras 7.7 e 7.8 apresentam os resultados para o problema DTLZ6. O valor da métrica de convergência para o MOPSO-GD é muito menor que a de todos os outros algoritmos em todas as instâncias do problema. Entretanto, é interessante observar que a convergência do MOPSO-GD varia sobre várias ordens de magnitude como indicado pelo tamanho das caixas correspondentes e *outliers*. Uma razão para isso pode ser o fato de que o problema DTLZ6 é um problema degenerado com muitas Frentes de Pareto locais. Em termos de IGD, de um modo geral, os resultados são similares para todos os algoritmos com exceção do MOPSO-CDR.

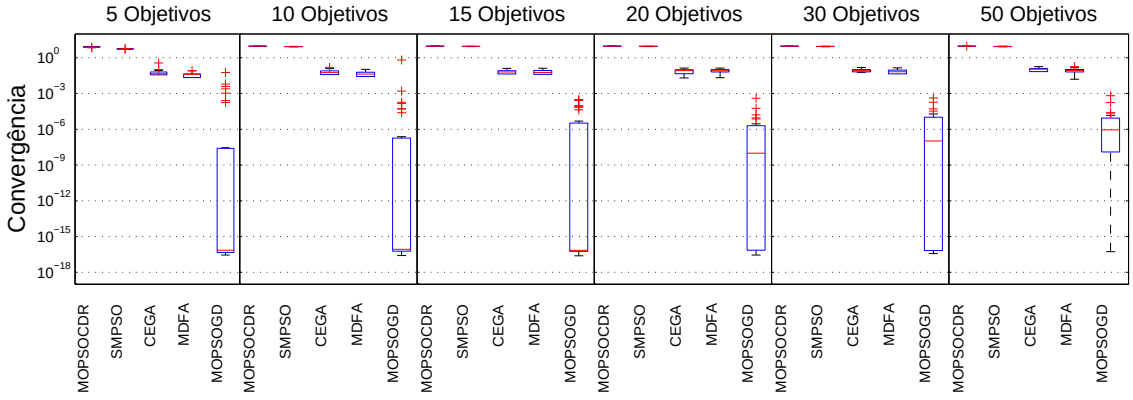


Figura 7.7 Métrica convergência para o problema DTLZ6.

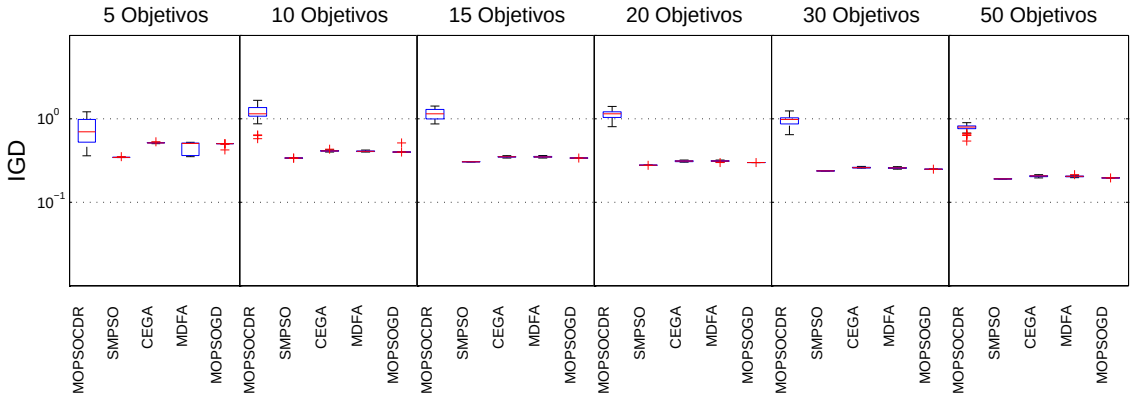


Figura 7.8 Métrica IGD para o problema DTLZ6.

7.1.5 Discussão dos Resultados para a Convergência e IGD

Os algoritmos MOPSO-CDR e SMPSO foram os que apresentaram piores valores em termos de convergência. Isto está de acordo com a literatura, no que diz respeito as desvantagens da dominância de Pareto. A explicação para essa incapacidade de convergência é que com o aumento do número de objetivos todas as soluções praticamente se tornam não-dominadas e o critério de escolha dos lideres nesses algoritmos é baseado somente no CD. O CD por sua vez privilegia soluções que estão muito longe dos seus vizinhos. Dessa forma, as soluções tendem a divergir uma das outras e assim a convergência para a Frente de Pareto é comprometida nesses algoritmos. Uma explicação desse mesmo fenômeno para o caso do NSGA II pode ser encontrada em [PF03]. O NSGA II assim como o MOPSO-CDR e o SMPSO utiliza o CD como estimador de densidade. Em suma, o MOPSO-CDR e o SMPSO além de não convergirem para a Frente de Pareto por causa da dominância de Pareto, eles podem acabar se afastando dela por causa do CD.

Por outro lado, quando métodos como o do GD e o do MDFA são usados, a convergência é privilegiada. Esse é o caso do MOPSO-GD que apresentou, em geral, uma melhor capacidade de convergência entre os algoritmos comparados. É importante destacar que o MOPSO-GD não tem um mecanismo explícito de diversidade. Por outro lado, a diversidade das soluções é mantida pelo uso da dominância de Pareto. Entretanto, quando o número de objetivos aumenta, a tendência é que as soluções do MOPSO-GD convirjam para um único ponto sobre a Frente de Pareto. No entanto, essa não é uma característica particular do MOPSO-GD; pois o MDFA e principalmente o CEGA tem um comportamento similar. Em suma, apesar da boa capacidade de convergência do MOPSO-GD, CEGA, MDFA, eles apresentam uma dificuldade em comum: todos eles convergem para uma região pequena e específica da Frente de Pareto.

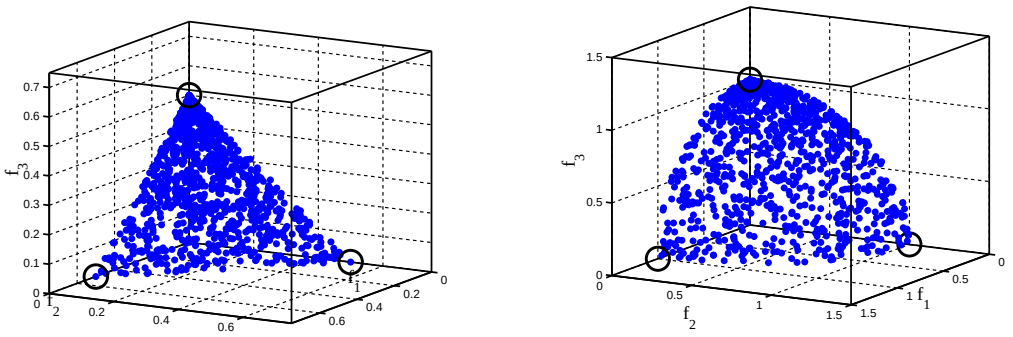
7.2 Análise Detalhada dos Resultados do CEGA, MDFA e MOPSO-GD

Nesta seção, os resultados do CEGA, MDFA, MOPSO-GD são discutidos em mais detalhes. Como foi discutido anteriormente, as soluções encontradas pelo MOPSO-GD, CEGA e MDFA tendem a convergir para uma região bastante pequena sobre a Frente de Pareto. Essas regiões

geralmente correspondem a soluções extremas da Frente de Pareto. Sendo assim, para explicar melhor os resultados obtidos, é necessário primeiro definir o que são soluções extremas. Para isso, a definição de Singh *et al.* [SIR11] de solução de canto (*corner solution*) será utilizada. Assim, uma solução extrema ou solução de canto é definida de acordo com a Definição 7.1:

Definição 7.1 (**Solução Extrema**)**.** Seja um problema multiobjetivo com m objetivos e considere um subconjunto F_k do conjunto de objetivos originais, com k objetivos ($k < m$). Se minimizando os k-objetivos neste subconjunto resulta em uma única solução no espaço objetivo com m dimensões, então esta solução é uma solução extrema da Frente de Pareto.

Nos problemas de teste considerados nesta dissertação, os problemas DTLZ{1,3,4} tem m soluções extremas, pois pode-se somente minimizar os objetivos em subconjuntos com $m-1$ objetivos. Por exemplo, para o problema DTLZ1 com três objetivos, pode-se somente minimizar simultaneamente os objetivos $\{f_1, f_2\}$, $\{f_1, f_3\}$ ou $\{f_2, f_3\}$. Para cada um desses subconjuntos correspondem soluções extremas sobre os eixos f_3, f_2 e f_1, respectivamente. A Figura 7.9(a) mostra as soluções extremas para o problema DTLZ1 com três objetivos e a Figura 7.9(b) mostra as soluções extremas para os problemas DTLZ1{3,4}.



(a) Problema DTLZ1. (b) Problema DTLZ{3,4}.

Figura 7.9 Soluções extremas dos problemas DTLZ1 e DTLZ{3,4} para três objetivos.

A Figura 7.10 mostra as soluções extremas para o problema DTLZ6 para um problema com três objetivos. Esse problema tem apenas 2 soluções extremas. Uma delas está sempre no eixo f_m (f_3, na figura) e a outra solução extrema está sobre o hiperplano $f_m = 0$ ($f_3 = 0$, na figura). Tomando o exemplo da Figura 7.10, isso ocorre porque somente é possível minimizar simultaneamente o par f_1 e f_2 de objetivos, o que resulta na solução extrema em f_3, e a função f_3 apenas, produzindo a solução extrema sobre o plano $f_3 = 0$.

Agora que a definição de solução extrema foi dada, pode-se explicar em mais detalhes os resultados obtidos pelos algoritmos. A análise será feita por número de objetivos. O número de objetivos considerados nessa análise foram 5 e 10. A partir de 10 objetivos, os resultados foram similares de modo que sua análise foi omitida por questões de espaço. Os resultados são apresentados em coordenadas paralelas nas quais os valores objetivos do conjunto aproximação *PF* dos algoritmos são projetados. As coordenadas paralelas são gráficos para visualização de dados em altas dimensões. Nesse gráfico o eixo horizontal representa cada um dos eixos do espaço objetivo, enquanto o eixo vertical representa o valor correspondente a cada um dos

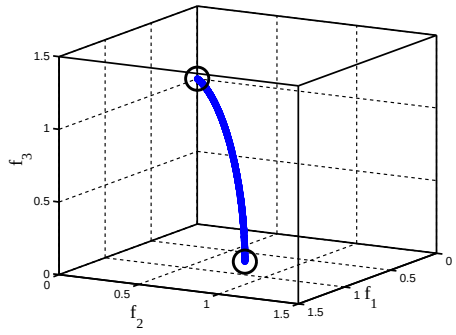


Figura 7.10 Soluções extremas do problema DTLZ6 para três objetivos.

objetivos. Cada solução, nesse gráfico, é representando por uma linha que intercepta cada um dos eixos objetivos no valor corresponde a esse objetivo. Mas especificamente, todos os *PF* das 31 execuções de cada algoritmo são apresentados simultaneamente para enfatizar a consistência (ausência de variabilidade) dos *PF* obtidos.

7.2.1 Análise Detalhada para Problemas com 5 Objetivos

Nesta seção, os problemas são analisados um por vez. O objetivo dessa análise é explicar em mais detalhes os resultados de convergência dos algoritmos. Além disso, essa análise permite conhecer para quais regiões da Frente de Pareto as soluções estão convergindo.

7.2.1.1 Problema DTLZ1

Como se observa na Figura 7.11, o MDFA conseguiu encontrar todas as soluções extremas considerando simultaneamente todas as execuções dos algoritmos. Ou seja, ele conseguiu encontrar os vetores $[0,5;0;0;0;0]$, $[0;0,5,0;0;0]$, $[0;0;0,5;0;0]$, $[0;0;0;0,5;0]$, e $[0;0;0;0;0,5]$. Além dessas soluções extremas, o MDFA conseguiu encontrar vários outros tipos de soluções. O CEGA, por sua vez, conseguiu encontrar apenas soluções sobre o eixo f_5, independentemente da execução do algoritmo. Em particular, ele encontrou a solução extrema sobre o eixo f_5, isto é, ele encontrou o vetor da forma $[0;0;0;0;0,5]$. O MOPSO-GD, por sua vez, conseguiu encontrar apenas um único tipo de solução sobre a Frente de Pareto.

7.2.1.2 Problema DTLZ3

A Figura 7.12 apresenta os resultados para o problema DTLZ3. Para este problema, o MDFA conseguiu encontrar soluções sobre os eixos f_4 e f_5. Algumas dessas soluções não pertencem à Frente de Pareto (muitas soluções apresentam um valor maior que 1 sobre o eixo f_5). No entanto, algumas dessas soluções são soluções extremas sobre os eixos f_4 e f_5. Assim o MDFA obteve um resultado enviesado no sentido de que apenas regiões específicas da Frente de Pareto foram exploradas. Uma explicação para esse viés é que, diferente do problema DTLZ1, a Frente de Pareto do problema DTLZ3 é côncava. Nesse tipo de problema, o MDFA não é capaz de encontrar qualquer solução, uma vez que ele é baseado no método da soma ponderada, que por

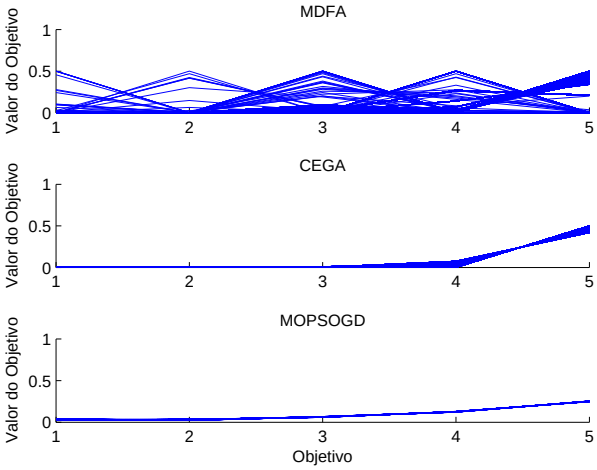


Figura 7.11 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ1 com 5 objetivos.

sua vez é incapaz de encontrar soluções em regiões não-convexas da Frente de Pareto [CLV07]. Outra explicação para o fato do MDFA convergir apenas para duas soluções extremas é que o problema DTLZ3 impõe uma grande dificuldade em termos de convergência. Nesse problema, o CEGA conseguiu encontrar soluções sobre o eixo f_5. Algumas dessas soluções pertencem à Frente de Pareto (solução extrema sobre o eixo f_5), enquanto outras não. O MOPSO-GD também conseguiu encontrar uma solução extrema sobre o eixo f_5. Mais especificamente, o MOPSO-GD foi o único que conseguiu encontrar apenas soluções sobre a Frente de Pareto, inclusive uma solução que não é extrema. Ou seja, cada vetor da forma $v = [f_1, f_2, f_3, f_4, f_5]$, que corresponde a uma linha do gráfico do MOPSO-GD, está aproximadamente sobre uma hiperesfera centrada na origem e com raio 1 ($\sqrt{\sum_{i=1}^{m} f_i^2} \sim 1$).

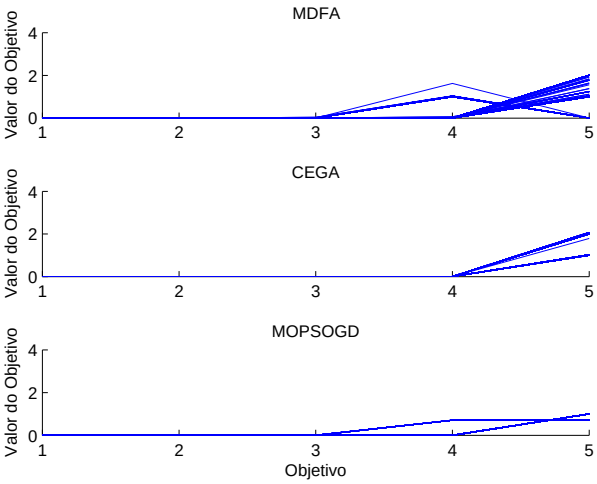


Figura 7.12 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ3 com 5 objetivos.

7.2.1.3 Problema DTLZ4

A Figura 7.13 apresenta os resultados para o problema DTLZ4. Como se pode observar por essa figura, o MDFA conseguiu encontrar todas as soluções extremas quando todas as execuções são consideradas simultaneamente. Neste problema, o CEGA conseguiu encontrar apenas soluções sobre um dos eixos: o eixo f_1. Algumas das soluções encontradas pelo CEGA estão sobre a Frente de Pareto, enquanto outras não. O MOSPO-GD também conseguiu encontrar todas as soluções extremas, considerando todas as execuções. Além disso, ele conseguiu encontrar muitas outras soluções além das soluções extremas como se observa pela figura. No gráfico do MOPSO-GD também se observa a presença de uma solução que não pertence à Frente de Pareto (uma solução apresenta um valor maior que 2 para o eixo f_5).

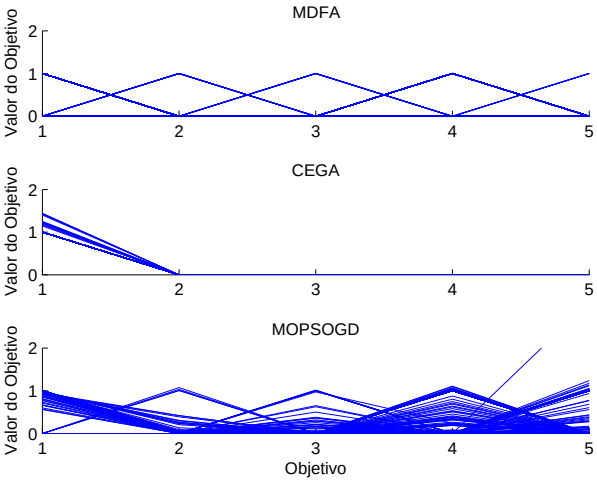


Figura 7.13 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ4 com 5 objetivos.

7.2.2 Problema DTLZ6

A Figura 7.14 apresenta os resultados para o problema DTLZ6. Como se pode observar por essa figura, o MOPSO-GD e o MDFA apresentaram resultados similares. No MOPSO-GD e no MDFA existem dois tipos de soluções. Essas soluções correspondem as duas únicas soluções extremas nesse problema, isto é, a solução sobre o eixo f_5 e sobre o hiperplano $f_5 = 0$. O CEGA, por sua vez, encontrou uma única solução extrema, a solução sobre o eixo f_5. No entanto, ele encontrou também soluções sobre o eixo f_5 mas que não pertencem à Frente de Pareto. Esse resultado mostra a ineficácia do mecanismo de diversidade do CEGA. Isso fica evidente quando se observa que o MOPSO-GD mesmo usando o GD encontra apenas soluções sobre a Frente de Pareto.

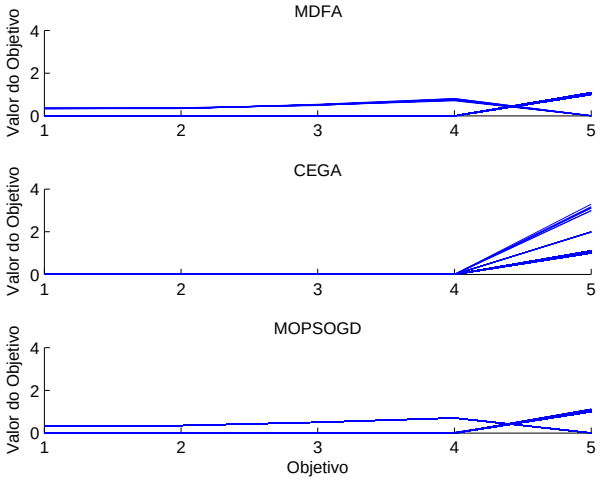


Figura 7.14 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ6 com 5 objetivos.

7.2.3 Análise Detalhada para Problemas com 10 Objetivos

7.2.3.1 Problema DTLZ1

A Figura 7.15 apresenta os resultados para o problema DTLZ1 com 10 objetivos. Nesse problema, o MOPSO-GD apresentou um comportamento diferente do CEGA e do MDFA. Como se observa, as soluções encontradas pelo MOPSO-GD não correspondem a uma solução extrema e estão muito próximas umas das outras. Assim, o MOPSO-GD obteve um comportamento similar a instância com 5 objetivos, conseguindo obter um único tipo de solução para o problema DTLZ1.

Com respeito ao MDFA, o MOPSO-GD encontrou apenas algumas soluções extremas, a saber, soluções extremas sobre os eixos f_7, f_8, f_9 e f_{10}. Comparando com a instância com 5 objetivos do problema em questão, observa-se que o MDFA não foi capaz de encontrar todas as soluções extremas, mesmo quando todas as execuções do algoritmo são consideradas simultaneamente. Isso pode ser explicado pelo fato do problema DTLZ1 se tornar cada vez mais difícil quando o número de objetivos aumenta. Além das soluções extremas, o MDFA conseguiu encontrar outros tipos de soluções como se pode observar pela figura.

O CEGA, por sua vez, conseguiu convergir para (ou próximo de) três soluções extremas, a saber, as soluções sobre os eixos f_8, f_9 e f_{10}. Além disso, o CEGA conseguiu encontrar outros tipos de soluções como se pode verificar pela figura.

7.2.3.2 Problema DTLZ3

A Figura 7.16 apresenta os resultados para o problema DTLZ3 com 10 objetivos. Como se pode observar pela figura, os algoritmos CEGA, MDFA e MOPSO-GD convergiram para soluções sobre o eixo f_{10}. Contudo, algumas das soluções encontradas pelo CEGA e pelo MDFA não pertencem à Frente de Pareto. Muitas das soluções possuem um valor maior que 1 sobre o eixo f_{10}. O mesmo não ocorre com o MOPSO-GD, que encontrou uma única solução sobre a

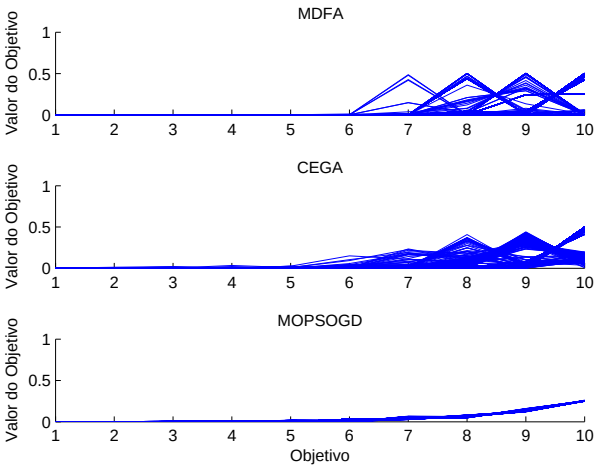


Figura 7.15 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ1 com 10 objetivos.

Frente de Pareto. Além disso, o MDFA conseguiu, diferentemente do CEGA e do MOPSO-GD, encontrar a solução extrema sobre o eixo f_9, embora ele tenha encontrado, em algumas execuções, soluções sobre o eixo f_9 e que não pertencem a Frente de Pareto. É importante mencionar novamente aqui que as soluções na Figura 7.16 são de execuções diferentes e que por isso não se pode fazer uma análise de dominância sobre elas. Ou seja, a presença de uma solução extrema sobre o eixo f_{10} com valor 1 não significa que ela foi obtida em uma mesma execução junto com uma solução sobre o eixo f_{10} com valor 2, por exemplo. Se isso ocorresse, a primeira solução dominaria a segunda.

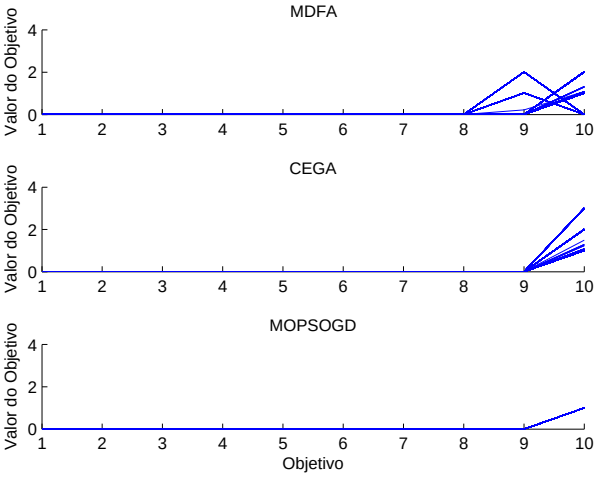


Figura 7.16 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ3 com 10 objetivos.

7.2.3.3 Problema DTLZ4

A Figura 7.17 apresenta os resultados para o problema DTLZ4. Como se pode observar, o CEGA e o MOPSO-GD obtiveram resultados iguais, isto é, em todas as execuções, todas as soluções do *PF* convergiram para a solução extrema sobre o eixo f_1. O MDFA, por outro lado, conseguiu encontrar todas as soluções extremas considerando todas as execuções em conjunto. No entanto, vale ressaltar, que considerando cada execução em separado, o MDFA conseguiu encontrar apenas algumas dessas soluções dependendo da inicialização do algoritmo.

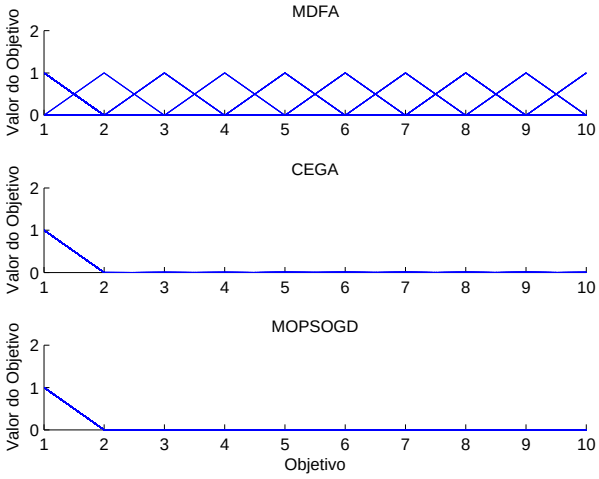


Figura 7.17 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ4 com 10 objetivos.

7.2.3.4 Problema DTLZ6

A Figura 7.18 apresenta os resultados para o problema DTLZ6. Como se pode observar por essa figura, o MDFA e o CEGA convergiram para a solução extrema sobre o eixo f_{10}. Diferentemente da instância com 5 objetivos para esse problema, o MDFA não conseguiu encontrar a outra solução extrema sobre o hiperplano $f_{10} = 0$. O MOPSO-GD também convergiu para a solução extrema sobre o eixo f_{10}. No entanto, o MOPSO-GD também encontrou uma solução que não pertence à Frente de Pareto como se observa pela figura.

7.2.4 Discussão dos Resultados sobre a Localização das Soluções nas Frentes de Pareto

Como foi discutido anteriormente, os algoritmos MOPSO-GD, CEGA, e MDFA tendem a convergir para regiões específicas sobre a Frente de Pareto. Encerrando a discussão entre o MOPSO-GD e o CEGA, isso ocorre pelo modo como o GD é definido. O GD é um método de atribuição de aptidão relativo, isto é, a aptidão de cada solução é determinada pelo desempenho do resto das soluções. Assim, como o objetivo é minimizar o GD, as soluções se atraem mutualmente para reduzir seu GD. O estado em que o GD das soluções é mínimo corresponde ao estado em que todas as soluções convergem para um único ponto no espaço objetivo. Entre esses pontos no espaço objetivo em que as soluções podem convergir, os mais estáveis são

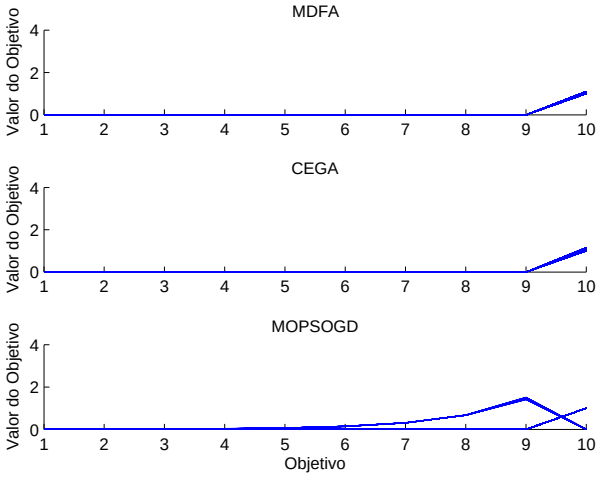


Figura 7.18 Aproximação da Frente de Pareto gerada pelos algoritmos para o problema DTLZ6 com 10 objetivos.

claramente os pontos sobre a Frente de Pareto, pois neles nenhum melhoramento é possível. Esse comportamento do GD explica a sua boa capacidade de atrair as soluções para a Frente de Pareto e portanto sua boa capacidade de convergência.

Por fim, é importante destacar que o desempenho do MOPSO-GD foi similar ou superior ao CEGA e ao MDFA em termos de convergência. Desse modo, o MOPSO-GD é uma técnica baseada em enxame que apresenta altos níveis de convergência, mesmo em problemas difíceis (com multimodalidade).

7.3 Análise do Tempo de Execução dos Algoritmos

Nesta seção, o tempo de execução dos algoritmos é analisado em todos os problemas em todas as instâncias. As Tabelas 7.1, 7.2, 7.3, 7.4, 7.5 e 7.6 apresentam os tempos de execução, em segundos, dos algoritmos para os problemas DTLZ{1,3,4,6} para 5, 10, 15, 20, 30 e 50 objetivos, respectivamente. Como se pode notar, ao se analisar todas as tabelas, o tempo de execução dos algoritmos aumenta à medida em que se aumenta o número de objetivos dos problemas. De todos os algoritmos, o CEGA é o que apresenta o maior tempo de execução em todos os problemas em todas as instâncias. Isso se deve principalmente ao algoritmo de agrupamento que ele utiliza. O MOPSO-GD é o segundo algoritmo com maior tempo de execução mas a diferença entre ele e o CEGA chega até duas ordens de grandeza. O maior tempo computacional do MOPSO-GD em relação ao SMPSO pode ser explicado em parte pelo fato do cálculo do GD ser mais custoso computacionalmente que o cálculo do CD. O GD tem complexidade $O(N^2)$ enquanto o CD tem complexidade $O(MNlogN)$ em que N é o tamanho do arquivo externo e M é o número de objetivos [DPAM02].

Tabela 7.1 Média e desvio padrão dos tempos de execução dos algoritmos para 5 objetivos.

Problema	MOPSOCDR	SMPSO	MDFA	CEGA	MOPSOGD
DTLZ1	1.1032	0.1587	0.6236	202.1663	6.5259
	0.12	0.05	0.02	3.85	0.45
DTLZ3	1.8374	0.2796	0.6397	358.9532	5.6704
	0.04	0.02	0.02	3.70	1.16
DTLZ4	1.7235	1.2975	0.7363	363.5105	13.0435
	0.07	0.09	0.01	3.02	1.28
DTLZ6	2.6920	0.6655	0.6217	341.3375	8.3633
	0.10	0.02	0.01	1.73	3.34

Tabela 7.2 Média e desvio padrão dos tempos de execução dos algoritmos para 10 objetivos.

Problema	MOPSOCDR	SMPSO	MDFA	CEGA	MOPSOGD
DTLZ1	2.7793	0.8099	1.1010	358.0378	15.1793
	0.38	0.09	0.02	4.75	1.09
DTLZ3	6.3037	1.4309	1.6761	581.1525	13.3357
	0.19	0.04	0.03	6.88	1.27
DTLZ4	7.7757	5.0893	2.1172	491.5983	42.9458
	0.22	0.17	0.04	3.31	1.34
DTLZ6	5.4948	1.8341	1.6515	396.3745	18.6179
	0.20	0.07	0.02	2.52	4.79

Tabela 7.3 Média e desvio padrão dos tempos de execução dos algoritmos para 15 objetivos.

Problema	MOPSOCDR	SMPSO	MDFA	CEGA	MOPSOGD
DTLZ1	7.2198	1.4376	1.6234	489.8751	18.5966
	1.01	0.14	0.02	4.87	2.72
DTLZ3	14.1592	2.7567	1.8517	738.1030	21.7206
	0.40	0.11	0.02	4.56	1.79
DTLZ4	16.3567	8.1438	2.5926	605.8264	62.0856
	0.22	0.20	0.02	3.94	0.72
DTLZ6	11.1652	2.8277	1.8428	573.4384	23.9055
	0.41	0.17	0.01	3.94	4.26

Tabela 7.4 Média e desvio padrão dos tempos de execução dos algoritmos para 20 objetivos.

Problema	MOPSOCDR	SMPSO	MDFA	CEGA	MOPSOGD
DTLZ1	9.6049	2.7176	1.7892	532.9716	23.1089
	1.62	0.22	0.02	5.80	4.01
DTLZ3	19.4286	4.6539	3.3968	644.2381	27.2948
	0.20	0.12	0.04	4.06	0.87
DTLZ4	21.7507	11.4613	4.9979	643.5418	84.2193
	0.18	0.13	0.14	4.88	0.84
DTLZ6	15.1373	4.1171	3.2008	698.0008	35.1276
	0.53	0.22	0.03	5.89	7.87

Tabela 7.5 Média e desvio padrão dos tempos de execução dos algoritmos para 30 objetivos.

Problema	MOPSOCDR	SMPSO	MDFA	CEGA	MOPSOGD
DTLZ1	15.7876	4.8113	3.1109	850.2313	32.8770
	3.15	0.71	0.02	9.39	6.67
DTLZ3	28.3449	8.3155	3.8281	980.6906	46.1011
	0.34	0.32	0.02	5.44	3.32
DTLZ4	32.9213	22.4742	6.7685	967.5902	133.6228
	0.46	0.34	0.07	8.07	6.10
DTLZ6	21.5566	6.2152	3.7892	810.6065	59.6103
	0.91	0.29	0.03	5.57	10.59

Tabela 7.6 Média e desvio padrão dos tempos de execução dos algoritmos para 50 objetivos.

Problema	MOPSOCDR	SMPSO	MDFA	CEGA	MOPSOGD
DTLZ1	23.3207	6.6855	5.1393	1328.5144	44.5351
	3.65	1.12	0.03	12.19	8.15
DTLZ3	37.0673	17.6078	9.5258	1796.0127	75.8020
	0.37	0.53	0.61	11.41	2.41
DTLZ4	46.4500	40.0498	18.3640	1281.0940	195.5221
	0.32	0.50	0.10	14.03	4.83
DTLZ6	28.6035	13.9318	9.0903	1388.7375	89.5053
	1.17	0.55	0.08	8.91	16.75

CAPÍTULO 8

Conclusão e Trabalhos Futuros

" *Viver não é necessário*
Necessário é criar. "
– Fernando Pessoa

8.1 Conclusão

Nessa dissertação foi abordado o tema da otimização de muitos objetivos que atualmente é uma área ativa e desafiadora no contexto da otimização multiobjetiva [ITN08]. Problemas com muitos objetivos, isto é, problemas multiobjetivos com quatro ou mais objetivos impõem várias dificuldades aos algoritmos que se propõem a resolvê-los. Em particular, MOEAs baseados em dominância de Pareto têm uma dificuldade intrínseca em resolver esses problemas. Isso ocorre porque a dominância de Pareto se torna ineficaz em discriminar as soluções, uma vez que todas as soluções de uma população se tornam não-dominadas à medida que o número de objetivos aumenta. Ou seja, à medida que o número de objetivos aumenta a probabilidade de uma solução dominar outra diminui rapidamente. Como todas as soluções tendem a se tornar não-dominadas, não existe uma pressão em direção à Frente de Pareto, isto é, esses algoritmos têm o seu desempenho de convergência mitigado.

Atualmente, os MOEAs baseados em dominância de Pareto são uma das abordagens mais promissoras e que tem sido aplicados em vários problemas reais [CLV07]. No entanto, esses algoritmos apenas funcionam bem para problemas multiobjetivos com dois ou três objetivos. Além dos MOEAs, existem também os MOPSOs baseados em dominância de Pareto que também obtiveram sucesso em problemas com poucos objetivos tanto teóricos como práticos [CLV07, BFFB$^+$11]. No entanto, como já foi mencionado, os MOEAs tem o seu desempenho mitigado em problemas com muitos objetivos. O mesmo ocorre com os MOPSOs baseados em dominância de Pareto pelas mesma razões que levam a ineficácia dos MOEAs. Por outro lado, muitos problemas práticos envolvem muitos objetivos e dessa forma existe a necessidade de adaptar tanto os MOEAs como os MOPSOs para esses problemas.

Recentemente, dois MOEAs foram propostos para lidar com problemas com muitos objetivos: o CEGA e o MDFA. Esses dois algoritmos apresentam uma alta capacidade de convergência em problemas com até 50 objetivos. Apesar desse avanço no desenvolvimento de MOEAs para problemas com muitos objetivos, ainda existe a necessidade de adaptar os MOPSOs para esses problemas. As poucas adaptações de MOPSOs para problemas com muitos objetivos presentes na literatura apresentam uma séria de dificuldades, tais como parâmetros difíceis de

ajustar, necessidade de informação do tomador de decisão, e dificuldade de convergência em problemas com multimodalidade.

Nesta dissertação, foi proposto um MOPSO para lidar com problemas multiobjetivo: o MOPSO-GD. A principal característica do MOPSO-GD é a sua alta capacidade de convergência mesmo em problemas com multimodalidade. O MOPSO-GD, para alcançar essa alta capacidade de convergência, utiliza um método de alta granularidade denominado de Detrimento Global. O MOPSO-GD foi comparado com dois MOPSOs baseados em dominância em Pareto, o SMPSO e o MOPSO-CDR, e dois MOEAs, o CEGA e o MDFA. Os resultados mostraram que o MOPSO-GD conseguiu obter níveis similares ou melhores de convergência que o CEGA o MDFA. Apesar dessa boa capacidade de convergência, o MOPSO-GD, assim como o MDFA e o CEGA, converge para uma região pequena e específica da Frente de Pareto. Mais geralmente, esse fenômeno é típico de algoritmos que utilizam métodos baseados em distância como métodos de atribuição de aptidão [CK07], como é o caso desses algoritmos. Esse fato motiva alguns trabalhos futuros (veja Seção 8.2) cujo objetivo deve ser melhorar a diversidade, mantendo a convergência, de MOPSOs e MOEAs em problemas com muitos objetivos.

Por fim, o MOPSO-GD, a apesar de ser avaliado nesta dissertação para um número de objetivos maior que cinco, pode ser aplicado em problemas com qualquer número de objetivos (≥ 2). Outro ponto que merece ser destacado é que o MOPSO-GD foi avaliado apenas em problemas teóricos. Assim, outro trabalho futuro é aplicar o MOPSO-GD em problemas práticos com muitos objetivos.

8.2 Trabalhos Futuros

Para se desenvolver MOPSOs e MOEAs que sejam capazes de aliar convergência e diversidade é necessário entender o que é diversidade no contexto de MaOPs. Para problemas com dois ou três objetivos, pode-se "entender" o que é diversidade. No entanto, em MaOPs esse "entendimento" é perdido. Uma explicação para esse fato se deve à incapacidade de se visualizar com facilidade a distribuição das soluções em altas dimensões. Além disso, com o aumento da dimensionalidade, o grau de liberdade das formas da Frente de Pareto é cada vez maior, ou seja, a Frente de Pareto pode assumir formas cada vez mais complexas. Assim, é necessário primeiramente estabelecer uma definição mais objetiva do que vem a ser diversidade no contexto de MaOPs. A partir disso, pode-se pensar em alternativas para o problema de perda de diversidade presente na maioria dos algoritmos desenvolvidos para resolver MaOPs.

Uma dificuldade em particular para estabelecer a diversidade em MaOPs é que nesses problemas é necessário, na maioria dos casos, utilizar milhares de pontos para aproximar a Frente de Pareto inteira com uma dada resolução. Para essa dificuldade, pode-se pensar que é desnecessário utilizar milhares de pontos para aproximar a Frente de Pareto, uma vez que o tomador de decisão é capaz de analisar apenas algumas soluções devido a limitações de tempo [LKC$^+$12]. Sendo assim, é mais importante focar em soluções estratégicas do que em várias regiões. Entre essas soluções, pode-se destacar as soluções extremas e as soluções nas regiões de joelho da Frente de Pareto.

Exemplos de soluções extremas são apresentados pelas Figuras 8.1 e 8.2. A Figura 8.1 apresenta as soluções extremas referentes a uma Frente de Pareto esférica. As soluções estão

dentro dos círculos. Como se pode observar nessa figura, todas as soluções extremas estão sobre os eixos coordenados. A Figura 8.2 apresenta as soluções extremas referentes a uma Frente de Pareto curva. Como se pode observar nessa figura, uma solução extrema está sobre o eixo coordenado f_3 e a outra solução extrema está sobre o plano $f_3 = 0$.

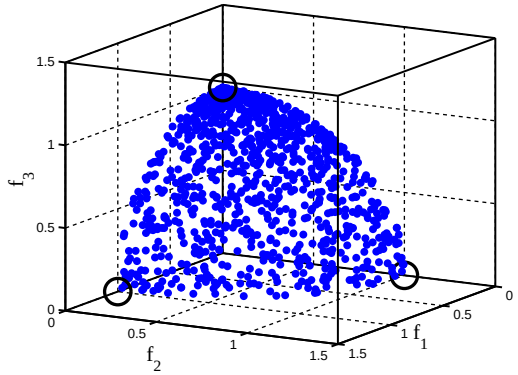


Figura 8.1 Soluções extremas (soluções dentro dos círculos) referentes a uma Frente de Pareto esférica.

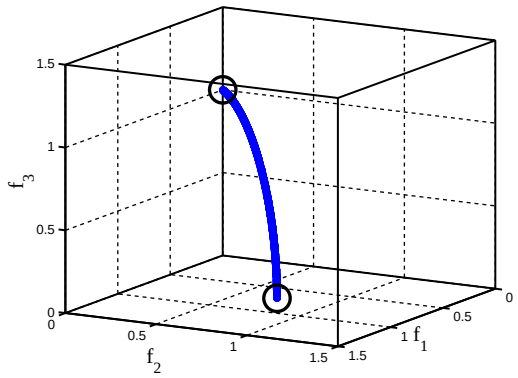


Figura 8.2 Soluções extremas (soluções dentro dos círculos) referentes a uma Frente de Pareto curva.

As soluções extremas oferecem vantagens, tais como:

1. elas podem permitir descobrir a dimensionalidade intrínseca da Frente de Pareto. Por exemplo, a Frente de Pareto da Figura 8.1 é uma superfície (2 dimensões) e tem três soluções extremas, enquanto a Frente de Pareto da Figura 8.2 é uma curva (1 dimensão) e tem duas soluções extremas [SIR11];

2. esses pontos correspondem aos pontos de maior diversidade e indicam toda a extensão da Frente de Pareto [SIR11].

As soluções na região do joelho, por sua vez, são soluções também importantes porque elas são geralmente as soluções escolhidas pelo tomador de decisão [BSG11]. Essas soluções representam os máximos *trade-offs* entre os objetivos. Ou seja, essas soluções são caracterizadas

pelo fato de que um pequeno melhoramento em um dos objetivos implica em uma grande deterioração de no mínimo um outro objetivo. Desse modo, na ausência de preferências explícitas do tomador de decisão, essas soluções são geralmente preferidas.

Assim, uma possível forma de promover a diversidade enquanto se mantém a convergência em MaOPs é desenvolver técnicas que busquem pelas soluções extremas e pelas soluções na região do joelho da Frente de Pareto. Com isso, se obterá a extensão da Frente de Pareto bem como o conjunto de soluções representando as regiões de máximo *trade-offs*. Após encontrar essas soluções, o tomador de decisão pode obter intuições sobre o perfil da Frente de Pareto e assim fornecer pontos de referência próximos dela e, com isso, MOEAs ou MOPSOs baseados em preferência podem se utilizados para focar a busca em torno desses pontos.

Desta forma, alguns trabalhos futuros são:

- Desenvolver algoritmos para encontrar as soluções extremas em problemas com muitos objetivos;
- Desenvolver um algoritmo para encontrar as soluções nas regiões de joelho da Frente de Pareto em problemas com muitos objetivos;
- Desenvolver um algoritmo para realizar as duas tarefas acima simultaneamente;
- Combinar a última técnica com algoritmos baseados em preferência;
- Avaliar os algoritmos desenvolvidos em um maior número de problemas de teste, incluindo problemas com Frentes de Pareto descontínuas [DTLZ05];
- Desenvolver métricas que possam avaliar os algoritmos mais adequadamente em problemas com muitos objetivos;
- Avaliar os algoritmos em problemas práticos em cenários com muitos objetivos.

Referências Bibliográficas

[Adr07] S. F. Adra. *Improving Convergence, Diversity and Pertinency in Multiobjective Optimisation*. PhD thesis, University of Sheffield, 2007.

[AEH09] M. R. AlRashidi and M. E. El-Hawary. A Survey of Particle Swarm Optimization Applications in Electric Power Systems. *IEEE Transactions on Evolutionary Computation*, 13(4):913–918, 2009.

[AT09] H. Aguirre and K. Tanaka. Adaptive e-ranking on many-objective problems. *Evolutionary Intelligence*, 2:183–206, 2009.

[Aze11] Carlos Renato Belo Azevedo. Geração de diversidade na otimização dinâmica multiobjetivo evolucionária por paisagens de não-dominância. Master's thesis, Universidade Federal de Pernambuco, 2011.

[BFFB+11] C. J. A. Bastos-Filho, E. M. N. Figueiredo, E. A. Barboza, J. F. Martins-Filho, M. E. V. Segatto, S. Cani, and M. J. Pontes. Simple design of raman fiber amplifiers using a multi-objective optimizer. In *Proceedings of the 11th International Conference on Intelligent Systems Design and Applications*, ISDA'11, pages 1128–1133, 2011.

[BHS97] T. Back, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[BK07] D. Bratton and J. Kennedy. Defining a standard for particule swarm optimization. In *IEEE Symposium on Swarm Intelligence*, SIS' 2007, pages 120–127, 2007.

[BNE07] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[BSG11] S. Bechikh, L. B. Said, and K. Ghédira. Searching for knee regions of the pareto front using mobile reference points. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15:1807–1823, 2011.

[BW97] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *(Chawdry et al, eds.) Soft Computing in Eng Design and Manufacturing*, pages 231–240, 1997.

[CK02] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[CK07] D. W. Corne and J. D. Knowles. Techniques for highly multiobjective optimisation: some nondominated points are better than others. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 773–780, 2007.

[CLV07] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, USA, 2007.

[Coe99] C. A. Coello Coello. A survey of constraint handling techniques used with evolutionary algorithms. Technical report, Laboratorio Nacional de Informática Avanzada, 1999.

[Coe06] C. A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, 2006.

[CPL04] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004.

[DA95] K. Deb and R. B. Agrawal. Simulated bunary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[Dar59] C. Darwin. *On The Origin of Species by Means of Natural Selection*. John Murray, London, 1859.

[DDB01] N. Drechsler, R. Drechsler, and B. Becker. Multi-objective Optimisation Based on Relation Favour. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, EMO '01, pages 154–166, 2001.

[Deb00] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186:311–338, 2000.

[DG96] K. Deb and M. Goyal. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[DGNN+09] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba. Multi-Objective Particle Swarm Optimizers: An Experimental Comparison. In *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2009)*, EMO '09, pages 495–509, 2009.

[DKS07] F. Di Pierro, S-T. Khu, and D. A. Savic. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45, 2007.

[DN11] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[DP12] A. B. De Carvalho and A. Pozo. Measuring the convergence and diversity of CDAS Multi-Objective Particle Swarm Optimization Algorithms: A study of many-objective problems. *Neurocomputing*, 75(1):43–51, 2012.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[DS05] K. Deb and D. K. Saxena. On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. Technical report, Kanpur Genetic Algorithms Laboratory (KanGAL) Indian Institute of Technology Kanpur, 2005.

[DS06] K. Deb and J. Sundar. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 635–642, 2006.

[DTLZ05] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, pages 105–145. Springer, 2005.

[Edg81] F. Y. Edgeworth. *Mathematical Physics*. P. Keagan, London, England, 1881.

[Eng07] A. P. Engelbrecht. *Computational Inteligence An Introduction*. John Wiley Sons, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2007.

[FA04] M. Farina and P. Amato. A fuzzy definition of "optimality"for many-criteria optimization problems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(3):315–326, 2004.

[FF93] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, 1993.

[FF95] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[FF98] C. M. Fonseca and P. J. Fleming. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms–Part I: A Unified Formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):26–37, 1998.

[FLM+13] E. Fernandez, E. Lopez, G. Mazcorro, R. Olmedo, and C. A. Coello Coello. Application of the non-outranked sorting genetic algorithm to public project portfolio selection. *Information Sciences*, 228:131–149, 2013.

[FPL05] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe. Many-objective optimization: an engineering design perspective. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 14–32, 2005.

[Gar09] M. Garza Fabre. Optimización de problemas con más de tres objetivos mediante algoritmos evolutivos. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Ciudad Victoria, Tamaulipas, México, 2009.

[GPC09] M. Garza Fabre, G. T. Pulido, and C. A. Coello Coello. Ranking methods for many-objective optimization. In *Proceedings of the 8th Mexican International Conference on Artificial Intelligence*, MICAI '09, pages 633–645, 2009.

[GPC10] M. Garza Fabre, G. T. Pulido, and C. A. Coello Coello. Two novel approaches for many-objective optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 1–8, 2010.

[HNG94] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Conference on Evolutionary Computation*, pages 82–87, 1994.

[Hug05] E. J. Hughes. Evolutionary many-objective optimisation: many once or one many? In *Proceedings of the Congress on Evolutionary Computation*, pages 222–227, 2005.

[ITN08] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *Proceedings of the Congress on Evolutionary Computation*, pages 2419–2426, 2008.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE Internacional Conference on Neural Networks*, pages 1945–1948, 1995.

[KG10] S. Kachroudi and M. Grossard. Average rank domination relation for NSGAII and SMPSO algorithms for many-objective optimization. In *2nd World Congress on Nature and Biologically Inspired Comp. (NaBIC)*, pages 19–24, 2010.

[Kha02] V. Khare. Performace scaling of multi-objective evolutionary algorithms. Master's thesis, The University of Birmingham, 2002.

[KY04] M. Köppen and K. Yoshida. A fuzzy scheme for the ranking of multivariate data and its application. In *Proceedings of the 2004 Annual Meeting of the NAFIPS*, pages 140–145, 2004.

[KY07] M. Köppen and K. Yoshida. Many-objective particle swarm optimization by gradual leader selection. In *Adaptive and Natural Computing Algorithms*, volume 4431, pages 323–331. Springer, 2007.

[LCF10] R. J. Lygoe, M. Cary, and P. J. Fleming. A many-objective optimisation decision-making process applied to automotive diesel engine calibration. In *Proceedings of the 8th international conference on Simulated evolution and learning*, SEAL'10, pages 638–646, 2010.

[LJC09] A. López Jaimes and C. A. Coello Coello. Some techniques to deal with many-objective problems. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 2693–2696, 2009.

[LKC+12] K. Li, S. Kwong, J. Cao, M. Li, J. Zheng, and R. Shen. Achieving balance between proximity and diversity in multi-objective evolutionary algorithm. *Information Sciences*, 182(1):220–242, 2012.

[MF00] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[MS08] S. Mostaghim and H. Schmeck. Distance based ranking in many-objective particle swarm optimization. In *Parallel Problem Solving from Nature PPSN X*, volume 5199, pages 753–762. Springer, 2008.

[NBE05] B. Naujoks, N. Beume, and M. Emmerich. Multi-objective optimisation using s-metric selection: application to three-dimensional solution spaces. In *Congress on Evolutionary Computation*, pages 1282–1289, 2005.

[NDG+09] A. J. Nebro, J. J. Durillo, J. García-Nieto, C. A. Coello Coello, F. Luna, and E. Alba. SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In *IEEE Symposium On Computational Intelligence in Multi-Criteria Decision-Making*, pages 66–73, 2009.

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.

[Par96] V. Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[PF03] R. C. Purshouse and P. J. Fleming. Evolutionary many-objective optimisation: an exploratory analysis. In *Proceedings of the Congress on Evolutionary Computation*, pages 2066–2073, 2003.

[PSG10] C. Peng, H. Sun, and J. Guo. Multi-objective optimal PMU placement using a non-dominated sorting differential evolution algorithm. *International Journal of Electrical Power & Energy Systems*, 32(8):886–892, 2010.

[Pur03] R. C. Purshouse. *On the Evolutionary Optimisation of Many Objectives*. PhD thesis, University of Sheffield, 2003.

[Rao09] S. S. Rao. *Engineering Optimization Theory and Practice*. John Wiley & Sons, 2009.

[RHH+13] P. M. Reed, D. Hadka, J. D. Herman, J. R. Kasprzyk, and J. B. Kollat. Evolutionary multiobjective optimization in water resources: The past, present, and future. *Advances in Water Resources*, 51(0):438–456, 2013.

[Rid04] M. Ridley. *Evolution*. Blackwell Publishing, 2004.

[RN05] C. R. Raquel and P. C. Naval Jr. An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 257–264, 2005.

[RS06] L. Rachmawati and D. Srinivasan. Preference incorporation in multi-objective evolutionary algorithms: A survey. In *Proceedings of the Congress on Evolutionary Computation*, pages 962–968, 2006.

[RSC06] M. Reyes-Sierra and C. A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *Int. Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[SAT07] H. Sato, H. E. Aguirre, and K. Tanaka. Controlling dominance area of solutions and its impact on the performance of MOEAs. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 5–20, 2007.

[Sch84] J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[Sch85] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms Genetic Algorithms and their Applications*, pages 93–100, 1985.

[SCO13] SCOPUS. Busca de Documentos do SCOPUS Database, 2013. Disponível em `www.scopus.com/home.url`. Acessado em 11/05/2013.

[SD94] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[SD07] D. K. Saxena and K. Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: employing correntropy and a novel maximum variance unfolding. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 772–787, 2007.

[SE98a] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the Congress on Evolutionary Computation*, pages 69–73, 1998.

[SE98b] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In *Proceedings of the 7th International Conference on Evolutionary Programming VII*, pages 591–600, 1998.

[Sek10] Z. Sekulski. Multi-objective topology and size optimization of high-speed vehicle-passenger catamaran structure by genetic algorithm. *Marine Structures*, 23(4):405–433, 2010.

[SIR11] H. K. Singh, A. Isaacs, and T. Ray. A pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, 15(4):539–556, 2011.

[SPBF09] R. A. Santana, M. R. Pontes, and C. J. A. Bastos-Filho. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications*, ISDA'09, pages 237–242, 2009.

[Tal09] E.-G. Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[Tre03] I. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003.

[Van06] F. Van Den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, 2006.

[VE06] F. Van Den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006.

[WBN07] T. Wagner, N. Beume, and B. Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Proceedings of the 4th international Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 742–756, 2007.

[WL08] U. Wickramasinghe and X. Li. Choosing leaders for multi-objective pso algorithms using differential evolution. In *Proceedings of the 7th International Conference on Simulated Evolution and Learning*, SEAL '08, pages 249–258, 2008.

[WL09] U. Wickramasinghe and X. Li. Using a distance metric to guide PSO algorithms for many-objective optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 667–674, 2009.

[Yan10] X-S Yang. *Engineering Optimization:An Introduction with Metaheuristic Applications*. John Wiley & Sons, 2010.

[Zit99] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Institut für Technische Informatik und Kommunikationsnetze, 1999.

[ZK04] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, pages 1–10, 2004.

[ZLT01] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Swiss Federal Institute of Technology (ETH), 2001.

[ZQL+11] A. Zhou, B-Y. Qu, H. Li, S-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[ZTL+03] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.