UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

PEDRO DA COSTA LOPES
Orientadora: Prof. Dra. Dayanne Gouveia Coelho

PBMM: ADAPTAÇÃO DOS PARÂMETROS DO ALGORITMO MOPSO-CD POR MEIO DE UM OPERADOR BASEADO EM MEMÓRIA

Ouro Preto, MG
2020

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

PEDRO DA COSTA LOPES

PBMM: ADAPTAÇÃO DOS PARÂMETROS DO ALGORITMO MOPSO-CD POR MEIO DE UM OPERADOR BASEADO EM MEMÓRIA

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Prof. Dra. Dayanne Gouveia Coelho

Ouro Preto, MG
2020

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

L864p Lopes, Pedro da Costa .
PBMM [manuscrito]: adaptação dos parâmetros do algoritmo mopso-cd por meio de um operador baseado em memória. / Pedro da Costa Lopes. - 2021.
52 f.

Orientadora: Profa. Dra. Dayanne Gouveia Coelho.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Otimização . 2. Heurística. 3. Parâmetros - Adaptação . I. Coelho, Dayanne Gouveia. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Celina Brasil Luiz - CRB6-1589

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Pedro da Costa Lopes

PBMM: Adaptação dos parâmetros do algoritmo MOPSO-CD por meio de um operador baseado em memória

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 28 de Abril de 2021.

Membros da banca

Dayanne Gouveia Coelho (Orientadora) - Doutora - Universidade Federal de Ouro Preto
Rodrigo César Pedrosa Silva (Examinador) - Doutor - Universidade Federal de Ouro Preto
Pedro Henrique Lopes Silva (Examinador) - Mestre - Universidade Federal de Ouro Preto

Dayanne Gouveia Coelho, Orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 28/04/2021.

Documento assinado eletronicamente por **Dayanne Gouveia Coelho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 28/04/2021, às 13:55, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0162973** e o código CRC **BB96C952**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Escava dentro de ti. É lá que está a fonte do bem, e esta pode jorrar continuamente, se a escavares sempre. [...] O universo muda; a nossa vida é aquilo que os nossos pensamentos fizerem dela.

Marco Aurélio

Agradecimentos

Agradeço à minha família por me proporcionarem um ambiente estável para o meu desenvolvimento como ser humano, por fazerem parte de cada etapa da minha vida, desde o meu nascimento até o que sou hoje, sempre com atenção, carinho e amor.

Ao meu pai Vanuci, por estar sempre presente nas boas e más decisões tomadas por mim e por ter me apresentado a área da computação.

À minha mãe Maria por me tornar um ser mais extrovertido, positivo e corajoso, sempre me incentivando a ir além.

Ao meu irmão Gabriel por ser o meu segundo pai nos momentos difíceis e também por ser o meu melhor amigo.

À Universidade Federal de Ouro Preto por proporcionar uma educação de qualidade e gratuita. Em especial ao Departamento de Computação e seus professores acima da média. Muitos aprendizados em um só lugar.

Aos majestosos integrantes da melhor chapa que o Centro Acadêmico do Departamento de Computação jamais possuirá. Em especial à estes amigos que fizeram parte e tornaram minha caminhada na universidade muito mais fácil. Viva a KOXAPA!

À eterna República Pureza por me acolher durante esta longa e cansativa jornada de cinco anos em Ouro Preto. Meu sincero obrigado à todos os irmãos que aqui moraram.

Aos eternos amigos de infância da grande Belo Horizonte, que estão sempre dispostos a tornar cada um do grupo as melhores versões de si mesmos.

À Avenue Code pela primeira oportunidade de emprego me concedida. Um sincero agradecimento pela confiança para o trabalho.

Por fim, mas não menos importante, à mim mesmo por possuir a resiliência, perseverança e a vontade de deixar os meus sonhos cada vez mais próximos de serem alcançados.

Resumo

Este trabalho propõe o uso de um operador de memória para fazer a adaptação dos parâmetros do algoritmo *Multi-objective Particle Swarm Optimization with Crowding Distance* (MOPSO-CD). O operador proposto, intitulado como *Parameter Adaptation Based on Meta-Heuristics Memory* (PBMM), utiliza uma estrutura de dados tabela *hash*, onde cada posição contém uma lista duplamente encadeada, com o objetivo de armazenar todas as soluções avaliadas, e usar estas soluções para realizar a adaptação dos parâmetros do algoritmo, melhorando o processo de exploração pelo espaço de busca. O operador, além da memória, utiliza de resultados de testes empíricos da literatura para realizar uma pertubação nos parâmetros do algoritmo. Esta pertubação é realizada nos parâmetros c_1, c_2 e ω como forma de aumentar o espaço da busca, aumentando a diversidade e convergência das soluções produzidas. Para verificar a eficiência do método proposto, foram realizados testes com 15 problemas testes disponíveis na literatura usando as métricas de desempenho hiper-volume (HV), *Generation Distance* (GD) e *Inverse Generation Distance* (IGD). Os resultados obtidos mostram uma melhora nestas três métricas de desempenho quando comparado ao algoritmo original. Porém, também foi mostrado que o aumento destas métricas ocorreu em detrimento do tempo de execução do algoritmo, visto que o processamento em cada avaliação de função objetivo é maior e mais complexo. Por fim, destaca-se que o resultado deste trabalho foi satisfatório por ser possível visualizar a repetição na produção de soluções e utilizar esta informação para realizar a adaptação dos parâmetros, e portanto melhorar o processo de exploração do espaço de busca.

Palavras-chave: Otimização multi-objetivo, meta-heurísticas,adaptação dos parâmetros, operador de memória, MOPSO-CD.

Abstract

This paperwork proposes the usage of memory operator to automatically adapt the parameters for the algorithm *Multi-Objective Particle Swarm Optimization with Crowding Distance* (MOPSO-CD). The proposed operator named *Parameter Adapatation Based on Meta-heuristics Memory* (PBMM) uses a double-linked *hash* table to store the evaluated solutions, and uses that information to adapt the parameters, improving the exploration and exploitation in the search space. In adittion to the memory, the operator uses some empirical tests results to produce an pertubation in each of the parameters. This pertubation is done on c_1, c_2 and ω parameters in order to expand the search space, and thus improving the convergence and the diversity of the solutions. To verify our method efficiency, we used 15 test functions to produce the value for Hypervolume (HV), Generational Distance (GD) and Inverse Generational Distance (IGD) metrics. The results obtained shows a improvement in those three perfomance metrics when compared to the original algorithm. However, the good perfomance in those three metrics lead to a greater execution time of the algorithm since the processing in each evaluation is more complex. Therefore, the results contained in this paperwork was satisfactory because was possible to visualize the repetitive solutions produced and could use this information to automatically adapt the parameters, and thus improve the process of search space exploitation.

Keywords: Multi-objective optimization, Heuristics, parameter adaptation, memory operators, MOPSO-CD.

Lista de Ilustrações

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

BSP	*Binary Space Partitioning*
CD	*Crowding Distance*
GD	*Generational Distance*
HV	*Hypervolume*
IGD	*Inverse Generational Distance*
MOEA	*Multi-Objective Evolutionary Algorithm*
MOP	*Multi-Objective Problem*
MOPSO-CD	*Multi-Objective Particle Swarm Optimization with Crowding Distance*
platEMO	*Evolutionary multi-objective optimization platform*
PCCS	*Parallel Cell Coordinate System*
PBMM	*Parameter Adaptation based on Meta-Heuristics*
PSO	*Particle Swarm Optimization*
VNS	*Variable Neighborhood Search*

Lista de Símbolos

$\mathcal{X}$	Conjunto factível
$\mathbb{R}^n$	Espaço das variáveis de decisão
$\mathbb{R}^m$	Espaço das funções objetivo
$\preceq$	Operador de dominância
$\npreceq$	Operador de dominância negado
$\mathcal{P}$	Conjunto Pareto-Ótimo
$\mathcal{FP}$	Fronteira de Pareto
ω	Peso de inércia
c_1	Parâmetro cognitivo
c_2	Parâmetro social
$pBest$	Direção do melhor local
$gBest$	Direção do melhor global
$\oplus$	Operador ou exclusivo
$bin2dec$	Operador de conversão de binário para decimal
$dec2bin$	Operador de conversão de decimal para binário

Sumário

1 Introdução

O estudo de metodologias de otimização é realizado de forma sistemática desde 1939, quando Kantorovich (1939) propôs um método até então inédito de resolução de um problema de planejamento de produção, intitulado como Programação Não-Linear. É interessante citar este autor, pois aqui iniciava-se o estudo para a resolução de problemas intangíveis para a matemática clássica. Dantzig (1951) também propõe um trabalho bastante importante para a literatura, quando mostra uma outra abordagem para a otimização mono-objetivo, que também utiliza o conceito de Programação Linear. O método foi denominado como *Simplex* e foi criado durante o planejamento dos processos que Dantzig realizava nas Forças Armadas Americanas, em 1947.

O Princípio da Dualidade, apresentado pelos autores Gale, Kuhn e Tucker (1951), foi outro conceito importante para o crescimento desta área de estudo. Este que define que problemas de otimização podem ser vistos por duas perspectivas: a primal e a dual. A solução de um problema dual é tratado como o limitante inferior do problema primal, este último que é exatamente o problema em que se busca uma solução ótima.

Outro passo importante para um apropriamento do estudo de otimização foi a definição das condições de otimalidade de um problema de otimização não-linear, conhecidas como as Condições de Karush-Kuhn-Tucker, ou somente KKT, e apresentadas pelos autores Karush, Kuhn e Tucker (1951). Este princípio também se torna importante para a evolução do estudo de otimização, visto que este provê um meio de verificação de possibilidade de otimização de um determinado problema.

Outro princípio proposto por Bellman (2003) também deve ser considerado no avanço da otimização por se tratar de uma simplificação de problemas complexos de otimização em outros sub-problemas mais simples que também possuem estrutura otimizável. O conceito foi nomeado como Programação Dinâmica pelo próprio autor numa época onde a mecanização do planejamento era um objeto de estudo bastante aplicado.

Vários outros métodos foram propostos após estes acontecimentos, como por exemplo aqueles citados na seção 2.1.1. Os Métodos Clássicos nos dão uma forma de resolver problemas de otimização utilizando os conceitos clássicos da programação linear e não-linear, das condições do KKT, dos princípios da dualidade e da técnica de programação dinâmica, assim como vários outros teoremas da matemática clássica. Este tipo de abordagem, apesar de ser efetiva em determinados problemas, com o passar dos anos verificava-se que em alguns problemas, estes métodos não se aplicavam de forma eficiente.

Um outro grande avanço na área de otimização foram através da criação das meta-heurísticas. As meta-heurísticas utilizam a melhor aproximação para a solução ótima e não a solução ótima exata, tornando uma solução ótima mais fácil de ser encontrada. Outro aspecto

importante é que as meta-heurísticas visam a otimização global e possuem possibilidade de computação paralela. Logo, a otimização se torna mais rápida e produz soluções melhores.

Este trabalho apresenta uma revisão dos principais conceitos de otimização multi-objetivo e uma breve descrição do algoritmo evolucionário utilizado para tratar problemas desta natureza, com destaque para o algoritmo *Multi-Objective Particle Swarm Optimization with Crowding Distance* (MOPSO-CD). Além da fundamentação teórica, é feita uma revisão da literatura apresentado os trabalhos relacionados à proposta desta monografia, com o intuito de investigar os trabalhos que identificam a aplicação de memórias em meta-heurísticas e adaptação de parâmetros do algoritmo MOPSO-CD.

Neste contexto de adaptação dos parâmetros vale ressaltar algumas definições a cerca de configuração de parâmetros em meta-heurísticas. Como mais detalhado na subseção 2.2.1, a configuração de parâmetros pode ser dividida em duas grandes áreas: ajuste de parâmetros e controle de parâmetros. Este último que pode também ser sub-dividido em controle determinístico, controle adaptativo e controle auto-adaptativo. Neste contexto, o objetivo do trabalho consiste na proposta de um operador para realizar a adaptação dos parâmetros do algoritmo MOPSO-CD utilizando uma estrutura de memória.

Para validar a metodologia proposta, são utilizados 15 problemas testes disponíveis na literatura e os resultados obtidos são comparados com resultados obtidos da versão clássica desse algoritmo considerando as métricas de desempenho hiper-volume, *Inverse Generation Distance* e *Generation Distance*.

A justificativa deste trabalho se dá, principalmente, pelo fato de ser um campo de estudo atual, especificamente sobre adaptação de parâmetros em meta-heurísticas. Poucos trabalhos citam formas de desenvolvimento de uma memória algorítmica para adaptação de parâmetros dos algoritmos em questão . Outra justificativa para este trabalho é a possibilidade de melhora na exploração do espaço de busca em meta-heurística, permitindo a visita por regiões pouco exploradas, e portanto, criando um algoritmo mais eficiente.

1.1 Objetivos

O objetivo principal deste trabalho consiste na proposta do operador para adaptação de parâmetros baseada em memória, denominado por *Parameter Adaptation based on Meta-Heuristics Memory* (PBMM). O operador proposto consiste de uma tabela *hash* utilizada para realizar uma pertubação nos parâmetros do algoritmo MOPSO-CD.

Para tanto, temos como objetivos específicos neste trabalho:

- Implementação da memória algorítmica que utiliza a estrutura de dados tabela *hash* para armazenar soluções já avaliadas.

- Implementação da versão do algoritmo intitulado como memMOPSO-CD, que consiste na versão clássica do MOPSO-CD com o acréscimo do operador de memória PBMM para adaptação dos seus parâmetros.
- Realizar a adaptação automática dos parâmetros c_1, c_2 e ω.
- Validação dos algoritmos utilizando problemas testes disponíveis na literatura.
- Estudo e aplicação de métricas de desempenho para medir a qualidade das soluções geradas pelas meta-heurísticas.

1.2 Organização do Trabalho

Esta monografia está dividida em 5 capítulos e, o restante do texto, encontra-se organizado como segue. O Capítulo 2 contém a fundamentação teórica deste trabalho apresentando os principais conceitos de otimização multi-objetivo, uma descrição do algoritmo MOPSO-CD, conceitos a cerca da configuração de parâmetros em meta-heurísticas e algumas métricas de desempenho utilizadas para medir a qualidade das soluções geradas por algoritmos de otimização multi-objetivo.

O Capítulo 3 apresenta uma breve revisão da literatura acerca de trabalhos relacionados com a proposta desta monografia. São apresentados os principais estudos acerca da adaptação dos parâmetros do algoritmo MOPSO-CD e alguns estudos que utilizam estratégia de memória em meta-heurísticas. Ainda neste capítulo é apresentado o operador de Adaptação de Parâmetros Baseada em Memória de Meta-heurísticas (do inglês, *Parameter Adaptation based on Meta-Heuristics Memory*, ou somente PBMM) .

No Capítulo 4 são descritos os experimentos computacionais realizados. São apresentados os problemas testes da literatura que serão utilizados para validação da metodologia proposta, e os resultados finais alcançados. Apresenta-se também neste capítulo uma análise comparativa entre os métodos considerando três métricas de desempenho.

Por fim, o Capítulo 5 apresenta as considerações finais sobre o trabalho desenvolvido e as propostas de trabalhos futuros.

2 Fundamentação Teórica

O propósito deste capítulo é introduzir os princípios básicos da otimização multi-objetivo. Estes conceitos dizem respeito à forma de otimizar problemas que apresentam mais de uma função objetivo. Assim, na Seção 2.1 é apresentada a formalização de um problema multi-objetivo, e também introduziremos alguns métodos diretos de resolução; na Seção 2.2 é explicado o funcionamento dos algoritmos evolucionários; e finalmente, na seção 2.3 são mostrados os conceitos sobre conjuntos de soluções aproximadas e medidas de desempenho. Os métodos clássicos são formulações matemáticas que iterativamente produzem novos vetores em direção do vetor ótimo. Ao contrário dos métodos mostrados na seção 2.2 que utilizam a relação de dominância na construção das soluções, os métodos clássicos produzem estas soluções utilizando informações específicas sobre o problema.

Para melhor compreensão dos conceitos expostos neste capítulo, considere as seguintes notações:

$\mathbb{R}^n$**:** Espaço das variáveis de decisão

$g_j(x)$**:** j-ésima restrição de desigualdade

$h_k(x)$**:** k-ésima restrição de igualdade

$\mathcal{X}$**:** Espaço das variáveis de decisão factíveis

f_i**:** i-ésima função objetivo do problema

$\mathbb{R}^m$**:** Espaço das funções objetivo

$\mathcal{P}$**:** Conjunto Pareto-Ótimo

$\mathcal{FP}$**:** Fronteira de Pareto

x^***:** Ponto pertencente ao conjunto Pareto-Ótimo.

$\bar{F}$**:** Função mono-objetivo na Abordagem Ponderada

2.1 Otimização Multi-objetivo

Um problema de otimização multi-objetivo (do inglês, *Multi-Objective Optimization Problem*, com sigla MOP) pode ser informalmente definido como sendo uma espécie de problema de otimização em que visa-se melhorar duas funções conflitantes de forma simultânea. Alguns exemplos simples de otimização multi-objetivo podem ser encontrados no cotidiano: o problema

de encontrar as ações no mercado financeiro que gerem os maiores dividendos com menores riscos; ou encontrar os carros com motores de maior potência vendidos ao menor preço; ou também encontrar as estratégias que levarão a empresa obterem os maiores lucros com menores custos operacionais. Todos estes problemas possuem duas características em comum: ambos os objetivos devem ser otimizados; e ambos os objetivos não podem ser aumentados em valor sem que seja perdido no outro. Desta forma não haverá apenas uma solução ótima, como em um problema mono-objetivo, e sim um conjunto de soluções ótimas onde cada uma oferecerá uma tendência para um ou outro objetivo.

A otimização multi-objetivo lida em gerar as soluções que compõem o conjunto Pareto-Ótimo, conceito introduzido por (PARETO, 1896). Neste conjunto estão presentes as soluções que apresentam um *trade-off* entre si, isto é, são soluções cujo valor em um dos objetivos não pode ser melhorado sem a piora do valor em quaisquer dos outros objetivos. Estas soluções são chamadas de não-dominadas, e são definidas como aquelas que exercem uma relação de dominância sobre as demais no espaço soluções do problema.

Definição 2.1. *Problema de Otimização Multi-Objetivo (MOP)*

$$\begin{aligned} \max / \min \quad & f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})) & \\ s.a \quad & g_j(\boldsymbol{x}) \geq 0, & j = 1, \ldots, J; \\ & h_k(\boldsymbol{x}) = 0, & k = 1, \ldots, K; \\ & \boldsymbol{x} \in \mathbb{R}^n & \end{aligned} \tag{2.1}$$

Uma solução $\mathbf{x}$ para o problema de otimização é um vetor de variáveis de decisão da forma $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, sujeito a J restrições de desigualdade, representas pela função $g_j(\mathbf{x})$, e K restrições de igualdade, representadas pela função $h_k(\mathbf{x})$ (DEB; KALYANMOY, 2001). O problema é composto por n variáveis de decisão, representadas pelas quantidades de valor de cada uma das componentes de $\mathbf{x} \in \mathbb{R}^n$, e m funções $f(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}^m$, onde $\mathcal{X}$ e $\mathbb{R}^m$ são espaços conhecidos como **espaço das variáveis de decisão factíveis** e **espaço das funções objetivo**, respectivamente (JAIMES; ZAPOTECAS-MARTíNEZ; COELLO, 2011). O conjunto $\mathcal{X}$ é formado pelas soluções que não restrinjam as limitações impostas pelas funções $g_j(\mathbf{x})$ e $h_k(\mathbf{x})$, e o conjunto $\mathbb{R}^m$ é composto pelas imagens de $\mathbf{x} \in \mathcal{X}$.

Em um problema de otimização com um único objetivo, a solução ótima se torna evidente ao utilizar-se os operadores $\geq$ e $\leq$ entre os pares de soluções propostas, dado que estes induzem uma ordem total no conjunto $\mathbb{R}$. Para fins de escolha das soluções para um problema multi-objetivo é necessário uma nova definição de relação, utilizando-se um operador de ordem mais "fraca" para compararmos os vetores de $\mathbb{R}^m$ (JAIMES; ZAPOTECAS-MARTíNEZ; COELLO, 2011). Esta nova relação de dominância está apresentada na Definição 2.2.

Definição 2.2. ***(Relação de Dominância)*** *Sejam duas soluções $\boldsymbol{x}_1$ e $\boldsymbol{x}_2$ a serem comparadas para um problema de otimização multi-objetivo. Considerando sem perda de generalidade apenas*

problemas de minimização, a solução $\boldsymbol{x}_1$ exerce dominância sobre $\boldsymbol{x}_2$, denotado por $\boldsymbol{x}_1 \preceq \boldsymbol{x}_2$, se e somente se:

- *A solução $\boldsymbol{x}_1$ não é pior do que $\boldsymbol{x}_2$ em qualquer dos objetivos do problema, i.e $f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2) \ \ \forall i = 1, 2, \ldots, m$.*
- *A solução $\boldsymbol{x}_1$ é estritamente menor do que $\boldsymbol{x}_2$ em pelo menos um objetivo, i.e $\exists \ i = 1, 2, \ldots, m$ tal que $f_i(\boldsymbol{x}_1) < f_i(\boldsymbol{x}_2)$.*

Em (DEB; KALYANMOY, 2001) é discutido algumas propriedades da relação de dominância:

***Reflexividade*:** do segundo item da Definição 2.2, é mostrado que para uma solução $\mathbf{x}_1$ exercer dominância sobre uma solução $\mathbf{x}_2$, a solução $\mathbf{x}_1$ deve ser estritamente menor do que $\mathbf{x}_2$ em algum dos objetivos. Desta forma, é impossível que uma solução $\mathbf{x}$ consiga exercer dominância sobre ela mesma. Portanto, a relação **não é reflexiva**.

***Simetria*:** Sejam duas soluções $\mathbf{x}_1$ e $\mathbf{x}_2$ tal que $\mathbf{x}_1 \preceq \mathbf{x}_2$. Pela Definição 2.2, a solução $\mathbf{x}_1$ deve ser estritamente menor do que a solução $\mathbf{x}_2$, logo é impossível que $\mathbf{x}_2$ seja menor do que $\mathbf{x}_1$, e portanto $\mathbf{x}_2$ não exerce dominância sobre $\mathbf{x}_1$, ou $\mathbf{x}_2 \npreceq \mathbf{x}_1$. Portanto, a relação também **não é simétrica** .

***Transitividade*:** Considere que $\mathbf{x}_1 \preceq \mathbf{x}_2$ e $\mathbf{x}_2 \preceq x_3$. A solução $\mathbf{x}_1$ é estritamente menor que $\mathbf{x}_2$ que por sua vez deve ser estritamente menor que x_3. Logo, se $\mathbf{x}_1 \preceq \mathbf{x}_2$ e $\mathbf{x}_2 \preceq x_3$ então $\mathbf{x}_1 \preceq x_3$. Portanto, a relação de dominância **é transitiva**.

Estas três propriedades de relações são úteis quando é necessário definir um novo operador com determinada ordem entre os elementos de um mesmo conjunto. Para conseguir empregar alguma ordem em uma relação, (CHANKONG; HAIMES, 2008) mostram que é necessário que esta seja pelo menos transitiva e que a relação de ordem parcial estrita (transitiva, irreflexiva e assimétrica) apresentada no operador de dominância é razoável na maioria dos problemas. Na seção 2.2 serão mostrados métodos para resolução de MOP's inteiramente baseados na construção de conjuntos de soluções através do operador de dominância.

Neste ponto, já é possível sumarizar os principais termos presentes em um MOP: o conjunto de soluções factíveis $\mathcal{X} \subseteq \mathbb{R}^n$; o conjunto de funções objetivo $f(x) : \mathcal{X} \to \mathbb{R}^m$; o espaço das funções objetivo $\mathbb{R}^m$ e, por fim, a relação de ordem $\preceq$.

Como foi visto anteriormente, o principal objetivo da otimização multi-objetivo é obter o conjunto de soluções Pareto-Ótimas. Assim, o conjunto Pareto-Ótimo, os elementos deste conjunto e a Fronteira de Pareto são definidos a seguir.

Definição 2.3. ***(Solução Pareto-Ótima:)*** *$x^* \in F_x$ é uma solução Pareto-Ótima do problema de otimização multi-objetivo se não existe qualquer outra solução $x^* \in F_x$ tal que $f(x) \leq f(x^*)$, ou seja, se x^* não é dominado por nenhum outro ponto factível do problema.*

Definição 2.4. ***Conjunto Pareto-Ótimo:*** *O conjunto Pareto-Ótimo $\mathcal{P}$ é definido como:*

$$\mathcal{P} = \{x \in \mathcal{X} \mid \nexists\, y \in \mathcal{X} : f(y) \preceq f(x)\} \tag{2.2}$$

Definição 2.5. ***Fronteira de Pareto:*** *A fronteira de Pareto $\mathcal{FP}$ é definida como:*

$$\mathcal{FP} = \{f(x) = (f_1(x), \ldots, f_m(x)) \in \mathbb{R}^m | \; x \in \mathcal{P}\} \tag{2.3}$$

A fronteira de Pareto, denotada por $\mathcal{FP}$ é um conjunto composto pelas imagens das soluções contidas no conjunto Pareto-Ótimo $\mathcal{P}$. O conjunto Pareto-Ótimo por sua vez, é composto por todas as soluções não-dominadas, ou seja, aquelas que exercem dominância sobre todas as outras contidas no conjunto factível $\mathcal{X}$. A Figura 2.1 ilustra uma população de soluções factíveis no conjunto $\mathcal{X}$ e a suas respectivas imagens no espaço das funções objetivo.

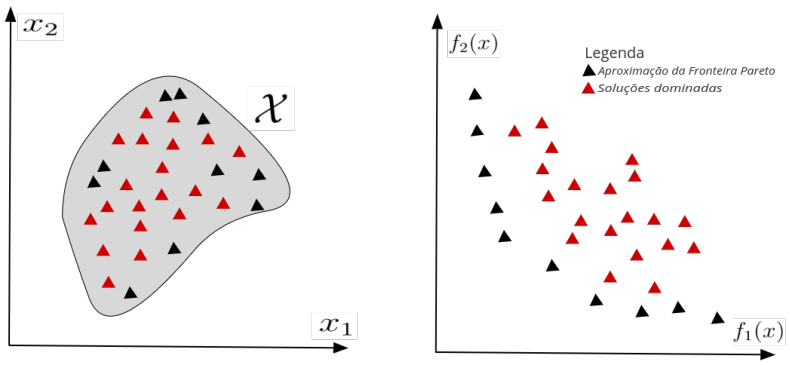

Figura 2.1 – (a) Espaço das soluções factíveis. Em preto estão as soluções que compõem o conjunto Pareto-Ótimo $\mathcal{P}$ (b) Espaço das funções objetivo e a representação da fronteira de Pareto $\mathcal{FP}$. Fonte: Elaborada pelo autor

As soluções pertencentes ao conjunto $\mathcal{FP}$ são ditas matematicamente equivalentes ou incomparáveis, pois o operador de dominância não consegue estabelecer uma ordem entre estas (DEB; KALYANMOY, 2001). Neste caso, as soluções devem ser avaliadas em cada objetivo por um tomador de decisão, a fim de escolher qual é a melhor.

2.1.1 Conjunto de Soluções Eficientes

Nesta seção são apresentados alguns métodos clássicos para a obtenção do conjunto de soluções não dominadas, ou soluções eficientes, de um problema de otimização multi-objetivo. Cada método possui suas formas e especifidades para tratar um problema específico de otimização. Neste contexto, alguns autores categorizaram esses métodos em classes. (COHON, 1985) inicialmente classifica os algoritmos nas classes métodos de geração e métodos de preferência,

que foram posteriormente reclassificados por (MIETTINEN, 2012) para: métodos sem preferência, métodos *a posteriori*, métodos *a priori* e métodos iterativos. Essas classes de métodos são brevemente relatadas e relacionadas à trabalhos externos.

Métodos sem preferência: Nesta abordagem não há preferência de uma solução para outra, logo uma única solução ótima é gerada pelo algoritmo sem a interferência do tomador de decisão em rejeitar ou aceitar esta solução, o que torna improvável que a solução encontrada satisfaça as condições do tomador de decisão (DEB; KALYANMOY, 2001). Um exemplo deste método é a *Multi-objective Proximal Bundle* (MIETTINEN; MÄKELÄ, 1995), que visa mover-se no espaço de busca em uma direção que melhore os valores de todos os objetivos.

Métodos a posteriori: O objetivo destes métodos é prover um conjunto de soluções Pareto-Ótimas para o tomador de decisão, a fim de que este escolha a solução com o objetivo que mais lhe atende. Aqui serão tratados dois métodos bastante conhecidos na literatura: a Abordagem Ponderada e a Abordagem via ϵ-Restrito, onde ambas utilizam do fato de ser possível transformar um problema multi-objetivo em um problema mono-objetivo, porém utilizam estratégias diferentes para este fim.

Métodos a priori: Nesta forma de resolução de problemas, o tomador de decisão deve especificar todas as suas preferências e expectativas a cerca do problema antes do processo de solução (MIETTINEN, 2012). Após a resolução do problema por algum método *a priori* é esperado apenas uma solução ótima baseada nas preferências especificadas anteriormente. Por exemplo, na ordenação lexicográfica, o tomador de decisão deve ordenar as funções objetivo do problema de acordo com a sua importância. Após este processo, a função objetivo mais importante é minimizada, e se uma solução única for encontrada, esta é a solução final para o problema multi-objetivo. Caso a solução não seja única para a primeira função objetivo, então a segunda função mais importante é minimizada e avaliada a sua solução (FISHBURN, 1974). Outro método *a priori* bastante conhecido é o *GOAL Programming*, em que o tomador de decisão define limites, conhecidos como *aspiration levels*, para os valores das soluções encontradas (MIETTINEN, 2012). Algumas extensões do método de *GOAL programming* podem ser encontradas em (IGNIZIO, 1983) e (CHARNES; COOPER, 1977).

Métodos iterativos: Os métodos iterativos exigem pouca ou nenhuma informação *a priori* a cerca do problema, por exemplo, nem mesmo a função objetivo é necessária para sua resolução. Durante o processo de otimização, o tomador de decisão envolve-se no processo a fim de dar alguma informação sobre direções de busca e pontos de referência (DEB; KALYANMOY, 2001). Alguns exemplos conhecidos de métodos iterativos para resolução de problemas são ISWT (*Interactive Surrogate worth trade-off*) e NIMBUS

(*Nondifferentiable interactive multi-objective bundle-based optimization system*) descritos, respectivamente, nos trabalhos de (CHANKONG; HAIMES, 1983) e (MIETTINEN; MÄKELÄ, 1995).

Os métodos clássicos pertencentes a classe *a posteriori*, Abordagem Ponderada e via ϵ-*Restrito*, por se tratarem de métodos bastante difundidos na literatura, serão abordados com mais detalhes, respectivamente, nas Subseções 2.1.1.1 e 2.1.1.2.

2.1.1.1 Abordagem Ponderada

O método Abordagem Ponderada, proposto em (GASS; SAATY, 1955) e estendido, posteriormente, em (ZADEH, 1963), é aplicado na resolução de problemas de otimização com dois ou mais objetivos. Este método consiste em transformar o vetor de funções objetivo $f = (f_1, \ldots, f_m)$ em uma única função objetivo escrita como uma soma ponderada de todas as funções objetivos do problema original. Para isto, é associado a cada objetivo um valor de peso λ_i tal que $i = 1, \ldots, m$. O novo problema é então reformulado conforme a Equação (2.4).

$$\begin{aligned} \max / \min \quad & \bar{F}(x) = \sum_{i=1}^{m} \lambda_i \, f_i(x) & \\ \text{s.a} \quad & g_j(x) \geq 0, & j = 1, \ldots, J; \\ & h_k(x) = 0, & k = 1, \ldots, K; \\ & \lambda_i \geq 0, & \forall \, i = 1, \ldots, m; \\ & x \in \mathbb{R}^n & \end{aligned} \tag{2.4}$$

O valor dos pesos λ_i é definido pelo tomador de decisão do problema, de forma que os valores estejam normalizados, ou seja, $\sum_{i=1}^{\lambda} w_i = 1$. Em (DEB; KALYANMOY, 2001) é citado a importância dos valores das funções objetivos estarem também normalizados, a fim de mantê-los na mesma escala de magnitude. Assim, com valores de f_i e λ_i normalizados, basta resolver o problema de otimização mono objetivo para a função $\bar{F}(x)$.

Dois resultados importantes a cerca da Abordagem Ponderada são os Teoremas 2.1 e 2.2 descritos em (MIETTINEN, 2012).

Teorema 2.1. *A solução para o problema da Equação (2.4) é uma solução Pareto-Ótima.*

Teorema 2.2. *Seja um problema de otimização multi-objetivo convexo. Se x^* é uma solução Pareto-Ótima, então existe um vetor de pesos $\lambda = (\lambda_1, \ldots, \lambda_m)$, tal que $\lambda_i \geq 0, \forall i = 1, \ldots, m$ e $\sum_{i=1}^{m} \lambda_i = 1$, de forma que x^* é uma solução para o problema da soma ponderada da Equação (2.4).*

Os Teoremas 2.1 e 2.2 são complementares no sentido de que se o problema da Equação (2.4) é convexo e possui uma solução x^*, logo existe um vetor de pesos normalizados, e portanto x^* é uma solução Pareto-Ótima para o problema. A demonstração de ambos os teoremas pode ser obtida em (MIETTINEN, 2012).

2.1.1.2 Abordagem via ϵ-Restrito

Este método foi proposto por (YV; LASDON; WISMER, 1971) a fim de, ao contrário da Abordagem Ponderada, trabalhar também com funções não-convexas. O principio deste método é de transformar o problema multi-objetivo em um problema mono-objetivo, similar. Porém, neste método o tomador de decisão escolhe uma das várias funções objetivo para otimizar, e o restante das funções são transformadas em restrições para o problema. Assim, o problema original passa a estar descrito conforme a Equação (2.5).

$$\begin{aligned} \max / \min \quad & f_\alpha(x) \\ \text{s.a} \quad & f_\mu(x) \leq \epsilon_\mu \qquad \mu = 1, \ldots, m \; e \; \mu \neq \alpha; \\ & g_j(x) \geq 0, \qquad\qquad j = 1, \ldots, J; \\ & h_k(x) = 0, \qquad\qquad k = 1, \ldots, K; \\ & x \in \mathbb{R}^n \end{aligned} \tag{2.5}$$

Teorema 2.3. *Uma solução x^* é Pareto-Ótima, se esta também for uma solução única para o problema da Equação (2.5) para qualquer vetor $\epsilon = (\epsilon_1, \ldots, \epsilon_m)^T$*

A prova para o Teorema 2.3 é apresentada em (MIETTINEN, 2012). Os valores de ϵ variam em um intervalo escolhido pelo tomador de decisão do problema. Para cada função objetivo $f_\mu(x)$ é atribuído um valor de ϵ contido no intervalo escolhido. (DEB; KALYANMOY, 2001) sugere que este intervalo esteja entre os valores mínimo e máximo que a função objetivo possa assumir. Portanto, é necessário que o tomador de decisão conheça a forma e a localização da verdadeira fronteira de Pareto do problema.

2.2 Algoritmos Evolucionários

O estudo de Algoritmos Evolucionários difere dos métodos diretos com a introdução de conceitos evolutivos, tais como população, evolução e cruzamento. O processo evolucionário é, de certa forma, inspirado na natureza humana e social. Resumidamente, este processo pode ser sumarizado em: geração e avaliação de uma população de indivíduos que passam por meios de cruzamento, mutação e recombinação para gerar novos indivíduos melhores e mais aptos. Toda a população é, normalmente, inicializada de forma aleatória, assim estes algoritmos não dependem mais de uma boa estimativa inicial. O processo evolucionário gera, iterativamente,

novos indivíduos que convergem para a solução ótima para qualquer problema de entrada. A natureza estocástica destes algoritmos traz soluções parciais para a questão do ótimo local.

Na Subseção 2.2.2 é apresentado o algoritmo MOPSO-CD (do inglês, *Multi-Objective Particle Swarm Optimization with Crowding Distance*), e também são apresentados alguns conceitos utilizados no processo de busca deste algoritmo, como por exemplo: o enxame de partículas e como estas se movimentam no espaço de busca; os parâmetros de voo c_1, c_2 e ω e as suas funções na delimitação do espaço de busca; a troca de informações entre as partículas do enxame por meio dos operadores de p_{Best} e g_{Best}; e a estratégia de atualização do arquivo de soluções aproximadas A.

2.2.1 Configuração de parâmetros em Meta-heurísticas

Nesta sub-seção, apresentaremos resumidamente conceitos à respeito da distinção das possíveis forma conhecidas de configuração de parâmetros de algoritmos evolutivos. SILVA (2012) faz um levantamento sobre as formas de configuração de parâmetros e suas respectivas distinções. Primeiramente, dois grandes grupos são criados a partir do termo configuração de parâmetros:

- Ajuste de Parâmetros: O ajuste de parâmetros é feito de maneira manual e é fixo durante toda a execução do algoritmo. Para cada execução do algoritmo é verificado a eficiência deste frente à diversos problemas e é também realizado o ajuste caso o agente queira.
- Controle de Parâmetros: Esta segunda forma trata o ajuste dos parâmetros de forma dinâmica, ou seja, durante a execução do algoritmo. Este grupo pode ser divido em três outros sub-grupos, que são: determinístico, adaptativo e auto-adaptativo.
  - Determinístico: O controle adaptativo indica uma forma de realizar a mudança dos parâmetros dinamicamente através de uma função determinística, que não utiliza nenhuma informação sobre o processo de busca.
  - Adaptativo: Este caso ocorre quando a mudança dos parâmetros se dá baseada em informações produzidas pelo processo de busca do algoritmo. É importante salientar que para esta forma de configuração os parâmetros permanecem externos ao processo evolutivo do algoritmo, mesmo que utilize as informações da busca.
  - Auto-adaptativo: Neste tipo de configuração os parâmetros são produzidos juntamente aos indivíduos, ou seja, estes são codificados juntamente ao restante do indivíduo e logo também é sujeito aos operadores genéticos (parâmetros de voo, cruzamento, mutação)

A ênfase deste trabalho é de produzir uma forma de controlar os parâmetros utilizando um modelo adaptativo de configuração de parâmetros.

2.2.2 *Multi-Objective Particle Swarm Optimization with Crowding Distance*

As justificativas para a escolha deste algoritmo são principalmente: o algoritmo MOPSO-CD é fácil de ser replicado pelo fato do tipo da busca do algoritmo utilizar apenas duas equações; esta versão é a mais eficiente dentre as versões clássicas do PSO, e ao mesmo tempo também é bastante simplificada quando comparada à outras versões com adaptação de parâmetros.

O algoritmo de otimização por enxame de partículas, ou *particle swarm optimization* (PSO) é uma forma de resolução de problemas de otimização que utiliza o conceito de população e busca multi-dimensional. O algoritmo foi inicialmente desenvolvido por (KENNEDY; EBERHART, 1995), que tentavam simular o comportamento de um enxame de indivíduos em busca por alimento, como é observado por exemplo, no cardume de peixes e no voo dos pássaros. Posteriormente, o PSO foi estendido em (Coello Coello; Lechuga, 2002) para resolução de problemas de otimização multi objetivo, sendo então denominado *Multi-Objective Particle Swarm Optimization* (MOPSO).

Finalmente, outro refinamento descrito em (RAQUEL; NAVAL, 2005) passa a adotar o operador de *Crowding Distance* no processo de aproximação da fronteira de Pareto. Este último algoritmo, conhecido como *Multi-Objective Particle Swarm Optimization with Crowding Distance* (MOPSO-CD) será aquele utilizado neste trabalho na otimização dos seus parâmetros. Todos os algoritmos e equações desta seção são parte do processo evolucionário do algoritmo MOPSO-CD.

O enxame é definido como um conjunto de partículas ou soluções candidatas. Para cada partícula movimentar-se de uma **posição** para outra dentro do espaço de busca, é somado à sua posição atual um valor de **velocidade**. Mais precisamente, cada partícula da forma x_i corresponde a uma solução para o problema. O conjunto contendo toda população de partículas é conhecido como enxame, denotado por $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Cada partícula possui uma posição correspondente no espaço de busca, representada por

$$x_i(t) = [\mathbf{x}_{i,1}(t), \ldots, \mathbf{x}_{i,D}(t)] \tag{2.6}$$

onde D é a dimensão do espaço de busca, t é o contador de iteração do algoritmo e $i = \{1, 2, \ldots, |S|\}$. Cada partícula também possui uma velocidade em uma iteração t do algoritmo, representada por

$$v_i(t) = [v_{i,1}(t), \ldots, v_{i,D}(t)] \tag{2.7}$$

O PSO é baseado em um modelo de simulação social, logo deve existir um mecanismo de troca de informações entre as partículas de forma com que estas compartilhem as suas experiências. Este algoritmo aproxima o valor ótimo da função objetivo por meio da informação da posição de cada partícula do enxame (PARSOPOULOS; VRAHATIS, 2010). Em vista disso, a velocidade

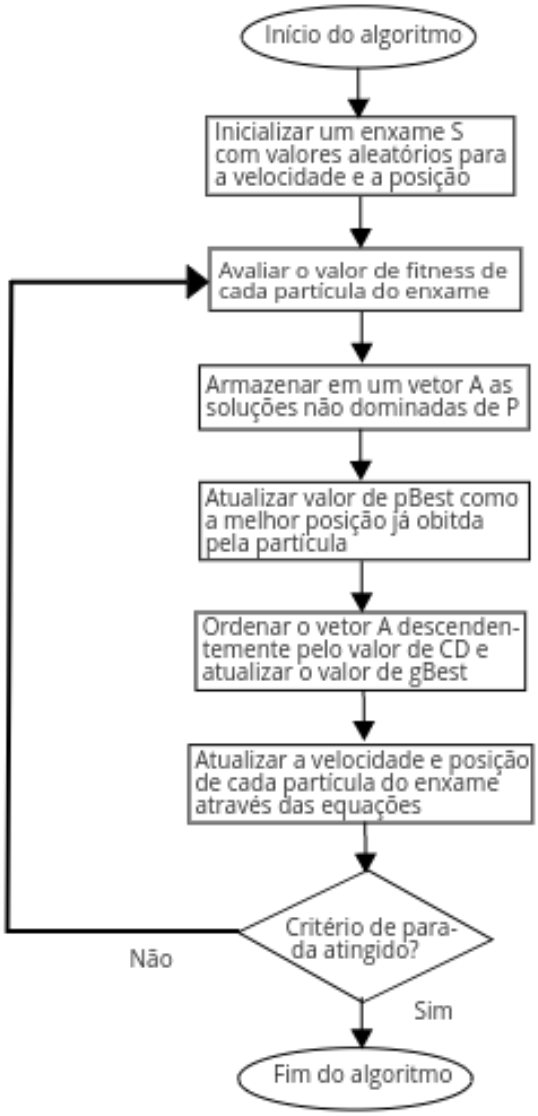

Figura 2.2 – Fluxograma que mostra, de forma geral, as etapas do algoritmo MOPSO-CD na busca do vetor de soluções não-dominadas com maior valor de CD. Fonte: Elaborada pelo autor.

de cada partícula é atualizada baseada em três fatores principais: a velocidade atual da partícula, a melhor posição encontrada por esta partícula e a melhor posição encontrada por todo o enxame de partículas.

A velocidade atual de uma partícula é dada pelo vetor $v(t)$ definido na equação 2.7. A melhor posição encontrada por uma partícula i, denotada por $pBest$, e a melhor posição encontrada por todo o enxame, denotado por $gBest$ são vetores da forma:

$$pBest_i(t) = [pBest_{i,1}(t), \ldots, pBest_{i,D}(t)] \tag{2.8}$$

$$gBest(t) = [gBest_1(t), \ldots, gBest_D(t)] \tag{2.9}$$

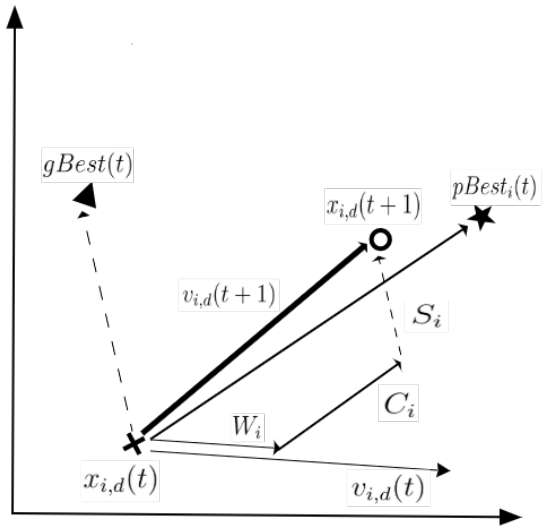

Figura 2.3 – Cálculo da nova posição de uma partícula do enxame na iteração $t+1$. A nova velocidade $v_{i,d}(t+1)$ é a combinação linear das componentes W_i, C_i e S_i. Neste caso, $W_i = \omega v_{i,d}(t)$, $C_i = c_1 r_1(pBest_{i,d}(t) - x_{i,d}(t))$ e $S_i = c_2 r_2(gBest_d(t) - x_{i,d}(t))$. Fonte: Elaborada pelo autor.

A escolha do vetor $pBest_i(t)$ é feita utilizando-se uma estrutura de memória em cada partícula do enxame. Esta memória armazena as posições que a partícula já frequentou. Estas posições são então avaliadas, e o valor de $pBest_i(t)$ armazena a posição que obteve o melhor resultado diante das funções objetivo.

Para a escolha do vetor $gBest(t)$ utilizamos o mesmo modelo proposto por (RAQUEL; NAVAL, 2005), que mantém um vetor A ordenado de forma descendente pelo valor de *crowding distance* das soluções não-dominadas do enxame. Daí, escolhe-se uma posição aleatória encontrada no topo do vetor A (*e.g* 10% das soluções contidas nas primeiras posições de A), e atribui-se a solução contida nesta posição ao vetor $gBest(t)$.

A velocidade de cada partícula é atualizada a cada iteração t do algoritmo utilizando as informações contidas em $v_i(t)$, $pBest_i(t)$ e $gBest(t)$. A partícula move-se no espaço de busca somando-se o valor atual de sua posição com a velocidade encontrada na iteração atual. As equações 2.10 e 2.11 definem como a velocidade e posição no espaço são atualizadas:

$$\mathbf{v}_{i,d}(t+1) = \omega \mathbf{v}_{i,d}(t) + c_1 r_1(\mathbf{pBest}_{i,d}(t) - \mathbf{x}_{i,d}(t)) + c_2 r_2(\mathbf{gBest}_d(t) - \mathbf{x}_{i,d}(t)) \tag{2.10}$$

$$\mathbf{x}_{i,d}(t+1) = \mathbf{x}_{i,d}(t) + \mathbf{v}_{i,d}(t+1) \tag{2.11}$$

Os próximos parágrafos, no que se referem às definições e nomenclatura dos parâmetros c_1, c_2 e ω, suas deficiências, soluções e outros conceitos relativos à estes parâmetros, utilizam de conceitos presentes no importante trabalho realizado em (PARSOPOULOS; VRAHATIS, 2010).

Os parâmetros r_1 e r_2 são variáveis aleatórias uniformemente distribuídas no intervalo $[0, 1]$. Os parâmetros ω, c_1 e c_2 são chamados de parâmetros de voo. Especialmente, os parâmetros c_1, c_2 e ω são chamados de **parâmetro cognitivo**, **parâmetro social** e **peso de inércia**, respectivamente. Por fim, a velocidade da partícula na iteração seguinte é a combinação linear das componentes de inércia, cognitiva e social, como mostradas nas equações 2.10 e 2.11 e ilustradas pela figura 2.3.

A magnitude da busca depende fortemente dos valores escolhidos para c_1 e c_2. Se é necessário realizar uma busca global, então deve-se escolher valores altos para c_1 e c_2, o que faz com que as partículas busquem em regiões mais distantes do espaço. Da mesma forma, se a situação é para uma busca local, então devemos escolher valores baixos para c_1 e c_2 para que as partículas busquem em uma região mais limitada do espaço. Outras análises sobre c_1 e c_2 são em relação à direção em que a partícula deve seguir (p_{best} ou g_{best}). Escolhendo valores para c_1 e c_2 tal que $c_1 < c_2$, o algoritmo será enviesado para seguir a direção imposta por p_{best}, enquanto que se $c_1 > c_2$ então o algoritmo segue a direção de g_{best}.

Um problema muito comum em relação ao PSO é o de explosão do enxame, ou do inglês, *swarm explosion*. A explosão se dá ao fato de um aumento incontrolável no valor da velocidade das partículas do enxame. Uma técnica para solucionar este problema bastante utilizada em versões iniciais do PSO é de simplesmente limitar a velocidade de todas as partículas a partir de uma constante pré-definida. Porém, depois de muito testada, esta técnica mostrou uma problemática para as partículas buscarem soluções promissoras por uma limitação imposta. Esta técnica foi então rapidamente substituída por um novo parâmetro chamado de peso de inércia proposto por (Shi; Eberhart, 1998). O peso de inércia denotado por ω é um fator associado à direção atual da partícula, assim como c_1 e c_2 estão associados ao p_{best} e g_{best}.

Apesar da grande melhoria trazida pelo parâmetro de peso de inércia em relação à convergência das soluções e explosão do enxame, o algoritmo ainda possui fortes tendências a permanecer preso em um ótimo local, especialmente em problemas complexos. Neste caso, problemas complexos podem ser entendidos como problemas de otimização multi-modais com funções não-convexas. Nestes tipos de funções, o enxame encontra um novo problema: a perda da diversidade das soluções. Com a perda de diversidade do enxame, as partículas perdem a capacidade de buscarem em outras regiões do espaço. Desta forma, as partículas convergem sempre para um mesmo espaço e prendem-se em um ótimo local.

A deficiência intitulada de perda de diversidade pode ser atribuída ao fato de que no PSO as partículas possuem um esquema de troca de informações sobre as melhores posições encontradas. Com este esquema todas as partículas assumem uma nova posição em uma mesma região das melhores posições encontradas.

Clerc (2010) demonstra o conceito de vizinhança no PSO, que trouxe uma resolução parcial para a perda de diversidade do enxame. Resumidamente, cada partícula assume um conjunto de partículas intituladas vizinhas, e a cada iteração do algoritmo esta partícula comunica

a melhor posição encontrada por esta (p_{best}) somente para os seus vizinhos. Perceba que a informação trocada por uma partícula limita-se a seus vizinhos na iteração atual, porém em longo prazo, esta informação é espalhada para todo o enxame através de outras vizinhanças. O tamanho e a topologia da vinhança podem ser arbitrárias desde que a quantidade de vizinhos não seja tão grande ao ponto de comprometer o algoritmo ao problema da perda de diversidade.

O pseudo-código 2.1 adaptado de (RAQUEL; NAVAL, 2005) mostra o processo evolutivo na convergência do enxame para o conjunto Pareto-Ótimo. Primeiramente, um enxame de tamanho M é inicializado com valores aleatórios, velocidades nulas e as posições de p_{best} e g_{best} são atualizadas conforme são encontradas posições x_i melhores (linhas 2 a 8). Enquanto o máximo de iterações não for atingido, o arquivo A contendo as soluções não-dominadas é ordenado de forma decrescente em relação ao valor de CD de cada indivíduo (linhas 11 a 13). Para cada partícula i sua posição e velocidade são atualizadas iterativamente (linhas 16 e 17), simulando o movimento no espaço de busca. Caso a nova posição x_i encontrada por uma partícula seja melhor do que todas as posições das partículas contidas no conjunto A, então x_i é adicionada em A e todas as partículas dominadas por esta são removidas (linhas 18 a 22). O valor de $gBest$ é atualizado como sendo a posição da primeira partícula contida no conjunto A (linha 15). O valor de $pBest_i$ é atualizado caso a posição encontrada pela partícula i seja melhor do que todas

as já encontradas por esta (linhas 23 a 25).

Algoritmo 2.1: *MOPSO-CD.*

Entrada: $M, maxIter$

Saída: A (aproximação da fronteira de Pareto)

```
1  início
2      para i = 1 até M faça
3          x_i ← aleatorio_d;
4          v_i ← 0;
5          pBest_i ← x_i;
6          gBest ← pBest_max;
7          P_i ← x_i;
8      fim
9      iter ← 0;
10     A ← nao_dominado(P);
11     enquanto (iter ≤ maxIter) faça
12         D ← crowding_distance(A);
13         ordenar_decrescente(A, D);
14         para i = 1 até M faça
15             gBest ← A_0;
16             v_i = ωv_i + c_1 r_1(pBest_i − x_i) + c_2 r_2(gBest − x_i);
17             x_i = x_i + v_{i+1};
18             P ← avalia(x_i);
19             se (x_i ≺ A) então
20                 adiciona(x_i, A);
21                 remove_dominados(A, x_i);
22             fim
23             se (x_i ≺ pBest_i) então
24                 pBest_i ← x_i;
25             fim
26         fim
27         iter ← iter + 1;
28     fim
29 fim
```

2.3 Medidas de Desempenho

Antes de sumarizar e definir as métricas que utilizaremos neste trabalho, vale ressaltar que, em geral, o resultado gerado por um algoritmo que resolve um MOP é apenas uma aproximação da fronteira de Pareto real do problema. Segue as definições de (Zitzler et al., 2003) para conjunto

aproximado e métrica de desempenho:

Definição 2.6. *Seja $A \subseteq \mathbb{R}^m$ um conjunto de soluções. Dizemos que A é uma aproximação, se qualquer solução de A não dominar ou não ser igual a nenhuma outra solução de A. O conjunto de todas as aproximações será denotado por Z.*

A partir da aproximação gerada por um algoritmo, podemos mapear um conjunto aproximado para um valor numérico real. Este valor é conhecido como métrica de desempenho e é utilizado para comparar a qualidade de um algoritmo ao resolver um MOP.

Definição 2.7. *Dados h conjuntos aproximados $A_i \in Z \mid i = 1, \ldots, h$, então uma métrica de performance h-ária I é dada por $I : Z^h \to \mathbb{R}$. Para cada conjunto A_i é atribuído um valor real $I(A_i)$.*

O conjunto de métricas existentes na literatura podem ser sumarizados em duas categorias como mostrado em (Riquelme; Von Lücken; Baran, 2015). A primeira categoria define a quantidade de conjuntos aproximados A_i que são levados em consideração para calcular o valor da métrica. No caso da métrica **unária**, esta recebe apenas um conjunto aproximado como parâmetro, ou seja $I(A) : Z \to \mathbb{R}$, e, no caso da métrica **binária**, dois conjuntos aproximados são necessários para o cálculo da métrica, ou seja $I(A, B) : Z^2 \to \mathbb{R}$. A segunda categoria considera o aspecto de avaliação dos valores dos conjuntos aproximados de Z:

- **Métricas de Cardinalidade**: Medem a quantidade de soluções contidas em uma aproximação A.
- **Métricas de Acurácia**: Medem a distância das soluções contidas em A para as soluções da fronteira de Pareto real $\mathcal{PF}$ conhecida. Para problemas onde não se conhece a fronteira real do problema, então é considerado um conjunto de referência R como alternativa.
- **Métricas de Diversidade**: Representa o quão bem distribuídas e espalhadas estão as soluções de A. A boa distribuição refere-se à distância relativa das soluções de A, enquanto que o espalhamento refere-se aos valores cobertos pela solução A em relação à fronteira de Pareto.

2.3.1 *Hypervolume*

A métrica de *Hypervolume* (HV), também conhecida como métrica-S, é uma métrica unária que computa o tamanho do espaço de objetivos coberto pelo conjunto aproximado encontrado (Zitzler; Thiele, 1999). É uma métrica completa, pois considera todos os aspectos de avaliação dos conjuntos aproximados: a cardinalidade, a acurácia e a diversidade. Segundo (Riquelme; Von Lücken; Baran, 2015), o hiper-volume é a métrica mais citada em trabalhos envolvendo otimização multi-objetivo por meio de algoritmos evolucionários. Isto se deve ao fato desta

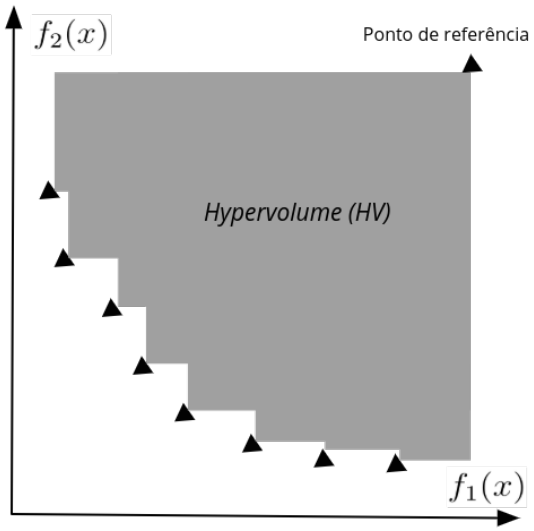

Figura 2.4 – Ilustração do cálculo da métrica *Hypervolume*. São geradas todas as áreas ao longo de todos os objetivos dos pontos do conjunto aproximado A até um ponto de referência escolhido. Fonte: Elaborada pelo autor.

métrica possuir propriedades únicas em relação à convergência do algoritmo em questão, como explica (ZITZLER; BROCKHOFF; THIELE, 2007): sempre que o valor de HV de um conjunto aproximado A for maior do que um conjunto B, então A domina completamente todas as soluções de B. Com isso, o conjunto com maior valor de HV contém todas as soluções Pareto-Ótimas encontradas, e portanto se torna a melhor aproximação encontrada.

Como ilustrado na figura 2.4, o valor HV é calculado da forma: para cada solução contida em A é traçado um hiperplano perpendicular aos eixos f_i , $i = 1, \ldots, n$ até o limite imposto, ao longo de todos os objetivos, pelo ponto de referência escolhido. O valor de HV é a soma das áreas de todos os hiperplanos produzidos.

2.3.2 *Inverse Generational Distance*

A métrica unária *Inverse Generational Distance* (IGD) diz respeito à distância média dos pontos de um conjunto aproximado A e da fronteira de Pareto real do problema. Esta métrica é um bom indicador de convergência quando se conhece a fronteira de Pareto real do problema. Foi proposta por (COELLO; CORTéS, 2005) como sendo a função inversa da métrica *Generational Distance* mostrada em (LAMONT; VELDHUIZEN, 1999). Primeiro, toma-se cada ponto da fronteira de Pareto real PF^* do problema como referência e calcula-se a distância euclidiana deste para o ponto mais próximo de A. Mais precisamente, pode ser definida pela fórmula:

$$IGD(PF^*, A) = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2.12}$$

onde n é a quantidade de soluções contidas em PF^*, d_i é a distância euclidiana do ponto

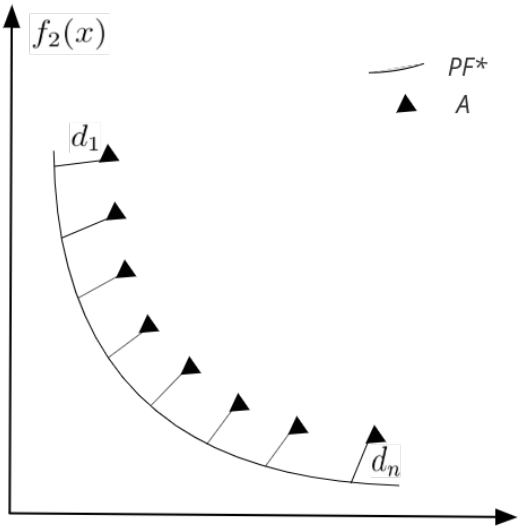

Figura 2.5 – Ilustração do processo de cálculo das n distâncias de $PF*$ até os pontos de A. Fonte: Elaborada pelo autor.

$i \in PF^*$ até o ponto mais próximo de A. A figura 2.5 ilustra estas distâncias.

Esta métrica também é bastante utilizada na literatura, pois: possui baixo custo computacional para computá-la; pode medir tanto a acurácia quanto a diversidade de A, caso uma quantidade suficiente de soluções de $PF*$ sejam conhecidas (Riquelme; Von Lücken; Baran, 2015). A maior desvantagem desta métrica é quando não se conhece a fronteira de Pareto real do problema, impossibilitando de usá-la. Porém, para medições com funções de teste com solução ótima conhecida, como é o caso deste trabalho, esta métrica se torna bastante viável.

2.3.3 *Generational Distance*

Esta métrica é literalmente a função inversa da função IGD. Foi proposta pelo mesmo autor em (LAMONT; VELDHUIZEN, 1999). Possui baixo custo computacional para computar o seu valor e é definida como sendo:

$$IGD(A, PF*) = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2.13}$$

Aqui, os pontos de referência fixados para o cálculo são os pontos do conjunto aproximado A. Daí, para cada ponto de A, calcula-se a distância euclidiana para os pontos da fronteira de Pareto $PF*$ conhecida.

2.4 Revisão da Literatura

Esta seção faz um breve resumo dos trabalhos relacionados com a proposta desta monografia.

2.4.1 Adaptação dos parâmetros no MOPSO-CD

O conceito de peso de inércia foi inicialmente proposto por (Shi; Eberhart, 1998). Neste trabalho os autores executam testes empíricos que mostram que: os valores de ω deveriam se sustentar dentro do intervalo $[0.5, 1.2]$, a fim de balancear a quantidade de buscas locais e globais do algoritmo, e consequentemente reduzindo o tempo de convergência; e que valores altos para ω facilitam a busca global, enquanto que valores baixo de ω são melhores em buscas locais. Xin, Chen e Hai (2009) propõem uma forma de adaptação de ω, utilizando um conjunto de funções que o tornam linearmente decrescente entre as iterações do algoritmo. Outras formas de decrescimento semelhantes ao decrescimento linear do parâmetro ω foram apresentadas em: (Guimin Chen et al., 2006) com um decrescimento exponencial, e em (Gao; An; Liu, 2008) com decrescimento logarítmico.

Os primeiro estudos empíricos em relação aos parâmetros c_1 e c_2 vieram através de (KENNEDY, 1998), mostrando que, para uma boa convergência do algoritmo, deve ser satisfeito $c_1 + c_2 \leq 4$. Várias configurações de testes para c_1 e c_2 são mostradas em (CARLISLE; DOZIER, 2001). Em (Clerc; Kennedy, 2002) são mostradas várias conclusões a respeito da estabilidade e convergência do PSO, onde conclui-se principalmente que: valores altos para ω e c_1 e valores baixos para c_2 aumentam a diversidade do enxame; por outro lado, valores baixos para ω e c_1 e valores altos para c_2 melhoram a convergência do algoritmo. A proposta de (HAN; LU; QIAO, 2017) ajusta os parâmetros ω, c_1 e c_2 a partir do espaçamento das partículas do enxame para obter melhores critérios de busca para resolver o problema dos sistemas de distribuição de água. No trabalho de (Hu; Yen, 2015) um sistema de PCCS (*Parallel Cell Coordinate System*) é introduzido para ajustar os parâmetros ω, c_1 e c_2, mostrando resultados satisfatórios diante de vários outros algoritmos evolucionários. Os resultados obtidos em (Ma et al., 2016) utilizando uma função senoidal de ajuste dinâmico dos parâmetros c_1 e c_2 foram bem superiores à versão original do algoritmo MOPSO/CD na maioria das funções de teste.

Em (MARTÍNEZ; COELLO, 2011) é proposto uma abordagem de decomposição em vários conjuntos de soluções para a atualização da posição de cada partícula do enxame. Outro método utilizando a decomposição pode ser encontrado em (LIN et al., 2015), onde cada partícula é atribuída para resolver um sub-problema decomposto do problema original. A proposta descrita em (ZHANG et al., 2018) utiliza um torneio entre pares de soluções a cada geração para a atualização dos valores de $pBest$ e $gBest$.

3 Adaptação dos Parâmetros

Neste capítulo será apresentado uma proposta de um operador com estratégia de memória para fazer a adaptação dos parâmetros na meta-heurísticas MOPSO-CD. Para tanto, na Seção 2.4 é apresentada uma breve revisão da literatura sobre as funções adaptativas para os parâmetros do MOPSO-CD (Subseção 2.4.1), e como os parâmetros deste algoritmo impactam na busca local, além dos principais conceitos sobre estratégias de memórias em meta-heurísticas através de estruturas de dados árvores e/ou tabelas hash, e como estas estruturas podem impactar no processo de busca do algoritmo (Subseção 3.1). Na Seção 3.2 é apresentada a proposta deste trabalho, que consiste em um operador que utiliza uma estrutura de memória para fazer a adaptação dos parâmetros principais das meta-heurísticas.

3.1 Estratégias de Memória em Meta-heurísticas

Su et al. (2020) apresentam uma falha dos algoritmos genéticos que é a re-avaliação de soluções. Por este motivo, os autores incorporam em um algoritmo genético comum uma memória algorítmica por meio de uma árvore BSP (ou, *Binary Space Partitioning*). A ideia é evitar a re-avaliação de soluções a fim de diminuir o tempo de execução. Em (YANG; WU, 2020), é utilizada a mesma estrutura de dados para realizar o processo de armazenamento de soluções já avaliadas. O autor aqui busca resolver o problema de *thresholding* de imagens em escala de cinza. Neste artigo, é inserido uma árvore BSP no algoritmo PSO, a fim de não re-avaliar soluções, visto que a função de avaliação aqui é muito custosa.

Uma das funções de um mecanismo de memória em algoritmos evolutivos é de evitar a reavaliação de soluções já avaliadas. Em casos de problemas de otimização que possuem funções objetivo muito complexas, este mecanismo reduz drasticamente este tempo de reavaliação. Outro ganho importante obtido através de um mecanismo de memória é a manutenção da diversidade das soluções, pois elimina as soluções duplicadas e redundantes. Vale ressaltar que com o mecanismo de memória, a reavaliação é resumida à uma busca em uma estrutura de dados (CARRANO; MOREIRA; TAKAHASHI, 2011). Além dos benefícios de prevenir uma reavaliação ou melhorar a diversidade, o mecanismo de memória pode ser um artefato a fim de produzir soluções melhores, por meio de adaptação dos parâmetros considerando informações contidas na memória. Este mecanismo de memória pode ser qualquer estrutura de dados, normalmente são utilizadas árvores ou tabelas *hash* para guardar estas soluções.

Carrano, Moreira e Takahashi (2011) utilizam a estrutura de dados tabela *hash* como memória algorítmica interna ao NSGA-II. Em (COELHO, 2016) é feita a adaptação da função hash definida em (CARRANO; MOREIRA; TAKAHASHI, 2011) e proposto um operador de memória, utilizando a tabela hash como estrutura de memória, na meta-heurística VNS (ou do

inglês, Variable Neighborhood Search), a fim evitar re-avaliações de soluções duplicadas e para criar uma vizinhança adaptativa para a busca local.

Tabelas hash são estruturas que armazenam dados cujos índices são transformações de uma chave. Cada posição pode ser composta por um ou vários valores, que no escopo deste artigo será uma solução ou um conjunto de soluções. Para armazenar uma solução somente, a tabela resume-se à um vetor, e para um conjunto de soluções esta passa a receber em cada posição uma lista encadeada.

Uma das medidas que podem ser tomadas para tratar colisões em tabelas hash é de utilizar uma lista encadeada em cada posição da tabela. Portanto, uma lista é formada e são adicionados nesta os valores que possuem mesmo índice produzido pela função de hash. O custo para buscar um valor em uma tabela hash pode variar de $\mathcal{O}(1)$ (um único valor em cada índice) até $\mathcal{O}(n)$ (todos os valores na mesma posição). A função mais comum e que vai ser utilizado neste trabalho é o hash de Pearson para cadeias textuais mostrado em (PEARSON, 1990).

Algoritmo 3.1: *pearson_hashing*.

Entrada: P

Saída: H (Valor de hash de um indivíduo P)

início

  $h[0] \leftarrow 0$;

  $n_{loops} \leftarrow \dfrac{n.n_{bits}}{b_{ind}}$;

  para $i = 1$ até n_{loops} **faça**

    $h[i] \leftarrow T[bin2dec(dec2bin(h[i-1]) \oplus key[(i-1).b_{ind}+1, \ldots, i.b_{ind}])]$;

  fim

  $retorna \;\; h[n_{loops}]$;

fim

Neste algoritmo: n é o número de variáveis de decisão do problema, n_{bits} é o número de bits por variável; T é o vetor que contém os valores provindos da permutação 2^{bind} ordenados aleatoriamente, onde $bind$ é um valor escolhido para o tamanho da tabela; $T[p]$ é um valor retirado desta tabela no índice p; $bin2dec$ é uma função que converte um número binário em decimal; $dec2bin$ transforma um número decimal em binário; $\oplus$ é o operador de ou exclusivo; o valor de $key[i]$ corresponde aos bits que correspondentes à posição i da palavra binária key.

Utilizamos esta função de *hash* para mapear os valores reais das funções objetivo com qualquer número de casas decimais para um valor inteiro que corresponde ao índice na tabela. Para cada algarismo do valor real de cada função objetivo avaliada, são retirados qualquer espécie de caracteres especiais (e.g pontos e letras) e os zeros, obtendo um número inteiro. A partir deste número, o índice na tabela para a solução avaliada é o *hash* de Pearson deste valor inteiro.

3.1.1 Discussão

No caso do MOPSO-CD, grande parte das propostas dos trabalhos relacionados dizem respeito à variação dos parâmetros de voo durante a execução do algoritmo. Outra forma de adaptação neste algoritmo é encontrar novas formas de determinar os valores de g_{Best} e p_{Best}. Em todos os trabalhos o algoritmo proposto obteve melhores resultados do que a versão original do algoritmo. Então, é notório que o ajuste automático dos parâmetros do algoritmo é mais eficiente do que estes com valores fixos.

3.2 *Parameter Adaptation based on Meta-Heuristics Memory*

Nesta seção será apresentado o operador de Adaptação de Parâmetros Baseada em Memória de Meta-Heurísticas (PBMM) proposto.

3.2.1 Operador PBMM

A forma de adaptação dos parâmetros neste trabalho consiste das etapas: para cada geração, guardar as soluções já avaliadas no operador de memória PBMM; verificar se a solução que está sendo avaliada já existe na memória, e se já existir, incrementa-se o valor do contador c; verificar se o limite de soluções iguais foi atingido; se o limite foi atingido, então alteram-se os valores dos parâmetros c_1, c_2 e ω, e limpa a memória. O objetivo neste tipo de adaptação é transformar os valores dos parâmetros citados, de forma que mudanças sejam feitas no processo de busca sem que comprometa o mesmo. A busca executada pelo algoritmo MOPSO-CD pode ser ampliada ou reduzida de acordo com os valores destes parâmetros. Ao mesmo tempo que, embora seja possível alterar estes parâmetros indefinidamente para que possamos vasculhar novos locais no espaço, existem problemas relacionados à este tipo de crescimento, como mostrado na seção 2.2.2. Por isso, também desejamos que existam restrições de crescimento dos parâmetros.

A primeira etapa do operador é armazenar as soluções já avaliadas em uma estrutura de dados. Para isto, consideramos o uso de uma tabela *hash* de tamanho fixo determinado igual a 256, com chave de tamanho 8-bit. A função de hash utilizada é a de Pearson detalhada em 3.1.

A segunda etapa é de percorrer a tabela criada a cada avaliação de função objetivo no algoritmo. Ao percorrer, buscamos por uma solução idêntica à que está sendo avaliada pelo algoritmo no momento, e se caso ocorra uma repetição deste valor, ou colisão, incrementamos o valor do contador n, que especifica o número de avaliações idênticas. O valor de contador n será então comparado à um limite L, estipulado pelo usuário do programa, que corresponde ao número máximo de colisões que o algoritmo venha a obter antes de alterar os parâmetros. Caso o valor de n tenha ultrapassado o valor de L, então o algoritmo realiza a adaptação dos parâmetros.

Na terceira etapa, o objetivo é de alterar os parâmetros relevantes (*i.e* c_1, c_2 e ω) com base no resultado obtido na segunda etapa. Cada um dos parâmetros é submetido à uma pertubação

mostrada na 3.1. A partir de um intervalo real $[\alpha_1; \alpha_2]$, definido pelo usuário do programa, valores aleatórios uniformes k_1, k_2 e k_2 são produzidos, que aqui podemos chamar de pesos aleatórios. Estes pesos são multiplicados pelo valor de contador c, como mostra a equação 3.1, e o resultado é atribuído ao valor dos parâmetros.

$$\begin{aligned} c_1 &= c_1 + (k_1 \times c) \\ c_2 &= c_2 - (k_2 \times c) \\ \omega &= \omega + (k_3 \times c) \end{aligned} \tag{3.1}$$

onde:

- k_1, k_2 e k_3 são variáveis aleatórias de uma distribuição uniforme escolhidas entre o intervalo definido pelo usuário (e.g $[0.05, 0.07]$).
- c é o contador de quantas vezes o número de colisões máximo foi ultrapassado do limite L

A escolha das equações se sustenta na revisão da literatura à respeito do algoritmo PSO, mostrado na seção 2.4.1. Para uma melhor convergência e diversidade das soluções obtidas, alguns testes empíricos mostram que valores altos de c_2 melhoram a busca global e por consequência a busca de novas soluções, facilitando a convergência em detrimento da diversidade. Enquanto que valores altos para ω e c_1 fazem com que as soluções se prendam mais no mesmo espaço de busca, trazendo uma maior diversidade em detrimento da convergência do algoritmo.

A partir dos testes empíricos, o que desejamos para a nova versão do algoritmo é que os parâmetros alterem de acordo com o que é estável na literatura. A escolha de incrementar os parâmetros c_1 e ω partem da ideia de que, no início da busca, valores baixos para estes são os desejados. Enquanto que ao final da busca, sejam desejados valores mais altos para c_1 e ω. Por outro lado, o parâmetro c_2 é decrementado a fim de que valores altos de c_2 no início da busca e valores baixos no fim busca tornem-o mais eficiente. Testes empíricos mostrados em (KENNEDY, 1998), (CARLISLE; DOZIER, 2001), (Hu; Yen, 2015), (Ma et al., 2016) e (HAN; LU; QIAO, 2017)

Outro fato a se atentar, como também mostram os testes das revisões bibliográficas, é que os valores dos parâmetros não devem crescer ou decrescer indefinidamente sem que siga alguma regra ou limitante. Os limitantes utilizados neste trabalho são:

- $c_1 + c_2 \leq 4.5$
- $c_1 \leq 3.0$
- $c_2 \leq 1.0$
- $\omega \leq 1.2$

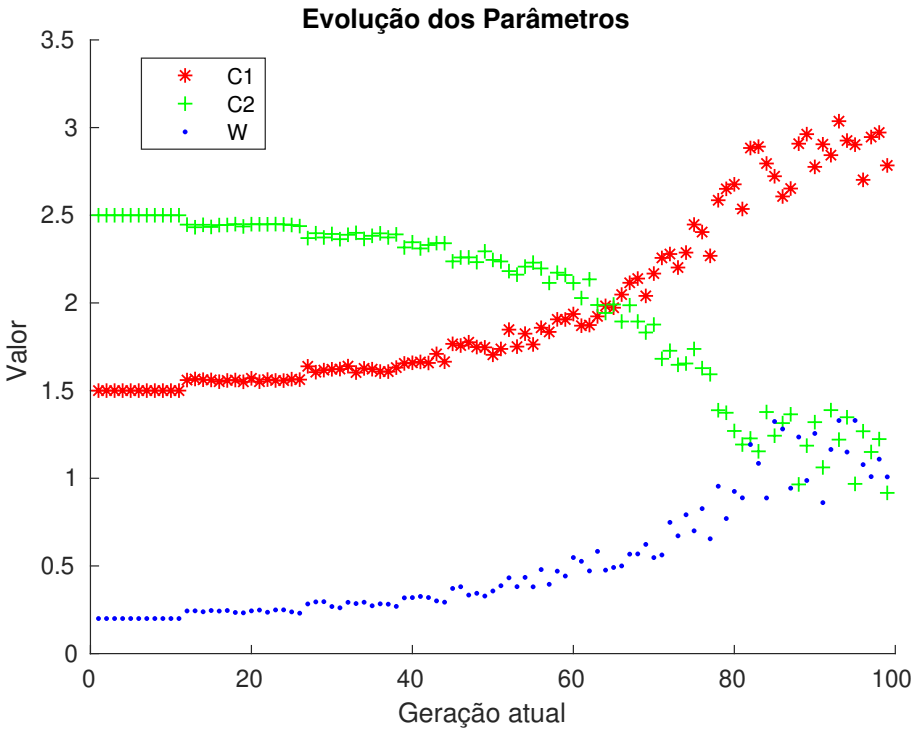

Figura 3.1 – Ilustração da evolução dos parâmetros no problema DTLZ1 com número máximo de avaliações igual a 30000 e população de tamanho 300

Ou seja, sempre que uma das violações acima forem rompidas, o algoritmo possui uma maneira de voltar o contador alguns passos, ou seja, realizar $c - b$, onde b é um valor estipulado pelo usuário. Desta forma, é possível manter os valores dos parâmetros dentro do intervalo considerado estável.

A proposta de algoritmo aqui é aliar estes conceitos empíricos à memória algorítmica. O algoritmo de enxame de partículas, como mostrado na seção 2.2.2, tem uma grande probabilidade de gerar as mesmas soluções no processo de otimização. Nesta etapa, a produção de soluções idênticas pode indicar que as partículas do enxame estão associadas à um mesmo sub-espaço de busca.

A pertubação nos algoritmos por meio da memória de soluções, diferente de outros trabalhos da revisão, só realiza a adaptação quando entende-se que o algoritmo pode estar preso em um mesmo local no espaço de busca, pois está produzindo soluções idênticas. Mesmo em problemas reais, quando realizamos os testes na seção 4.3, o algoritmo teve um bom desempenho quanto à gerar soluções iguais e realizar a adaptação. A figura 3.1 mostra a evolução dos parâmetros durante a otimização em uma instância.

O maior espaçamento entre os valores dos parâmetros identifica que mais soluções idênticas estão sendo criadas, pois o valor do contador c é maior. Por isso, no início do algoritmo há um pequeno espaço entre as soluções geradas, enquanto que no final existe um espaço maior. Esta variação durante todas as gerações é devido ao peso k_i ser um valor aleatório produzido

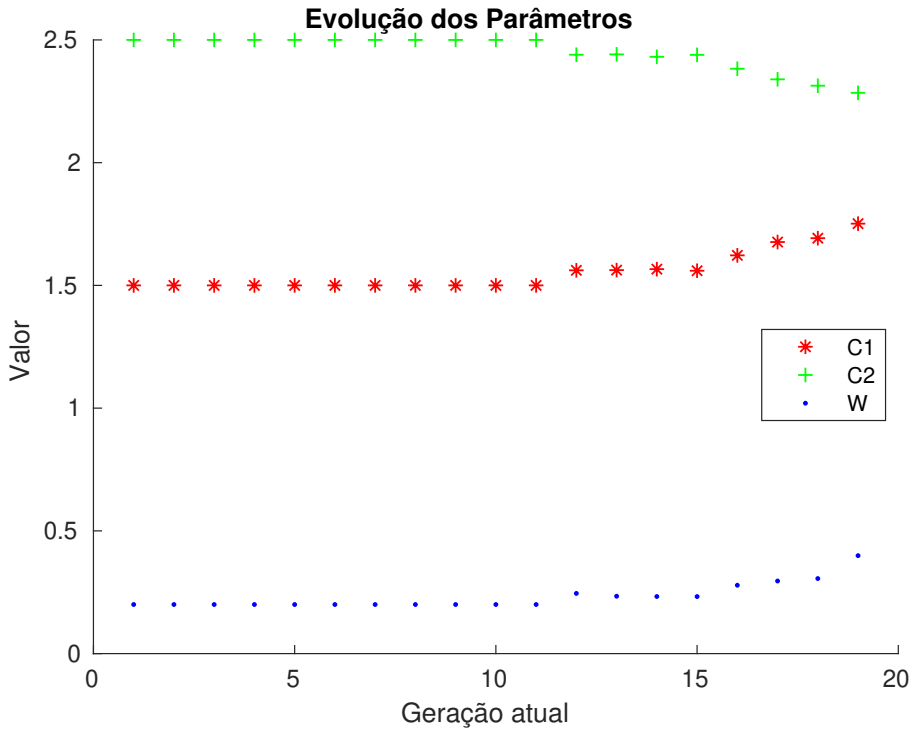

Figura 3.2 – Ilustração da evolução dos parâmetros no problema ZDT1, número máximo de avaliações igual a 10000 e população de tamanho 500

dentro de um intervalo especificado.

Veja que, com o valores mostrados na figura, visualizamos o comportamento esperado para que o algoritmo produza soluções com métricas de convergência e diversidade boas segundo os resultados empíricos da literatura citados anteriormente. Os valores dos parâmetros também se limitam dentro do intervalo considerado como estável contra problemas conhecidos, como mostrado na seção 2.2.2 e 2.4.1.

No exemplo da figura 3.2 visualizamos o comportamento contrário ao que esperamos para uma boa convergência e diversidade das soluções, mas que pode ocorrer em cenários específicos. Neste caso, escolhemos uma instância que produz somente 20 gerações e por isso não tivemos muitas soluções idênticas sendo produzidas.

O pseudo-código 3.2 sumariza os pontos mostrados até aqui sobre o algoritmo desenvolvido. A nova versão do algoritmo MOPSO-CD com o operador PBMM foi denominada memMOPSOCD. Perceba que, em relação à versão original mostrada em 2.1, a mudança se dá nos passos de avaliação de soluções. Na nova versão, para cada solução, inserimos a solução na memória algorítmica $PBMM$, na linha 20. Ao fim da avaliação, realizamos uma chamada ao procedimento 3.3 que tratará de: percorrer a memória, verificar se a solução produzida já existe na memória, atualizar o valor do contador c conforme definido e realizar a adaptação dos

parâmetros.

Algoritmo 3.2: *memMOPSOCD*.

Entrada: $M, maxIter, limiteColisoes$

Saída: A (aproximação da fronteira de Pareto)

início
  $PBMM \leftarrow construtor\_tabelaHash$;
  para $i = 1$ até M **faça**
    $x_i \leftarrow aleatorio_d$;
    $v_i \leftarrow 0$;
    $pBest_i \leftarrow x_i$;
    $gBest \leftarrow pBest_{max}$;
    $P_i \leftarrow x_i$;
  fim
  $c \leftarrow 0$;
  $iter \leftarrow 0$;
  $A \leftarrow nao\_dominado(P)$;
  enquanto $(iter \leq maxIter)$ **faça**
    $D \leftarrow crowding\_distance(A)$;
    $ordenar\_decrescente(A, D)$;
    para $i = 1$ até M **faça**
      $gBest \leftarrow A_0$;
      $v_i = \omega v_i + c_1 r_1 (pBest_i - x_i) + c_2 r_2 (gBest - x_i)$;
      $x_i = x_i + v_{i+1}$;
      $k \leftarrow pearson\_hash(P_t)$;
      $PBMM_k \leftarrow P_t$;
      $P \leftarrow avalia(x_i)$;
      se $(x_i \prec A)$ **então**
        $adiciona(x_i, A)$;
        $remove\_dominados(A, x_i)$;
      fim
      se $(x_i \prec pBest_i)$ **então**
        $pBest_i \leftarrow x_i$;
      fim
      $alteraParametros(PBMM, c, x_i)$;
    fim
    $iter \leftarrow iter + 1$;
  fim
fim

A rotina $alteraParametros$ realiza a alteração dos parâmetros com base nos argumentos de entrada: a tabela com as soluções, a solução atual avaliada e o contador. No inicio do procedimento, inicializamos os valores iniciais e finais que os parâmetros c_1, c_2 e ω atingirão. Na linha 12 avaliamos a solução atual e comparamos com as outras da memória. Na linha 15, incrementamos o valor do contador conforme ultrapasse um limite de colisões. Logo: n representa o número de soluções repetidas produzidas, c quantas vezes essa mesma solução foi gerada, L o limite desejado de soluções e k_1, k_2 e k_3 são número aleatórios escolhidos sob um intervalo definido pelo usuário. Caso os parâmetros ultrapassem as restrições impostas, o código da linha 18 trata de decrementar o valor do contador por um valor de correção b, que também é definido pelo usuário.

Algoritmo 3.3: *alteraParametros*.

Entrada: $PBMM, c, x$

Saída: c_1, c_2, ω

```
1  início
2      c_1 ← 1.5;
3      c_2 ← 2.5;
4      ω ← 0.2;
5      b ← 2;
6      L ← 15;
7      n ← 0;
8      lista_enc ← pearson_hash(x) ;
9      para k = 1 até |lista_enc| faça
10         se x = lista_enc_k então
11             n ← n + 1;
12         fim
13     fim
14     se n > L então
15         c ← c + 1;
16     fim
17     se (c_1 > 3.0) ou(c_2 < 1.0) ou(ω > 1.2) ou(c_1 + c_2 ≥ 4.5) então
18         c ← c − b;
19     fim
20     c_1 ← c_1 + k_1 × c;
21     c_2 ← c_2 − k_2 × c;
22     ω ← ω + k_3 × c;
23 fim
```

4 Resultados

Todos os algoritmos foram implementados utilizando a linguagem de programação MATLAB. Todas as simulações, avaliações de funções, cálculo das métricas de desempenho e implementação das versões originais dos algoritmos foram feitas a partir de uma extensão do framework PlatEMO (ou, *Evolutionary Multi-bjective Optimization Platform* proposto em (Tian et al., 2017), e que pode ser encontrado em <https://github.com/BIMK/PlatEMO>. Este framework oferece algumas funcionalidades prontas para uso, o que torna mais fácil a produção de tantos testes em uma grande quantidade de funções objetivo. As mudanças realizadas podem ser vistas em <https://github.com/pedrolopes9-7/PlatEMO/pull/2/files>.

Os teste foram realizados em um computador com as seguintes configurações: i7-3610QM CPU @ 2.30GHz, 8GB de RAM @ 1.6GHz e sistema operacional Ubuntu 18.04 LTS.

O restante deste capítulo apresenta a seguinte organização: na Seção 4.1 são apresentados os problemas testes utilizados para avaliar a metodologia proposta, na Seção 2.3 é definido o processo de implementação das métricas de desempenho, na Seção 4.3 são apresentados os resultados finais do trabalho.

4.1 Descrição dos Problemas Teste

Os problemas testes de otimização multiobjetivo usados para avaliar o desempenho da metodologia proposta em comparação com a versão clássica dos algoritmos NSGA-II e MOPSO-CD, são descritos nesta seção.

Os problemas conhecidos como IMOP são definidos por (Tian et al., 2019). Os problemas IMOP [1-3] possuem duas funções objetivo apenas, enquanto que os problemas IMOP [4-8] possuem três. Os parâmetros a_1, a_2 e a_3 enviesam as posições das soluções na fronteira de Pareto, afetando também a diversidade do conjunto aproximado encontrado. Em todos os testes os valores de a_1, a_2 e a_3 foram fixados em $0.05, 0.05$ e 10, respectivamente, de forma equivalente aos testes feitos no artigo original.

- IMOP1:

$$f_1(x) = g + cos^8(\frac{\pi}{2}y_1)$$
$$f_2(x) = g + sin^8(\frac{\pi}{2}y_1)$$

- IMOP2:

$$f_1(x) = g + cos^{0.5}(\frac{\pi}{2}y_1)$$
$$f_2(x) = g + sin^{0.5}(\frac{\pi}{2}y_1)$$

- IMOP3:

$$f_1(x) = g + 1 + \frac{1}{5}cos(10\pi y_1) - y_1$$
$$f_2(x) = g + y_1$$

- IMOP4:

$$f_1(x) = (1+g)y_1$$
$$f_2(x) = (1+g)(y_1 + \frac{1}{10}sin(10\pi y_1))$$
$$f_3(x) = (1+g)(1-y_1)$$

- IMOP5:

$$h_1 = 0.4cos(\frac{\pi}{4}\lceil 8y_2 \rceil) + 0.1y_3cos(16\pi y_2)$$
$$h_2 = 0.4sin(\frac{\pi}{4}\lceil 8y_2 \rceil) + 0.1y_3sin(16\pi y_2)$$
$$f_1(x) = g + h_1$$
$$f_2(x) = g + h_2$$
$$f_3(x) = g + 0.5 - h_1 - h_2$$

- IMOP6:

$$r = max\{0, min\{sin^2(3\pi y_2), sin^2(3\pi y_3)\} - 0.05\}$$
$$f_1(x) = (1+g)y_2 + \lceil r \rceil$$
$$f_2(x) = (1+g)y_3 + \lceil r \rceil$$
$$f_3(x) = (0.5+g)(2 - y_2 - y_3) + \lceil r \rceil$$

- IMOP7:

$$h_1 = (1+g)cos(\frac{\pi}{2}y_2)cos(\frac{\pi}{2}y_3)$$
$$h_2 = (1+g)cos(\frac{\pi}{2}y_2)sin(\frac{\pi}{2}y_3)$$
$$h_3 = (1+g)sin(\frac{\pi}{2}y_2)$$
$$r = min\{min\{|h_1 - h_2|, |h_2 - h_3|\}, |h_3 - h_1|\}$$
$$f_1(x) = h_1 + 10max\{0, r - 0.1\}$$
$$f_2(x) = h_2 + 10max\{0, r - 0.1\}$$
$$f_3(x) = h_3 + 10max\{0, r - 0.1\}$$

- IMOP8:

$$f_1(x) = y_2$$
$$f_2(x) = y_3$$
$$f_3(x) = (1+g)\left[3 - \sum_{i=2}^{3} \frac{y_i(1 + sin(19\pi y_1))}{1+g}\right]$$

onde:

$$x = (x_1, \ldots, x_K, x_{K+1}, \ldots, x_{K+L}) \in [0,1]^{K+L}$$
$$y_1 = \left(\frac{1}{K} \sum_{i=1}^{K} x_i \right)^{a_1}$$
$$y_2 = \left(\frac{1}{\lceil K/2 \rceil} \sum_{i=1}^{\lceil K/2 \rceil} x_i \right)^{a_2}$$
$$y_3 = \left(\frac{1}{\lceil K/2 \rceil} \sum_{i=\lceil K/2 \rceil + 1}^{K} x_i \right)^{a_3}$$
$$g = \sum_{i=K+1}^{K+L} (x_i - 0.5)^2$$

Também utilizaremos a plataforma de testes DTLZ proposta por (DEB et al., 2005) para realizarmos os testes de comparação dos algoritmos.

- DTLZ1:

$$min\ f_1(x) = \frac{1}{2} x_1 x_2 \ldots x_{M-1}(1 + g(x_M))$$
$$min\ f_2(x) = \frac{1}{2} x_1 x_2 \ldots (1 - x_{M-1})(1 + g(x_M))$$
$$\vdots$$
$$min\ f_{M-1}(x) = \frac{1}{2} x_1 (1 - x_2)(1 + g(x_M))$$
$$min\ f_M(x) = \frac{1}{2} (1 - x_1)(1 + g(x_M))$$
$$s.a\ 0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n$$
$$g(x_M) = 100 \left[|x_M| + \sum_{x_i \in X_M} (x_i - 0.5)^2 - cos(20\pi(x_i - 0.5)) \right]$$

- DTLZ2:

$$min\ f_1(x) = (1 + g(x_M)) cos(x_1 \frac{\pi}{2}) \ldots cos(x_{M-2} \frac{\pi}{2}) cos(x_{M-1} \frac{\pi}{2}),$$
$$min\ f_2(x) = (1 + g(x_M)) cos(x_1 \frac{\pi}{2}) \ldots cos(x_{M-2} \frac{\pi}{2}) sin(x_{M-1} \frac{\pi}{2}),$$
$$min\ f_3(x) = (1 + g(x_M)) cos(x_1 \frac{\pi}{2}) \ldots sin(x_{M-2} \frac{\pi}{2}),$$
$$\vdots$$
$$min f_M(x) = (1 + g(x_M)) sin(x_1 \frac{\pi}{2})$$
$$g(x_M) = \sum_{x_i \in X_M} (x_i - 0.5)^2,$$
$$0 \leq x_i \leq 1,\ i = 1, 2, \ldots, n.$$

- DTLZ3:

$$
\begin{aligned}
&min\ f_1(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})\ldots cos(x_{M-2}\frac{\pi}{2})cos(x_{M-1}\frac{\pi}{2}),\\
&min\ f_2(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})\ldots cos(x_{M-2}\frac{\pi}{2})sin(x_{M-1}\frac{\pi}{2}),\\
&min\ f_3(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})\ldots sin(x_{M-2}\frac{\pi}{2}),\\
&\vdots\\
&min\ f_M(x) = (1 + g(x_M))sin(x_1\frac{\pi}{2})\\
&g(x_M) = 100\left[|x_M| + \sum_{x_i \in X_M}(x_i - 0.5)^2 - cos(20\pi(x_i - 0.5))\right],\\
&0 \leq x_i \leq 1,\ i = 1, 2, \ldots, n.
\end{aligned}
$$

- DTLZ4:

$$
\begin{aligned}
&min\ f_1(x) = (1 + g(x_M))cos(x_1^\alpha\frac{\pi}{2})\ldots cos(x_{M-2}^\alpha\frac{\pi}{2})cos(x_{M-1}^\alpha\frac{\pi}{2}),\\
&min\ f_2(x) = (1 + g(x_M))cos(x_1^\alpha\frac{\pi}{2})\ldots cos(x_{M-2}^\alpha\frac{\pi}{2})sin(x_{M-1}^\alpha\frac{\pi}{2}),\\
&min\ f_3(x) = (1 + g(x_M))cos(x_1^\alpha\frac{\pi}{2})\ldots sin(x_{M-2}^\alpha\frac{\pi}{2}),\\
&\vdots\\
&minf_M(x) = (1 + g(x_M))sin(x_1^\alpha\frac{\pi}{2})\\
&g(x_M) = \sum_{x_i \in X_M}(x_i - 0.5)^2,\\
&0 \leq x_i \leq 1,\ i = 1, 2, \ldots, n,\\
&\alpha \text{ é uma constante tal que } \alpha = 100.
\end{aligned}
$$

- DTLZ5:

$$
\begin{aligned}
&min\ f_1(x) = (1 + g(x_M))cos(\theta_1\frac{\pi}{2})\ldots cos(\theta_{M-2}\frac{\pi}{2})cos(\theta_{M-1}\frac{\pi}{2}),\\
&min\ f_2(x) = (1 + g(x_M))cos(\theta_1\frac{\pi}{2})\ldots cos(\theta_{M-2}\frac{\pi}{2})sin(\theta_{M-1}\frac{\pi}{2}),\\
&min\ f_3(x) = (1 + g(x_M))cos(\theta_1\frac{\pi}{2})\ldots sin(\theta_{M-2}\frac{\pi}{2}),\\
&\vdots\\
&min\ f_M(x) = (1 + g(x_M))sin(\theta_1\frac{\pi}{2})\\
&\theta_i = \frac{\pi}{4(1 + g(X_M))}(1 + 2g(X_M)x_i),\ \ i = 2, 3, \ldots, (M - 1),\\
&g(x_M) = \sum_{x_j \in X_M}(x_i - 0.5)^2,\\
&0 \leq x_j \leq 1,\ j = 1, 2, \ldots, n.
\end{aligned}
$$

- DTLZ6:

$$
\begin{aligned}
&min\ f_1(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})\dots cos(x_{M-2}\frac{\pi}{2})cos(x_{M-1}\frac{\pi}{2}),\\
&min\ f_2(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})\dots cos(x_{M-2}\frac{\pi}{2})sin(x_{M-1}\frac{\pi}{2}),\\
&min\ f_3(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})\dots sin(x_{M-2}\frac{\pi}{2}),\\
&\vdots\\
&min\ f_M(x) = (1 + g(x_M))sin(x_1\frac{\pi}{2})\\
&g(x_M) = \sum_{x_i \in X_M} x_i^{0.1}\\
&0 \leq x_i \leq 1,\ i = 1, 2, \dots, n.
\end{aligned}
$$

- DTLZ7:

$$
\begin{aligned}
&min\ f_1(x) = x_1,\\
&min\ f_2(x) = x_2,\\
&min\ f_{M-1}(x_{M-1}) = x_{M-1},\\
&\vdots\\
&min\ f_M(x) = (1 + g(x_M))h(f_1, f_2, \dots, f_{M-1}, g),\\
&g(x_M) = 1 + \frac{9}{x_M}\sum_{x_i \in x_M} x_i,\\
&h(f_1, f_2, \dots, f_{M-1}, g) = M - \sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1 + sin(3\pi f_i))\right],\\
&s.a\ 0 \leq x_i \leq 1,\ i = 1, 2, \dots, n.
\end{aligned}
$$

- DTLZ8:

$$
\begin{aligned}
&min\ f_j(x) = \frac{1}{\frac{n}{M}}, \sum_{i=(j-1)\frac{n}{M}} x_i^{j\frac{n}{M}}, j = 1, ..., M\\
&g_j(x) = f_M(x) + 4f_j(x) - 1 \geq 0, 1, ..., (M-1)\\
&g_M(x) = 2f_M(x) + min_{i,j=1}^{M-1}[f_i(x) + f_j(x)] - 1 \geq 0\\
&0 \geq x_i \geq 1
\end{aligned}
$$

- DTLZ9:

$$
\begin{aligned}
&min\ f_j(x) = \frac{1}{\frac{n}{M}}, \sum_{i=(j-1)\frac{n}{M}} x_i^{j\frac{n}{M}}, j = 1, ..., M\\
&g_j(x) = f_M^2(x) + 4f_j^2(x) - 1 \geq 0, 1, ..., (M-1)\\
&0 \geq x_i \geq 1
\end{aligned}
$$

4.2 Implementação das Métricas de Desempenho

Para o cálculo da métrica de *hypervolume*, especificamente, é necessário que o avaliador utilize de um ponto de referência. Este ponto é um objeto N-dimensional, tal que N é o número de objetivos. Os valores do ponto referência escolhido, são fixados em 1 em todas as componentes do vetor de referência.

Na métrica de IGD e GD cada ponto da fronteira aproximada é comparada com a fronteira de Pareto do problema. A fronteira de Pareto do problema é conhecida em todas as funções de teste utilizadas no trabalho.

4.3 Resultados Numéricos

Para fins de teste e validez da nova versão do algoritmo, utilizaremos algumas simulações para calcular os valores das métricas de IGD, GD, HV e tempo de execução em 50 execuções do algoritmo. Cada simulação foi construída de forma à atender critérios que tornam o algoritmo eficiente ou ineficiente, destacando cenários que produzem soluções melhores que a versão original e cenários que podem ser melhorados para que o mesmo ocorra.

Os parâmetros escolhidos para o algoritmo original MOPSO-CD foram os mesmos escolhidos pelo autor em seu artigo original, i.e $c_1 = 1.5$, $c_2 = 2.5$ e $\omega = 0.2$.

Em cada simulação de teste mostraremos: as informações relativas ao gráfico de evolução dos valores médios dos parâmetros nas 50 execuções do algoritmo; uma tabela contendo o valor médio (valor fora dos parênteses) de HV, GD, IGD e tempo de execução e do desvio padrão destas métricas (valor dentro dos parênteses); e o gráfico de barras de erro.

4.3.1 Simulação 1: Comparação do operador de memória com um limite de colisões igual a 15 para avaliar a adaptação dos parâmetros e a produção de métricas melhores que o original

- Parâmetros fixos no MOPSO-CD: $c_1 = 1.5$, $c_2 = 2.5$ e $\omega = 0.2$.
- Tamanho da População: 300.
- Critério de Parada: número máximo de 30000 avaliações de função objetivo.
- Número de variáveis de decisão: 10.
- Número de funções objetivo: variável de acordo com o problema teste. Para os problemas IMOP[1-4] são utilizadas duas funções objetivo por problema, enquanto que nos problemas. IMOP[5-8] e DTLZ[1-9] são utilizadas três funções objetivo por problema.
- Limite de colisões L: 15.

- Intervalo escolhido para k_i: $[0.05; 0.07]$.
- Valor de correção b: 2.

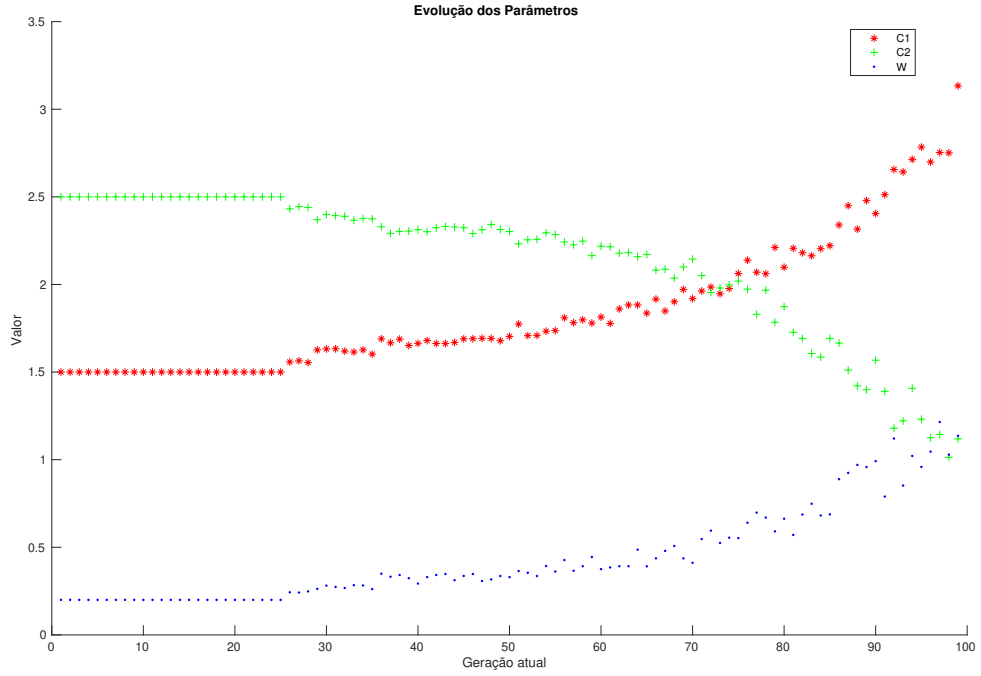

Figura 4.1 – Ilustração da evolução média dos parâmetros na simulação 1 em 50 execuções do algoritmo

Esta simulação é a melhor obtida, e retrata bem o objetivo do trabalho que é tirar proveito da produção de soluções idênticas no processo evolutivo do PSO. Nesta simulação reparamos que os valores dos parâmetros estão sendo produzidos como esperados para vencer o algoritmo original. Isto se deve ao fato de que nesta simulação utilizamos um valor limite L para as colisões igual a 15. Utilizando este valor para L, o algoritmo tratará mais colisões na tabela e os parâmetros sofrerão as pertubações esperadas.

A figura 4.1 mostra a evolução dos parâmetros do algoritmo. As figuras 4.2, 4.3 e 4.4 refletem os mesmos valores das tabelas 4.1, 4.2 e 4.3, onde cada ponto é a média e a barra de erro é o desvio padrão. Nas métricas de GD e IGD houve uma significante melhora em algumas funções objetivo, e em outras uma melhora não tanto significativa. Isto pode ser dado à natureza de cada função objetivo do problema. Em funções mais complexas ambas as versões do algoritmo não convergem de maneira efetiva para a fronteira de Pareto real do problema, fazendo com que as soluções estejam muito espaçadas e assim não produzindo soluções idênticas.

Problem	MOPSOCD	memMOPSOCD
DTLZ1	8.1365e+0 (1.34e+0)	6.5277e+0 (7.93e-1)
DTLZ2	6.2982e-2 (7.39e-3)	2.4912e-3 (2.35e-4)
DTLZ3	5.5929e+0 (9.11e+0)	4.2683e+1 (4.23e+0)
DTLZ4	1.0408e-1 (1.70e-2)	1.4202e-2 (5.82e-3)
DTLZ5	9.2099e-2 (1.07e-2)	3.3549e-4 (7.17e-5)
DTLZ6	1.7813e-3 (1.91e-1)	1.5972e-6 (2.49e-7)
DTLZ7	1.0233e+0 (4.72e-1)	1.5031e-3 (1.36e-4)
DTLZ8	3.1308e-2 (2.50e-2)	2.1365e-1 (2.32e-1)
DTLZ9	5.6322e-1 (3.34e-1)	5.6386e-1 (4.80e-1)
IMOP1	1.6921e-1 (1.21e-1)	1.8870e-4 (3.23e-4)
IMOP2	9.7428e-2 (8.02e-2)	2.4579e-4 (5.79e-4)
IMOP3	1.9285e-1 (1.31e-1)	6.5442e-4 (1.16e-3)
IMOP4	1.3677e-1 (3.61e-2)	2.4921e-3 (6.69e-3)
IMOP5	5.4073e-2 (4.68e-2)	1.5997e-3 (2.33e-3)
IMOP6	1.1063e-1 (5.45e-2)	3.8099e-3 (2.59e-3)
IMOP7	6.9806e-2 (1.96e-2)	1.1810e-2 (8.26e-3)
IMOP8	9.9845e-2 (2.36e-2)	2.2025e-2 (4.79e-3)

Tabela 4.1 – Métrica de GD para a Simulação 1. Veja Gráfico 4.2

Problem	MOPSOCD	memMOPSOCD
DTLZ1	3.2488e+1 (4.50e+0)	2.1301e+1 (5.50e+0)
DTLZ2	3.1463e-1 (2.44e-2)	5.5431e-2 (3.38e-3)
DTLZ3	2.0455e+2 (1.22e+1)	1.4734e+2 (2.81e+1)
DTLZ4	6.5821e-1 (8.57e-2)	2.2244e-1 (3.61e-2)
DTLZ5	2.2738e-1 (3.26e-2)	4.8230e-3 (8.09e-4)
DTLZ6	8.7862e-1 (4.40e-1)	8.3596e-3 (3.39e-3)
DTLZ7	2.5365e+0 (9.35e-1)	4.6089e-2 (2.52e-3)
DTLZ8	1.8384e-1 (1.26e-1)	4.6089e+0 (2.52e+0)
DTLZ9	1.4525e+0 (6.03e-1)	1.2906e+0 (8.45e-1)
IMOP1	6.4984e-1 (1.08e-1)	1.2545e-1 (9.37e-3)
IMOP2	3.9621e-1 (1.02e-1)	1.3404e-1 (1.06e-2)
IMOP3	5.4238e-1 (1.06e-1)	1.4239e-1 (2.12e-2)
IMOP4	6.5465e-1 (5.44e-2)	1.8099e-1 (5.03e-2)
IMOP5	5.6255e-1 (8.71e-2)	2.4550e-1 (2.22e-2)
IMOP6	5.7928e-1 (7.02e-2)	2.2530e-1 (3.44e-2)
IMOP7	6.8578e-1 (5.86e-2)	3.7393e-1 (5.80e-2)
IMOP8	6.3554e-1 (4.38e-2)	3.5556e-1 (5.98e-2)

Tabela 4.2 – Métrica IGD para a Simulação 1. Veja Gráfico 4.3

Problem	MOPSOCD	memMOPSOCD
DTLZ1	4.9258e-3 (9.83e-3)	9.4330e-3 (1.18e-2)
DTLZ2	3.5313e-1 (3.44e-2)	3.8256e-1 (1.82e-2)
DTLZ3	1.9937e-2 (7.67e-2)	2.5464e-1 (6.78e-2)
DTLZ4	5.0336e-2 (3.12e-2)	1.2119e-1 (3.63e-2)
DTLZ5	6.6370e-2 (7.02e-2)	7.6481e-1 (3.57e-2)
DTLZ6	1.4904e-2 (1.75e-1)	2.7147e-1 (7.48e-2)
DTLZ7	8.5945e-2 (1.07e-1)	4.1927e-1 (2.91e-2)
DTLZ8	2.2217e-1 (2.05e-1)	2.1119e-1 (3.12e-2)
DTLZ9	1.7479e-1 (1.76e-1)	2.3354e-1 (2.09e-1)
IMOP1	1.5623e-1 (8.66e-2)	2.6697e-1 (2.21e-2)
IMOP2	2.4865e-1 (1.13e-1)	2.2599e-1 (1.58e-2)
IMOP3	3.3086e-1 (1.51e-1)	1.9323e-1 (1.44e-2)
IMOP4	1.5427e-2 (8.27e-2)	3.3920e-1 (6.38e-2)
IMOP5	3.1115e-1 (3.04e-1)	2.3029e-1 (1.72e-1)
IMOP6	1.8587e-1 (7.11e-2)	1.7239e-1 (4.55e-2)
IMOP7	4.5753e-2 (2.12e-2)	1.0805e-1 (4.73e-2)
IMOP8	1.0872e-1 (4.28e-2)	1.1389e-1 (4.56e-2)

Tabela 4.3 – Métrica HV para Simulação 1. Veja Gráfico 4.4

Problem	MOPSOCD	memMOPSOCD
DTLZ1	5.4133e-1 (1.74e-2)	1.6173e+1 (4.32e-1)
DTLZ2	5.7084e-1 (2.33e-2)	1.8375e+1 (6.50e-1)
DTLZ3	5.5580e-1 (2.23e-2)	1.7244e+1 (5.84e-1)
DTLZ4	5.5166e-1 (2.50e-2)	1.8712e+1 (6.37e-1)
DTLZ5	5.5211e-1 (1.60e-2)	1.9148e+1 (4.02e-1)
DTLZ6	5.7582e-1 (2.86e-2)	1.8236e+1 (7.30e-1)
DTLZ7	5.7865e-1 (1.92e-2)	1.6688e+1 (3.72e-1)
DTLZ8	6.5834e-1 (3.99e-2)	1.8955e+1 (5.20e-1)
DTLZ9	2.1255e-1 (9.45e-3)	4.6807e+0 (1.74e-1)
IMOP1	5.4131e-1 (1.54e-2)	1.6678e+1 (4.06e-1)
IMOP2	5.3523e-1 (1.74e-2)	1.6080e+1 (4.19e-1)
IMOP3	5.2989e-1 (1.28e-2)	1.6683e+1 (5.41e-1)
IMOP4	7.1890e-1 (1.89e-2)	2.6380e+1 (8.32e-1)
IMOP5	5.4413e-1 (2.04e-2)	1.8968e+1 (8.40e-1)
IMOP6	5.5812e-1 (2.13e-2)	1.8606e+1 (6.28e-1)
IMOP7	5.6040e-1 (4.20e-2)	1.8741e+1 (5.42e-1)
IMOP8	5.5696e-1 (2.62e-2)	1.8564e+1 (7.12e-1)

Tabela 4.4 – Tempo de execução em segundos para a simulação 1. Veja Gráfico 4.5

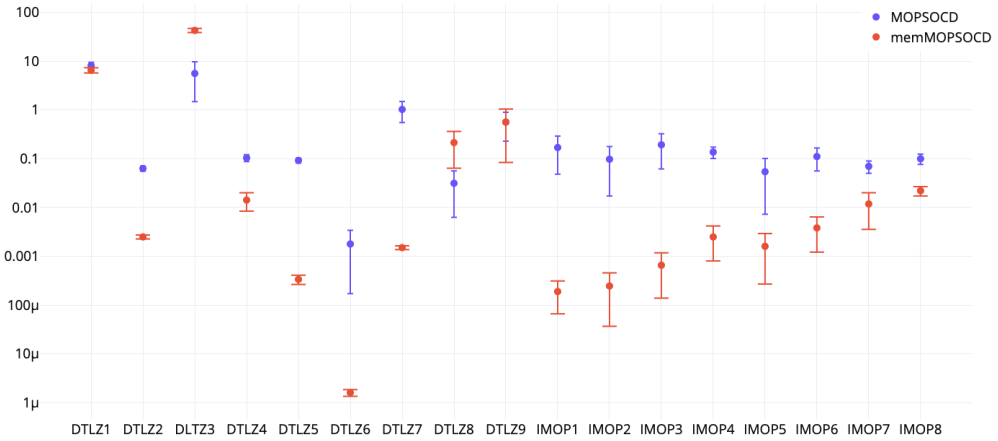

Figura 4.2 – Gráfico para a tabela 4.1, contendo a o valor médio da métrica GD e o erro em formato de barra e em escala logarítmica

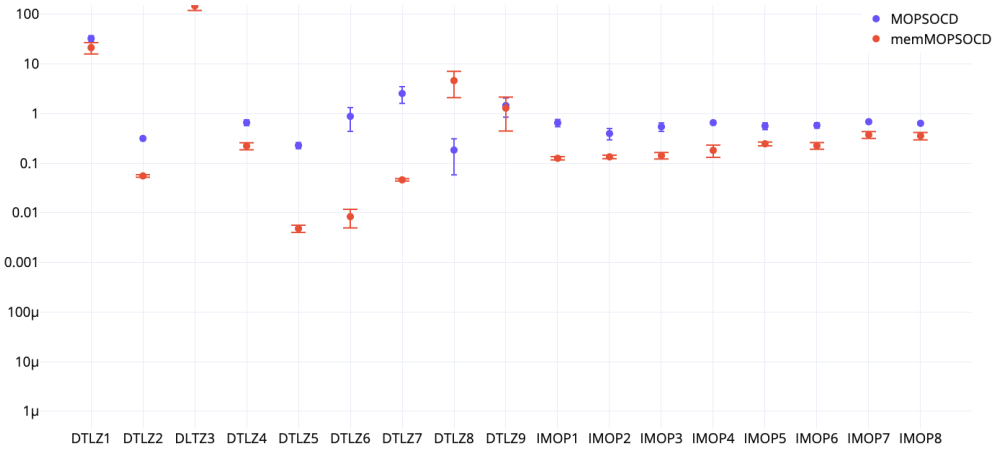

Figura 4.3 – Gráfico para a tabela 4.2, contendo a o valor médio da métrica IGD e o erro em formato de barra e em escala logarítmica

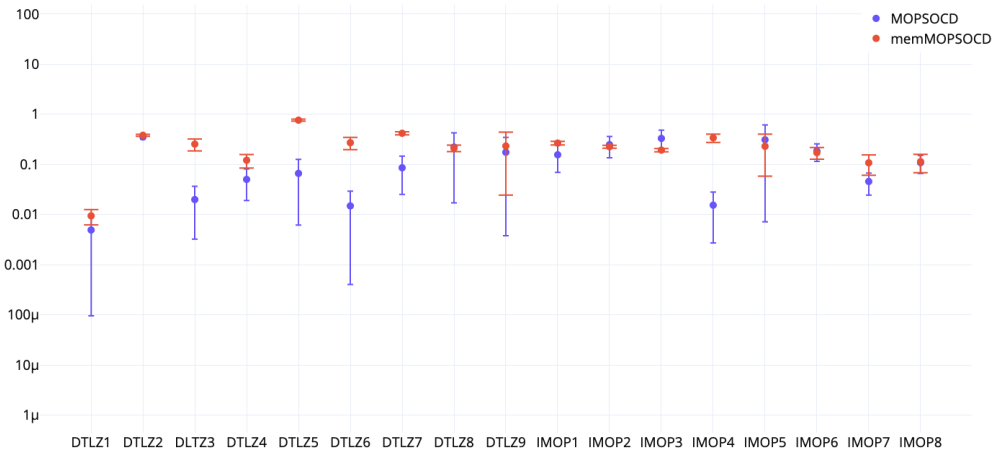

Figura 4.4 – Gráfico para a tabela 4.3, contendo a o valor médio da métrica HV e o erro em formato de barra e em escala logarítmica

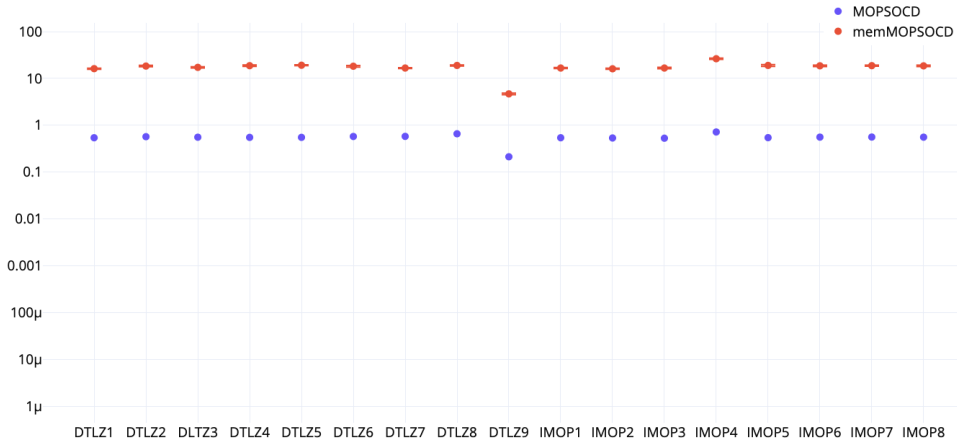

Figura 4.5 – Gráfico para a tabela 4.4, contendo a o valor médio da métrica tempo de execução e o erro em formato de barra e em escala logarítmica

4.3.2 Simulação 2: Comparação do operador de memória com um limite de colisões igual a 100 para avaliar o cenário com nenhuma adaptação de parâmetros e nenhuma mudança nas métricas de desempenho

- Parâmetros fixos no MOPSO-CD: $c_1 = 1.5$, $c_2 = 2.5$ e $\omega = 0.2$.
- Tamanho da População: 300.
- Critério de Parada: número máximo de 30000 avaliações de função objetivo.
- Número de variáveis de decisão: 10.

- Número de funções objetivo: variável de acordo com o problema teste. Para os problemas IMOP[1-4] são utilizadas duas funções objetivo por problema, enquanto que nos problemas. IMOP[5-8] e DTLZ[1-9] são utilizadas três funções objetivo por problema.
- Limite de colisões L: 100.
- Intervalo escolhido para k_i: $[0.05; 0.07]$.
- Valor de correção b: 2.

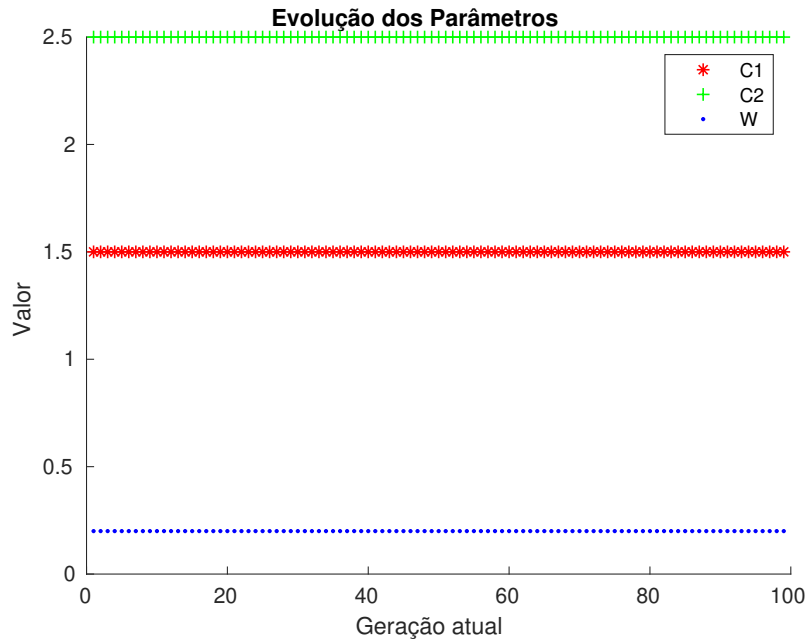

Figura 4.6 – Ilustração da evolução média dos parâmetros na simulação 2 em 50 execuções do algoritmo

A figura 4.6 mostra a estagnação dos parâmetros do algoritmo. Esta simulação, utilizando um valor alto para o limite de colisões, tem como objetivo mostrar o cenário em que exista pouca ou nenhuma colisão. O valor do contador c, em todos os casos desta simulação, não foi incrementado e por isso nenhum dos parâmetros sofreram alguma alteração. Veja que as tabelas relativas às métricas de desempenho, mostradas em 4.5, 4.3.2 e 4.3.2, são idênticas e por consequência a fronteira de Pareto produzida é exatamente a mesma que o algoritmo original. As figuras 4.5, 4.3.2, 4.3.2 e 4.8 ilustram os mesmos resultados destas tabelas. A evolução dos parâmetros permanece constante e por isso não há mudança alguma nas métricas.

Reparamos que em ambas as simulação, a nova versão do algoritmo utiliza tempo de execução muito maior, chegando a quase 100 vezes mais do que a versão original. Isto se dá pelo processo de busca na tabela, operações de inserção e remoção e recálculo dos operadores. Como pode-se reparar do pseudo-código 3.2, todas estas operações são feitas dentro do laço de avaliação das funções. No pior caso, a nova versão torna o algoritmo k vezes mais lento devido

ao laço da função de adaptação, mostrada em 3.3, onde k é o tamanho da lista encadeada no índice da tabela. O mau desempenho no tempo de execução era esperado, e neste trabalho foi um *trade-off* para melhor performance em outras métricas.

Problem	MOPSOCD	memMOPSOCD
DTLZ1	7.9976e+0 (1.13e+0)	7.9976e+0 (1.13e+0)
DTLZ2	5.8189e-2 (7.53e-3)	5.8189e-2 (7.53e-3)
DTLZ3	5.2797e+1 (6.83e+0)	5.2797e+1 (6.83e+0)
DTLZ4	9.9305e-2 (1.45e-2)	9.9305e-2 (1.45e-2)
DTLZ5	8.6376e-2 (1.62e-2)	8.6376e-2 (1.62e-2)
DTLZ6	1.2270e-1 (1.66e-1)	1.2270e-1 (1.66e-1)
DTLZ7	1.1953e+0 (4.34e-1)	1.1953e+0 (4.34e-1)
DTLZ8	2.9053e-2 (7.22e-3)	2.9053e-2 (7.22e-3)
DTLZ9	7.9895e-1 (4.31e-1)	7.9895e-1 (4.31e-1)
IMOP1	1.4219e-1 (9.97e-2)	1.4219e-1 (9.97e-2)
IMOP2	9.8269e-2 (5.77e-2)	9.8269e-2 (5.77e-2)
IMOP3	2.0959e-1 (9.05e-2)	2.0959e-1 (9.05e-2)
IMOP4	9.0390e-2 (2.98e-2)	9.0390e-2 (2.98e-2)
IMOP5	5.3115e-2 (4.75e-2)	5.3115e-2 (4.75e-2)
IMOP6	7.8699e-2 (2.69e-2)	7.8699e-2 (2.69e-2)
IMOP7	5.6365e-2 (1.88e-2)	5.6365e-2 (1.88e-2)
IMOP8	9.3206e-2 (2.35e-2)	9.3206e-2 (2.35e-2)

Tabela 4.5 – Métrica GD para a Simulação 2. Veja Gráfico 4.7

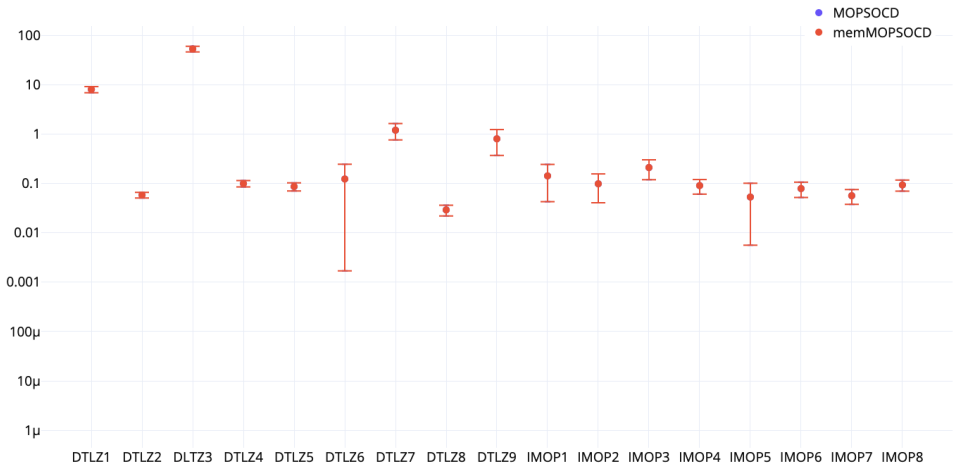

Figura 4.7 – Gráfico para a tabela 4.5, contendo a o valor médio da métrica GD e o erro em formato de barra e em escala logarítmica

Problem	MOPSOCD	memMOPSOCD
DTLZ1	3.2864e+1 (2.15e+0)	3.2864e+1 (2.15e+0)
DTLZ2	3.0406e-1 (2.55e-2)	3.0406e-1 (2.55e-2)
DTLZ3	1.9542e+2 (1.27e+1)	1.9542e+2 (1.27e+1)
DTLZ4	6.4488e-1 (7.33e-2)	6.4488e-1 (7.33e-2)
DTLZ5	2.1549e-1 (3.01e-2)	2.1549e-1 (3.01e-2)
DTLZ6	7.4684e-1 (3.83e-1)	7.4684e-1 (3.83e-1)
DTLZ7	2.9049e+0 (9.29e-1)	2.9049e+0 (9.29e-1)
DTLZ8	1.9778e-1 (1.71e-1)	1.9778e-1 (1.71e-1)
DTLZ9	1.6011e+0 (7.70e-1)	1.6011e+0 (7.70e-1)
IMOP1	6.0385e-1 (9.95e-2)	6.0385e-1 (9.95e-2)
IMOP2	4.1550e-1 (8.53e-2)	4.1550e-1 (8.53e-2)
IMOP3	5.7331e-1 (7.99e-2)	5.7331e-1 (7.99e-2)
IMOP4	5.6204e-1 (6.83e-2)	5.6204e-1 (6.83e-2)
IMOP5	5.1913e-1 (3.20e-2)	5.1913e-1 (3.20e-2)
IMOP6	5.8069e-1 (7.09e-2)	5.8069e-1 (7.09e-2)
IMOP7	6.3526e-1 (3.23e-2)	6.3526e-1 (3.23e-2)
IMOP8	6.3473e-1 (5.07e-2)	6.3473e-1 (5.07e-2)

Tabela 4.6 – Métrica IGD para a Simulação 2. Veja Gráfico 4.8

Problem	MOPSOCD	memMOPSOCD
DTLZ1	6.9992e-3 (1.61e-2)	6.9992e-3 (1.61e-2)
DTLZ2	3.6128e-1 (4.02e-2)	3.6128e-1 (4.02e-2)
DTLZ3	2.0761e-1 (8.77e-2)	2.0761e-1 (8.77e-2)
DTLZ4	5.5327e-2 (5.57e-2)	5.5327e-2 (5.57e-2)
DTLZ5	6.9042e-1 (8.51e-2)	6.9042e-1 (8.51e-2)
DTLZ6	1.0499e-1 (1.54e-1)	1.0499e-1 (1.54e-1)
DTLZ7	1.8218e-1 (2.21e-1)	1.8218e-1 (2.21e-1)
DTLZ8	2.5190e-1 (1.58e-1)	2.5190e-1 (1.58e-1)
DTLZ9	3.5154e-1 (2.92e-1)	3.5154e-1 (2.92e-1)
IMOP1	1.8811e-1 (1.16e-1)	1.8811e-1 (1.16e-1)
IMOP2	2.2541e-1 (6.30e-2)	2.2541e-1 (6.30e-2)
IMOP3	3.4554e-1 (1.33e-1)	3.4554e-1 (1.33e-1)
IMOP4	1.3811e-1 (7.92e-2)	1.3811e-1 (7.92e-2)
IMOP5	2.4355e-1 (1.50e-1)	2.4355e-1 (1.50e-1)
IMOP6	1.7899e-1 (4.80e-2)	1.7899e-1 (4.80e-2)
IMOP7	3.4874e-2 (1.14e-2)	3.4874e-2 (1.14e-2)
IMOP8	1.1184e-1 (3.74e-2)	1.1184e-1 (3.74e-2)

Tabela 4.7 – Métrica HV para a Simulação 2. Veja Gráfico 4.9

Problem	MOPSOCD	memMOPSOCD
DTLZ1	5.4145e-1 (1.64e-2)	1.6161e+1 (4.09e-1)
DTLZ2	5.6995e-1 (2.22e-2)	1.8315e+1 (6.42e-1)
DTLZ3	5.5382e-1 (2.19e-2)	1.7255e+1 (5.52e-1)
DTLZ4	5.5027e-1 (2.39e-2)	1.8473e+1 (8.07e-1)
DTLZ5	5.5004e-1 (1.48e-2)	1.8930e+1 (6.86e-1)
DTLZ6	5.6898e-1 (2.93e-2)	1.8023e+1 (8.24e-1)
DTLZ7	5.7718e-1 (1.74e-2)	1.6486e+1 (5.55e-1)
DTLZ8	6.5521e-1 (3.59e-2)	1.8755e+1 (7.40e-1)
DTLZ9	2.1097e-1 (1.01e-2)	4.6524e+0 (1.93e-1)
IMOP1	5.3875e-1 (1.58e-2)	1.6494e+1 (5.83e-1)
IMOP2	5.3414e-1 (1.57e-2)	1.5958e+1 (4.82e-1)
IMOP3	5.2752e-1 (1.24e-2)	1.6475e+1 (6.56e-1)
IMOP4	7.1528e-1 (1.83e-2)	2.6203e+1 (8.26e-1)
IMOP5	5.4089e-1 (2.02e-2)	1.8887e+1 (7.65e-1)
IMOP6	5.5188e-1 (2.29e-2)	1.8434e+1 (6.96e-1)
IMOP7	5.5540e-1 (3.87e-2)	1.8687e+1 (5.14e-1)
IMOP8	5.5303e-1 (2.45e-2)	1.8335e+1 (8.11e-1)

Tabela 4.8 – Tempo de execução em segundos na Simulação 2. Veja Gráfico 4.10

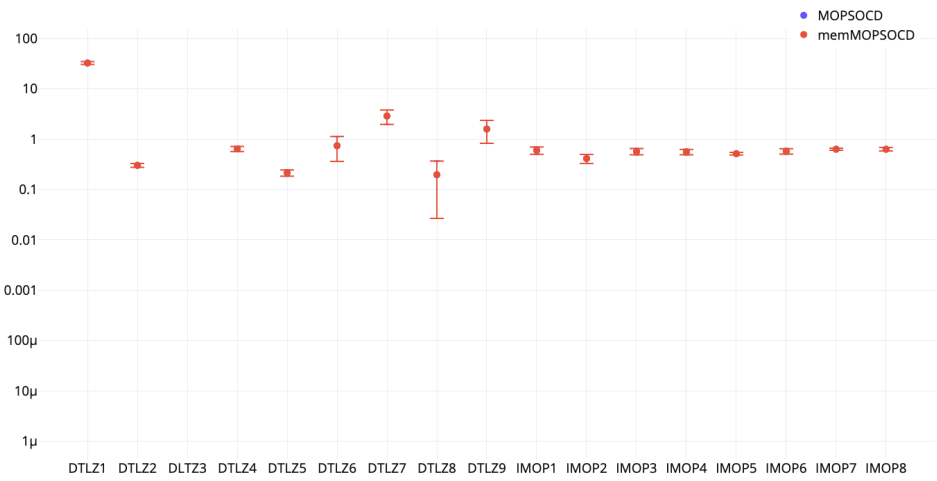

Figura 4.8 – Gráfico para a tabela 4.3.2, contendo a o valor médio da métrica IGD e o erro em formato de barra e em escala logarítmica

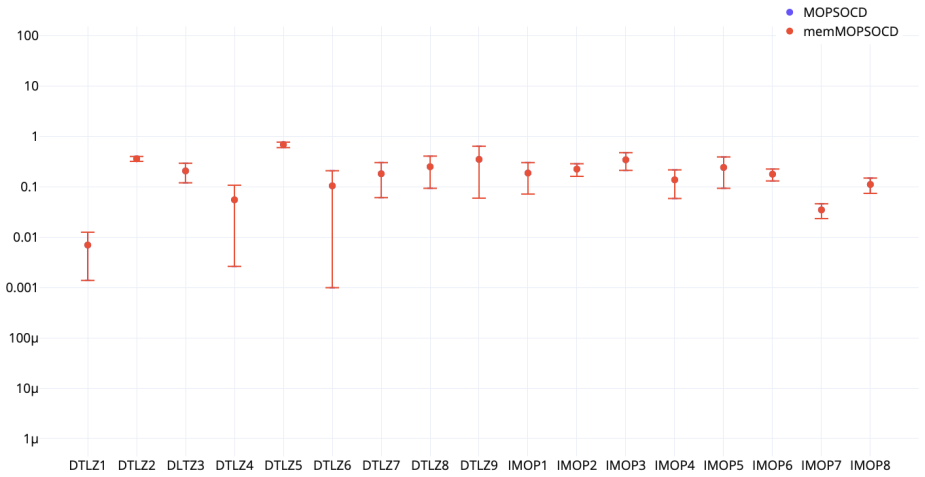

Figura 4.9 – Gráfico para a tabela 4.3.2, contendo a o valor médio da métrica HV e o erro em formato de barra e em escala logarítmica

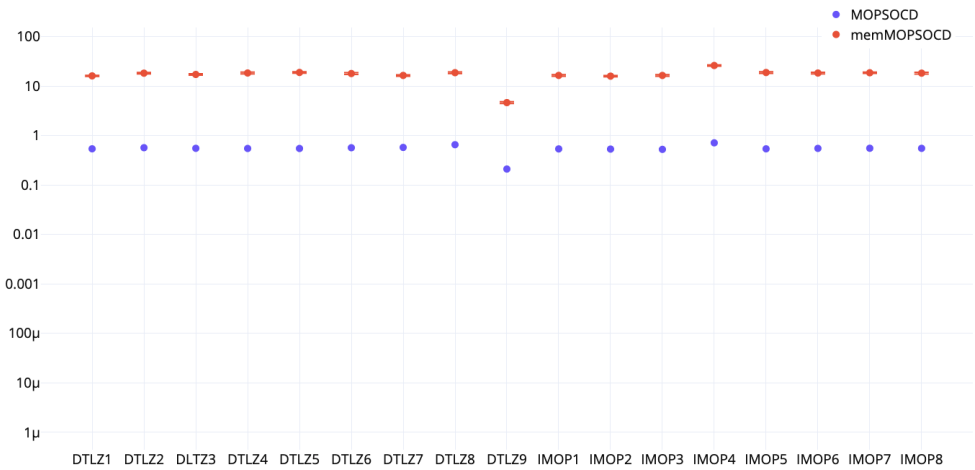

Figura 4.10 – Gráfico para a tabela 4.8, contendo a o valor médio da métrica tempo de execução e o erro em formato de barra e em escala logarítmica

5 Considerações Finais

Os objetivos deste trabalho foram: fazer um estudo profundo sobre o que existe de fundamentos e de empiricismo no que cerca o algoritmo de otimização por enxame de partículas (MOPSO-CD); a partir disto, projetar e implementar uma nova versão do algoritmo, que ainda utilize parte da estrutura da versão original, porém com parâmetros de voo adaptativos; implementar um modelo de memória para armazenamento das soluções já avaliadas, e utilizar esta memória para fazer a adaptação do parâmetros; realizar os testes estatísticos necessários, e comparar se a versão proposta supera, em alguma métrica, a versão original com parâmetros fixos.

Vimos até aqui vários destes fundamentos e testes empíricos para o PSO e suas variantes, como o MOPSO-CD. Os parâmetros de voo c_1, c_2 e ω possuem um impacto forte na direção e magnitude em que as partículas do enxame estão buscando. Valores altos de c_1 e ω somados à valores altos de c_2 induzem à uma busca mais fechada à um mesmo sub-espaço. Enquanto que, valores baixos de c_1 e ω e altos para c_2 tornam a busca por novas regiões mais eficiente. É também mostrado nestes fundamentos que, valores indefinidamente grande para os parâmetros, ou seja, que não possuem nenhum limite pré-estabelecido, o algoritmo se torna instável e ineficiente. É com base nestes que conceitos, mostrados na seção 2.4.1, que montamos parte da nossa proposta.

Também foi desenvolvido uma estrutura de dados que age como uma memória contendo a soluções já avaliadas. Esta estrutura para um algoritmo como o MOPSO-CD é eficiente, pois o algoritmo produz muitas soluções iguais durante o processo de busca. A soma desta estrutura de memória com os testes empíricos e fundamentos sobre os parâmetros do algoritmo nos trouxeram até o operador PBMM. Este operador utiliza uma lógica bem simples de pertubação nos parâmetros a partir da memória, como mostrado no capítulo 3.

Os testes da sub-seção 4.3 nos mostram resultados superiores da versão com o PBMM em relação à versão original na maioria das métricas de desempenho propostas. Nas métricas de GD e IGD, por exemplo, que medem a convergência para a fronteira de Pareto real e a diversidade das soluções, obtivemos resultados muito superiores. Na métrica de HV, apesar de não ter havido uma mudança tão significativa, o algoritmo performou igual ou superior à versão original do algoritmo.

O algoritmo possui agora uma sub-rotina (detalhada em 3.3) que assintoticamente o tornariam mais lento em qualquer cenário que compararmos com a versão original. Por este motivo, o algoritmo performou de maneira muito pior na métrica de tempo de execução, chegando a ser mais de dez vezes mais lento que o original. Apesar deste resultado ser esperado desde o princípio do trabalho, propostas futuras podem ser feitas para melhorar o desempenho do algoritmo nesta métrica (detalhado em 5.1).

Outro fato importante para salientarmos é de que a memória algorítmica em meta-

heurísticas (detalhada em 3.2) pode ser uma forma eficiente de realizar a adaptação dos parâmetros no MOPSO-CD, e não somente um método para extração das soluções já avaliadas para evitar a re-avaliação. A seção de resultados 4.3 mostra os gráficos da evolução dos parâmetros conforme as novas gerações. A interpretação destes gráficos conclui que a partir de uma determinada geração, o algoritmo começa a produzir muitas soluções iguais e que a adaptação dos parâmetros, ao gerar estas pertubações, pode fazer com que o algoritmo saia de um ótimo local.

Concluímos que, com base em todas as informações até então, o resultado deste trabalho foi satisfatório, pois: conseguimos visualizar a produção de soluções idênticas durante o processo evolutivo, e com base nisso tomar uma decisão em relação à busca; propor uma nova forma de realizar a adaptação dos parâmetros, mantendo a estabilidade e convergência do algoritmo, inclusive melhorando-o em algumas métricas de desempenho; tornar a memória algorítmica, que até então era ferramenta utilizada para evitar re-avaliações, uma ferramenta de tomada de decisão em relação ao processo evolutivo.

5.1 Trabalhos Futuros

Apesar da boa performance do algoritmo, algumas características deste podem ser feitas ou refeitas para este trabalho. Veja, que na seção de resultados 4.3, a métrica mais destoante em que o algoritmo perde é o tempo de execução. Apesar da memória ser uma tabela *hash*, que possui rápida inserção e remoção de elementos, no pior caso, caso não encontremos a solução na chave *hash* buscada, o algoritmo percorrerá toda a memória com a soluções. O que pode acontecer então é uma busca exaustiva na memória, juntamente com o cálculo de valor de *hash* para um variável N-dimensional (onde N é o número de variáveis de decisão), remoção de elementos caso haja uma colisão e recálculo dos operadores. Todas estas operações aumentam o custo do algoritmo tanto em tempo de execução quanto em espaço em memória. Trabalhos futuros podem visar otimizar estas operações para o algoritmo ter um desempenho igual ou melhor na métrica de tempo de execução.

Outra possível melhoria em relação à memória pode ser em relação à estrutura de dados escolhida. A tabela *hash*, dependendo da função escolhida e do tamanho da tabela, pode enviesar o processo de re-avaliações, fazendo com que colisões não sejam entendidas explicitamente como produção de soluções similares. Os índices da tabela, em problemas reais que foram testados, não diziam muito sobre a correlação das soluções, ou seja, um mesmo índice contém soluções que podem ou não serem similares. Logo, a escolha de uma árvore do tipo BSP, como mostrado nos trabalhos relacionados em 2.4.1 e 3.1, pode ser mais efetiva. Esta árvore trabalha com nós que correspondem à sub-espaços de um espaço original. A busca por uma solução equivale-se à uma busca em profundidade na árvore, onde o nó folha corresponde a uma única solução. No pior caso, buscas em árvores possuem complexidade assintótica igual a $O(n)$ para n igual ao número de avaliações de função objetivo, enquanto que uma busca em tabela *hash* neste mesmo caso

possui complexidade igual a $O(m)$ onde m é o tamanho da lista encadeada no índice de colisão, provado em (CORMEN et al., 2009). Em geral é verdade que $m < n$, a tabela normalmente é mais rápida. Mesmo que a tabela seja mais eficiente em tempo e igual em espaço, a árvore sempre possuirá no seu nó folha uma única solução, e cada um dos sub-espaços possuem somente soluções correlacionadas. Portanto, o uso da árvore pode ser melhor neste caso de uso, por motivos de maior confiabilidade de que um sub-espaço possua soluções de mesma característica.

Por fim, testes utilizando funções inteiras ou binárias podem ser adicionados nos resultados, para que possamos investigar novas naturezas de problemas. Outro ponto importante é também realizar uma investigação a cerca das funções reais retratadas aqui, e algumas de suas particularidades. Algumas destas funções teste são mais complexas e em algumas instâncias ambos os algoritmos tiveram problemas na convergência para a fronteira de Pareto real do problema. Novas formas de adaptação específicas para uma natureza específica de funções de teste pode ser estudadas mais adiante a fim de tornar o algoritmo ainda mais eficaz.

Referências

BELLMAN, R. E. Dynamic Programming. USA: Dover Publications, Inc., 2003. ISBN 0486428095.

CARLISLE, A.; DOZIER, G. An off-the-shelf pso. In: . [S.l.: s.n.], 2001.

CARRANO, E.; MOREIRA, L.; TAKAHASHI, R. A new memory based variable-length encoding genetic algorithm for multiobjective optimization. In: . [S.l.: s.n.], 2011. v. 6576, p. 328–342.

CHANKONG, V.; HAIMES, Y. Y. Optimization-based methods for multiobjective decision-making-an overview. Large Scale Systems In Information And Decision Technologies, ELSEVIER SCIENCE BV PO BOX 211, 1000 AE AMSTERDAM, NETHERLANDS, v. 5, n. 1, p. 1–33, 1983.

CHANKONG, V.; HAIMES, Y. Y. Multiobjective decision making: theory and methodology. [S.l.]: Courier Dover Publications, 2008.

CHARNES, A.; COOPER, W. W. Goal programming and multiple objective optimizations: Part 1. European journal of operational research, Elsevier, v. 1, n. 1, p. 39–54, 1977.

CLERC, M. Particle swarm optimization. [S.l.]: John Wiley & Sons, 2010. v. 93.

Clerc, M.; Kennedy, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, v. 6, n. 1, p. 58–73, 2002.

COELHO, D. Estruturas de Memória e Operadores Adaptativos em Meta-Heurísticas. Tese (Doutorado) — Tese de Doutorado. UFMG-PPGEE, 2016.

COELLO, C.; CORTéS, N. Solving multiobjective optimization problems using an artificial immune system. Genetic Programming and Evolvable Machines, v. 6, p. 163–190, 06 2005.

Coello Coello, C. A.; Lechuga, M. S. Mopso: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). [S.l.: s.n.], 2002. v. 2, p. 1051–1056 vol.2.

COHON, J. L. Multicriteria programming: Brief review and application. In: Design optimization. [S.l.]: Elsevier, 1985. p. 163–191.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to Algorithms, Third Edition. 3rd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262033844.

DANTZIG, G. B. Problems for the Numerical Analysis of the Future. [S.l.]: US Government Printing Office, 1951. v. 15.

DEB, K.; KALYANMOY, D. Multi-Objective Optimization Using Evolutionary Algorithms. USA: John Wiley & Sons, Inc., 2001. ISBN 047187339X.

DEB, K.; THIELE, L.; LAUMANNS, M.; ZITZLER, E. Scalable test problems for evolutionary multiobjective optimization. In: ____. Evolutionary Multiobjective Optimization: Theoretical Advances and Applications. London: Springer London, 2005. p. 105–145. ISBN 978-1-84628-137-2. Disponível em: <https://doi.org/10.1007/1-84628-137-7_6>.

FISHBURN, P. C. Exceptional paper—lexicographic orders, utilities and decision rules: A survey. Management science, INFORMS, v. 20, n. 11, p. 1442–1471, 1974.

GALE, D.; KUHN, H. W.; TUCKER, A. W. Linear programming and the theory of games. Activity analysis of production and allocation, Cowles Commission Monographs, v. 13, p. 317–335, 1951.

Gao, Y.; An, X.; Liu, J. A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation. In: 2008 International Conference on Computational Intelligence and Security. [S.l.: s.n.], 2008. v. 1, p. 61–65.

GASS, S.; SAATY, T. The computational algorithm for the parametric objective function. Naval research logistics quarterly, Wiley Online Library, v. 2, n. 1-2, p. 39–45, 1955.

Guimin Chen; Xinbo Huang; Jianyuan Jia; Zhengfeng Min. Natural exponential inertia weight strategy in particle swarm optimization. In: 2006 6th World Congress on Intelligent Control and Automation. [S.l.: s.n.], 2006. v. 1, p. 3672–3675.

HAN, H.-G.; LU, W.; QIAO, J. An adaptive multiobjective particle swarm optimization based on multiple adaptive methods. IEEE Transactions on Cybernetics, PP, p. 1–14, 04 2017.

Hu, W.; Yen, G. G. Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system. IEEE Transactions on Evolutionary Computation, v. 19, n. 1, p. 1–18, 2015.

IGNIZIO, J. P. Generalized goal programming an overview. Computers & Operations Research, Elsevier, v. 10, n. 4, p. 277–289, 1983.

JAIMES, A. L.; ZAPOTECAS-MARTíNEZ, S.; COELLO, C. An introduction to multiobjective optimization techniques. In: ____. [S.l.: s.n.], 2011. p. 29–57. ISBN 978-1-61122-818-2.

KANTOROVICH, L. V. Mathematical methods of organizing and planning production. Manage. Sci., INFORMS, Linthicum, MD, USA, v. 6, n. 4, p. 366–422, jul. 1939. ISSN 0025-1909. Disponível em: <https://doi.org/10.1287/mnsc.6.4.366>.

KARUSH; KUHN; TUCKER. Nonlinear programming. In: Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability. [S.l.: s.n.], 1951. p. 481–492.

KENNEDY, J. The behavior of particles. In: PORTO, V. W.; SARAVANAN, N.; WAAGEN, D.; EIBEN, A. E. (Ed.). Evolutionary Programming VII. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 579–589. ISBN 978-3-540-68515-9.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. Proceedings of ICNN'95-International Conference on Neural Networks. [S.l.], 1995. v. 4, p. 1942–1948.

LAMONT, G. B.; VELDHUIZEN, D. A. van. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. In: . [S.l.: s.n.], 1999.

LIN, Q.; LI, J.; DU, Z.; CHEN, J.; MING, Z. A novel multi-objective particle swarm optimization with multiple search strategies. European Journal of Operational Research, v. 247, n. 3, p. 732 – 744, 2015. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221715006256>.

Ma, B.; Hua, J.; Ma, Z.; Li, X. Imopso: An improved multi-objective particle swarm optimization algorithm. In: 2016 5th International Conference on Computer Science and Network Technology (ICCSNT). [S.l.: s.n.], 2016. p. 376–380.

MARTÍNEZ, S. Z.; COELLO, C. A. C. A multi-objective particle swarm optimizer based on decomposition. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. New York, NY, USA: Association for Computing Machinery, 2011. (GECCO '11), p. 69–76. ISBN 9781450305570. Disponível em: <https://doi.org/10.1145/2001576.2001587>.

MIETTINEN, K. Nonlinear multiobjective optimization. [S.l.]: Springer Science & Business Media, 2012. v. 12.

MIETTINEN, K.; MÄKELÄ, M. Interactive bundle-based method for nondifferentiable multiobjeective optimization: Nimbus. Optimization, Taylor & Francis, v. 34, n. 3, p. 231–246, 1995.

PARETO, V. Cours D'Economie Politique. Lausanne, Switzerland: F. Rouge, 1896.

PARSOPOULOS, K. E.; VRAHATIS, M. N. Particle swarm optimization and intelligence: advances and applications. Information Science Reference New York, 2010.

PEARSON, P. K. Fast hashing of variable-length text strings. Commun. ACM, Association for Computing Machinery, New York, NY, USA, v. 33, n. 6, p. 677–680, jun. 1990. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/78973.78978>.

RAQUEL, C. R.; NAVAL, P. C. An effective use of crowding distance in multiobjective particle swarm optimization. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. New York, NY, USA: Association for Computing Machinery, 2005. (GECCO '05), p. 257–264. ISBN 1595930108. Disponível em: <https://doi.org/10.1145/1068009.1068047>.

Riquelme, N.; Von Lücken, C.; Baran, B. Performance metrics in multi-objective optimization. In: 2015 Latin American Computing Conference (CLEI). [S.l.: s.n.], 2015. p. 1–11.

Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). [S.l.: s.n.], 1998. p. 69–73.

SILVA, R. C. P. Um estudo sobre a configuração automática do algoritmo de evolução diferencial. Universidade Federal de Minas Gerais, 2012. Disponível em: <http://hdl.handle.net/1843/BUOS-9ABKE7>.

SU, Y.; GUO, N.; TIAN, Y.; ZHANG, X. A non-revisiting genetic algorithm based on a novel binary space partition tree. Information Sciences, v. 512, p. 661 – 674, 2020. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025519309739>.

Tian, Y.; Cheng, R.; Zhang, X.; Jin, Y. Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. IEEE Computational Intelligence Magazine, v. 12, n. 4, p. 73–87, 2017.

Tian, Y.; Cheng, R.; Zhang, X.; Li, M.; Jin, Y. Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [research frontier]. IEEE Computational Intelligence Magazine, v. 14, n. 3, p. 61–74, 2019.

XIN, J.; CHEN, G.; HAI, Y. A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In: . [S.l.: s.n.], 2009. p. 505–508.

YANG, Z.; WU, A. A non-revisiting quantum-behaved particle swarm optimization based multilevel thresholding for image segmentation. Neural Computing and Applications, v. 32, 08 2020.

YV, Y. H.; LASDON, L. S.; WISMER, D. D. On a bicriterion formation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man and Cybernetics, Institute of Electrical and Electronics Engineers Inc., n. 3, p. 296–297, 1971.

ZADEH, L. Optimality and non-scalar-valued performance criteria. IEEE transactions on Automatic Control, IEEE, v. 8, n. 1, p. 59–60, 1963.

ZHANG, X.; ZHENG, X.; CHENG, R.; QIU, J.; JIN, Y. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. Information Sciences, v. 427, p. 63 – 76, 2018. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025517310344>.

ZITZLER, E.; BROCKHOFF, D.; THIELE, L. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: OBAYASHI, S.; DEB, K.; POLONI, C.; HIROYASU, T.; MURATA, T. (Ed.). Evolutionary Multi-Criterion Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 862–876. ISBN 978-3-540-70928-2.

Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation, v. 3, n. 4, p. 257–271, 1999.

Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C. M.; da Fonseca, V. G. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation, v. 7, n. 2, p. 117–132, 2003.