Estratégias bio-inspiradas aplicadas em problemas discretos com muitos objetivos

Tiago Peres França

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2018

Tiago Peres França

Estratégias bio-inspiradas aplicadas em problemas discretos com muitos objetivos

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Gina Maira Barbosa de Oliveira
Coorientador: Luiz Gustavo Almeida Martins

Uberlândia
2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

F814e
2018

França, Tiago Peres, 1989-
Estratégias bio-inspiradas aplicadas em problemas discretos com muitos objetivos [recurso eletrônico] / Tiago Peres França. - 2018.

Orientadora: Gina Maira Barbosa de Oliveira.
Coorientador: Luiz Gustavo Almeida Martins.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.368
Inclui bibliografia.
Inclui ilustrações.

1. Computação. 2. Algoritmos genéticos. 3. Roteamento (Administração de redes de computadores). 4. Problema da mochila (Matemática). I. Oliveira, Gina Maira Barbosa de, 1967- (Orient.). II. Martins, Luiz Gustavo Almeida, 1974-, (Coorient.). III. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Maria Salete de Freitas Pinheiro - CRB6/1262

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Estratégias bioinspiradas aplicadas em problemas discretos com muitos objetivos**" por **Tiago Peres França** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação.**

Uberlândia, 15 de junho de 2018

Orientadora: _____
Prof. Dr. Gina Maira Barbosa de Oliveira
Universidade Federal de Uberlândia

Coorientador: _____
Prof. Dr. Luiz Gustavo Almeida Martins
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Paulo Henrique Ribeiro Gabriel
Universidade Federal de Uberlândia

_____
Prof. Dr. Myriam R. B. S. Delgado
Universidade Tecnológica Federal do Paraná

Este trabalho é dedicado
aos meus orientadores Gina Maira e Luiz Gustavo, pela dedicação;
aos meus pais Wagner e Iara, pelo apoio;
e aos meus irmãos e amigos, pelo companheirismo.

Agradecimentos

Agradeço principalmente à minha orientadora Gina Maira pela paciência e pelos ensinamentos durante esta trajetória. Agradeço ao meu co-orientador Luiz Gustavo pelas grandes ideias e correções de texto. Agradeço aos professores José Gustavo e Rita Maria pelas excelentes aulas durante o curso e à professora Márcia Aparecida, que além de ter ministrado suas reconhecidamente ótimas aulas de análise de algoritmos, sempre me incentivou a obter o título de mestre.

Meus agradecimentos também se extendem aos meus pais: Wagner e Iara; meus irmãos: Vinícius, Bruno e Felipe; e aos meus grandes amigos: Lucas, Ana Carolyne, João Marcos, Mariana, Débora e Lívia que sempre me apoiaram e, direta ou indeiretamente, contribuiram muito para a realização deste trabalho.

Agradeço também a Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG) que providenciou o apoio financeiro para este trabalho.

"Viver é arriscar tudo. Se não fosse, seríamos apenas um amontoado inerte de moléculas organizadas aleatóriamente flutuando onde quer que o universo nos sopre."
(Rick and Morty)

Resumo

Problemas de otimização multiobjetivo são muito comuns no nosso dia-a-dia e surgem em diversas áreas do conhecimento. Neste trabalho, exploram-se e comparam-se várias estratégias de otimização multiobjetivo em dois problemas discretos bem conhecidos na computação: o problema da mochila e o problema do roteamento *multicast*. Dentre as estratégias para solucionar problemas multiobjetivos discretos, destacam-se os algoritmos genéticos (AGs) e os algoritmos baseados em colônias de formigas (ACOs). Ambas as abordagens são investigadas neste trabalho por meio de experimentos que avaliaram o desempenho de 10 algoritmos diferentes. Todos os algoritmos avaliados foram adaptados para os problemas propostos e, além disso, desenvolveu-se um novo algoritmo denominado *Many-objective Ant Colony Optimization based on Non-Dominated Sets* (MACO/NDS), o qual foi avaliado e comparado com todos os outros métodos aqui investigados, considerando-se diferentes métricas de desempenho. A partir dos resultados experimentais, foi possível comprovar a eficiência e eficácia do método proposto. Em diversos cenários, ele foi capaz de encontrar conjuntos de soluções superiores aos produzidos pelos demais algoritmos e levou menos tempo para executar.

Palavras-chave: algoritmos many-objectives; algoritmos genéticos; otimização por colônia de formigas; problema do roteamento multicast; problema da mochila multiobjetivo.

Abstract

Multi-objective optimization problems are very common in the day-to-day life and come up in many fields of knowledge. In this work, several strategies for multi-objective optimization have been explored and compared on two well known discrete problems in computer science: the knapsack problem and the multicast routing problem. Among all strategies to solve multi-objective discrete problems, genetic algorithms (GAs) and ant colony optimization (ACO) are the ones which generally provide the best results. In this work both approaches are explored through several experiments involving 10 different algorithms. All of the algorithms evaluated were adapted to the proposed problems. Furthermore, a new algorithm has been proposed, the Many-objective Ant Colony Optimization based on Non-Dominated Sets (MACO/NDS), which has been evaluated and compared against all other methods investigated here, considering different performance metrics. In many scenarios, it has been capable of finding superior sets of solutions and took less time to execute.

Keywords: many-objective algorithms; genetic algorithms; ant colony optimization; multicast routing problem; multi-objective knapsack problem.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ACO *Ant Colony Optimization*

AEMMD Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias

AEMMT Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas

AEMO Algoritmo Evolutivo Multiobjetivo

AG Algoritmo Genético

BACO *Binary Ant Colony Optimization*

BRACIS *Brazilian Conference on Intelligent Systems*

ER Taxa de Erro

GD *Generational Distance*

GDp *Generational Distance (Excluding Pareto)*

HV Hiper-Volume

IGD *Inverted Generational Distance*

MACO ACO multiobjetivo

MACO/NDS *Many-objective Ant Colony Optimization based on Non-dominated Decomposed Sets*

MEAMT *Multi-Objective Evolutionary Algorithm with Many Tables*

MEANDS *Multi-Objective Evolutionary Algorithm based on Non-dominated Decomposed Sets*

MOACS *Multi-Objective Ant Colony Optimization Algorithm*

MOEA/D *Multiobjective Evolutionary Algorithm Based on Decomposition*

MOEA/D-ACO *Multiobjective Evolutionary Algorithm Based on Decomposition ACO*

MOGA *Multi-Objective Genetic Algorithms*

NSGA *Non-Dominated Sorting Genetic Algorithm*

NSGA-II *Non-Dominated Sorting Genetic Algorithm II*

NSGA-III *Non-Dominated Sorting Genetic Algorithm III*

PM Problema da Mochila

PMM Problema da Mochila Multiobjetivo

PMO Problema Multiobjetivo

PRM Problema do Roteamento Multicast

PS *Pareto Subset*

PSO *Particle Swarm Optimization*

QoS Qualidade de Serviço

SDE *Shift-Based Density Estimation*

SMTTP *Single Machine Total Tardiness Problem*

SPEA *Strength Pareto Evolutionary Algorithm*

SPEA2 *Strength Pareto Evolutionary Algorithm 2*

SPEA2-SDE *SPEA2 with Shift-Based Density Estimation*

UBQP Problema de Programação Binária Quadrática Irrestrito

VEGA Vector Evaluated Genetic Algorithm

Sumário

Capítulo 1

Introdução

A otimização é um campo de suma importância da ciência da computação que procura encontrar a melhor solução para um problema matemático. Em tais problemas, deve-se maximizar ou minimizar alguma característica, por exemplo, ao decidir um caminho entre duas cidades, é desejável minimizar a distância a se percorrer. Infelizmente, a maior parte dos problemas interessantes são também considerados impossíveis de se resolver em tempo razoável, por essa razão, ao invés de procurar pela melhor solução possível, é usualmente preferível aproximá-la através de algoritmos mais rápidos (CORMEN et al., 2009). A grande importância da pesquisa em otimização está na frequência com a qual esses problemas surgem no dia-a-dia, perguntas como "qual o menor caminho para atingir um destino", "qual o melhor carro para se comprar dado um orçamento" ou "qual a forma mais rápida de se executar uma tarefa" fazem parte da vida da maioria das pessoas e resolvê-las de forma eficiente ainda representa um desafio para a computação.

Uma forma de garantir a obtenção da melhor solução (solução ótima) para um dado problema é por meio da avaliação de todas as soluções possíveis (busca exaustiva). Para a maioria dos problemas interessantes, essa é uma tarefa difícil e inviável. Por exemplo, combinar e testar todas as possibilidades de caminho em um grafo com muitas arestas é um problema NP-difícil, ou seja, não pode ser resolvido em tempo hábil considerando o atual estado da arte da computação. Sendo assim, a pesquisa em otimização também busca desenvolver meios para se encontrar soluções suficientemente próximas da ótima, ou seja, trabalha-se com estratégias de aproximação (CORMEN et al., 2009). Dois dos métodos mais comuns de otimização são os algoritmos gulosos e os algoritmos de busca bio-inspirados. Dentre os bio-inspirados, destacamos os algoritmos genéticos, a otimização por colônia de formigas e a inteligência de enxames (PSO). Os algoritmos gulosos são interessantes para se resolver problemas de otimização mono-objetivo, mas existem otimizações mais complexas, onde se deve considerar mais de uma característica (problemas multiobjetivos). Por exemplo, ao escolher um carro, uma pessoa não se preocupa apenas com o preço, mas também com o desempenho, a durabilidade, o consumo, entre outros fatores. Nesse caso, não basta otimizar um único objetivo, mas encontrar uma opção

que apresente uma boa relação custo-benefício considerando todos os fatores avaliados. A maneira mais simples de se contornar esse problema é transformando as diversas métricas em uma única função através de uma média ponderada dos objetivos. Por outro lado, essa estratégia pode não ser a ideal, pois é necessário ter um conhecimento prévio do problema para se decidir a relevância (peso) de cada objetivo. Por essa razão, normalmente deseja-se encontrar todas as soluções que são superiores às demais em pelo menos um dos objetivos. Dessa forma, existirão várias soluções que podem ser classificadas como melhores, dependendo da métrica analisada. Esse conjunto de soluções é denominado conjunto Pareto ótimo. Assim, na solução de Problemas Multiobjetivos (PMOs), se faz natural a escolha dos algoritmos de busca que apresentem múltiplas soluções e aproximem, da melhor maneira possível, o conjunto Pareto ótimo (SRINIVAS; DEB, 1994).

Muitos problemas da vida real podem tirar proveito da otimização multiobjetivo, fazendo necessária a elaboração de estratégias eficientes para resolvê-los. Vários algoritmos bio-inspirados foram propostos nos últimos anos (DEB et al., 2002a; ZITZLER; LAUMANNS; THIELE, 2001; DEB; JAIN, 2014), tornando a otimização multiobjetivo um campo com alta atividade no ramo da Computação Bio-inspirada. O primeiro algoritmo evolutivo multiobjetivo (AEMO) proposto foi o Vector Evaluated Genetic Algorithm (VEGA) (SCHAFFER, 1984), mas os métodos mais bem consolidados na literatura são o NSGA-II (DEB et al., 2002a) e o SPEA2 (ZITZLER; LAUMANNS; THIELE, 2001), propostos no início dos anos 2000 e considerados os métodos clássicos da otimização multiobjetivo. Entretanto, mais recentemente, foi levantado que, devido a uma pressão seletiva fraca em direção ao conjunto de soluções ótimas em espaços de alta dimensionalidade, tais *frameworks* não funcionam bem quando aplicados a problemas com muitas funções objetivo (4 ou mais) (DEB; JAIN, 2014). Os problemas com 4 ou mais objetivos são geralmente chamados de *many-objective* e para resolvê-los foram propostas novas abordagens como decomposição, e.g. MOEA/D (ZHANG; LI, 2007), relações de dominância diferenciadas, e.g. ε-MOEA (AGUIRRE; TANAKA, 2009) e evolução baseada em indicadores, e.g. HypE (BADER; ZITZLER, 2011). Recentemente, novos algoritmos genéticos foram propostos para resolver a problemática *many-objective*, dentre eles destaca-se aqui: NSGA-III (DEB; JAIN, 2014), MEAMT (BRASIL; DELBEM; SILVA, 2013) e MEANDS (LAFETÁ et al., 2018).

A maior parte dos trabalhos em otimização multiobjetivo utiliza Algoritmos Genéticos (AGs). Entretanto, outras técnicas bio-inspiradas também podem ser empregadas, tais como: os métodos baseados em colônias de formigas, em inglês *Ant Colony Optimization* (ACO), e os algoritmos inspirados em inteligência de enxame, em inglês *Particle Swarm Optimization* (PSO). Devido a seus cálculos vetoriais difíceis de se traduzir para um ambiente discreto, os PSOs são indicados especialmente para problemas contínuos e, portanto, foram deixados de fora deste estudo. Por outro lado, os ACOs lidam especialmente com problemas discretos e representações em grafos, tornando-os interessantes

para os problemas estudados. Dentre os ACOs investigados na literatura para problemas com muitos objetivos, destacam-se o MOACS (BARÁN; SCHAERER, 2003) e o MOEA/D-ACO (KE; ZHANG; BATTITI, 2013).

Neste trabalho, investigaram-se diversos algoritmos bio-inspirados presentes na literatura de otimização multiobjetivo e propôs-se um novo método de otimização baseado em colônias de formigas (ACO), o *Many-objective Ant Colony Optimization based on Non-dominated Decomposed Sets* (MACO/NDS) (FRANçA; MARTINS; OLIVEIRA, 2018), em português, otimização para muitos objetivos com colônia de formigas baseada em conjuntos de soluções não-dominadas. O algoritmo proposto adota algumas ideias do MEANDS (LAFETÁ et al., 2018), transferindo-as para uma aplicação em ACO, que tem uma abordagem diferente dos algoritmos genéticos usados na maioria dos métodos de otimização multiobjetivo.

A fim de comparar o desempenho dos algoritmos investigados em diferentes PMOs, utilizaram-se dois problemas discretos bem conhecidos na literatura de otimização multiobjetivo: o Problema da Mochila Multiobjetivo (PMM) e o Problema do Roteamento Multicast (PRM). O primeiro é uma versão multiobjetivo do clássico problema da mochila 0/1. O problema da mochila original consiste de uma mochila e um conjunto de itens, no qual deve-se encontrar a melhor combinação de objetos para se colocar na mochila de forma que não se ultrapasse a capacidade da mesma e se maximize o valor (lucro) dos itens carregados. O PMM é uma variação na qual ao invés de um único valor de lucro, todo item possui m valores (m é o número de objetivos). O PMM é um problema de combinação multimodal, no qual a ordem não influencia na qualidade da solução (várias combinações representam a mesma solução). Esse problema reflete uma abordagem teórica e apesar de testar os algoritmos em seus extremos, devido a seu enorme espaço de busca, nem sempre reflete a realidade dos problemas cotidianos. O PRM, por sua vez, é um problema de permutação, no qual a ordem influencia na eficácia da solução. Além disso, é um problema prático geralmente encontrado em comunicações de rede no qual se deseja transmitir uma mensagem de um dispositivo fonte para múltiplos destinos em uma rede de computadores, com o objetivo de utilizar os recursos disponíveis da forma mais eficiente possível. Esse problema é de extrema importância considerando-se o grande número de transmissões multimídia e aplicações em tempo real que se beneficiariam de algoritmos eficientes para cálculos de rota. Em cada problema, os vários AEMOs são avaliados em diversas formulações de objetivos, variando-se de 2 a 6 objetivos, analisando-se a qualidade das soluções obtidas e discutindo as características presentes nos algoritmos que propiciam a obtenção de determinado resultado. De acordo com os resultados experimentais do capítulo 6, o MACO/NDS mostrou-se ser um método interessante devido a sua rapidez no problema do roteamento multicast e eficácia no problema da mochila.

Os ACOs são pouco explorados na literatura multiobjetivo quando comparados aos algoritmos genéticos. Este trabalho expande a coleção de métodos inspirados em colônias de

formigas ao propor um novo *framework* ACO para problemas multiobjetivos com muitas funções de otimização (*many-objective*). Os ACOs são meta-heurísticas criadas especialmente para problemas discretos, o que pode representar uma grande vantagem em relação aos AGs, tanto em matéria de tempo como em qualidade das soluções. O MACO/NDS utiliza várias tabelas de feromônios para guiar a população de formigas, uma para cada combinação possível de objetivos. Cada grupo de formigas é associada a uma estrutura de feromônios diferente e, a cada passo do algoritmo, a atualiza de acordo com as novas soluções ótimas encontradas. Cada tabela de feromônio representa um subproblema e é responsável por guiar as soluções de acordo com um conjunto de objetivos diferentes, dessa forma, o algoritmo proposto (MACO/NDS) começa explorando subproblemas mais simples e manipula os mais complexos no decorrer do algoritmo. Para o PRM, os experimentos revelaram que o MACO/NDS, junto ao MOACS são os algoritmos mais rápidos e eficazes. Para o PRM, o MACO/NDS e MOEA/D-ACO conquistam as melhores soluções. Ou seja, o MACO/NDS é um método interessante para ambos os problemas, e dentre os ACOs testados é que apresentou maior estabilidade considerando ambos os problemas e suas diferentes instâncias.

1.1 Objetivos

Esta dissertação estende os trabalhos de Lafetá et al. (2016) e Bueno e Oliveira (2010) sobre o problema do roteamento multicast, introduzindo um novo tipo de heurística bio-inspirada multiobjetivo (as colônias de formigas) e um novo problema discreto (o problema da mochila multiobjetivo). O principal objetivo deste trabalho é propor um novo algoritmo ACO para otimização em problemas *many-objectives* e compará-lo com as principais estratégias presentes na literatura multiobjetivo tanto de AGs quanto de ACOs. Em linhas gerais, esta pesquisa almeja:

- ❏ Propor um modelo para a construção de soluções em algoritmos baseados em colônias de formigas, de acordo com as características de cada um dos problemas investigados (PMM e PRM). Tais modelos são partes essenciais para a proposição do algoritmo ACO.

- ❏ Propor o novo algoritmo baseado em ACO para problemas com muitos objetivos: O MACO/NDS.

- ❏ Investigar dois problemas multiobjetivos discretos. O problema do roteamento multicast (PRM) foi utilizado para comparar o desempenho de AEMOs nos trabalhos anteriores do nosso grupo de pesquisa. A fim de melhor entender a influência do tipo do problema no desempenho/eficiência dos algoritmos estudados, investigou-se outro problema neste trabalho, o PMM que, apesar de ter uma aplicação mais

restrita, apresenta diferenças interessantes em relação ao PRM, possibilitando uma análise mais profunda sobre comportamento dos vários algoritmos.

- ❑ Para ambos os problemas (PMM e PRM), adaptar os algoritmos encontrados na literatura e analisar o comportamento de cada um em relação à complexidade do espaço de busca e ao número de objetivos a fim de guiar as decisões sobre a construção do algoritmo proposto MACO/NDS.

- ❑ Além das métricas dependentes do conhecimento sobre o conjunto Pareto ótimo do problema, empregar uma métrica não paramétrica. Neste caso, o hipervolume foi utilizado, ele permite avaliar o desempenho dos algoritmos multi e *many-objective* em problemas nos quais a fronteira de Pareto não é conhecida. Essa métrica permitiu comprovar a vantagem dos ACOs (inclusive do método proposto, MACO/NDS) sobre os AGs em instâncias complexas do PMM e do PRM, onde não é possível estimar as fronteiras de Pareto.

1.2 Contribuições

Este trabalho contribui para o campo de otimização multiobjetivo, assim como o de comunicações em rede (através do problema do roteamento multicast). Os principais resultados desta pesquisa são resumidos nos seguintes tópicos:

- ❑ O *Multi-Objective Evolutionary Algorithm with Many Tables* (MEAMT), ou Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas (AEMMT), em português, foi proposto originalmente para predição de estruturas de proteínas (BRASIL; DELBEM; SILVA, 2013) e posteriormente, utilizado em (LAFETÁ et al., 2016) para resolver o problema do roteamento multicast. Neste trabalho, explora-se uma nova aplicação do algoritmo ao utilizá-lo para encontrar soluções para uma versão multiobjetivo do problema da mochila.

- ❑ O *Multi-Objective Evolutionary Algorithm based on Non-dominated Decomposed Sets* (MEANDS), ou Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias (AEMMD), em português, foi proposto e analisado em (LAFETÁ et al., 2016) para resolver o PRM. Sua eficácia como algoritmo multiobjetivo em outro problema diferente do roteamento ainda não havia sido investigada. Neste trabalho, mostra-se como é possível aplicar o AEMMD também ao PMM.

- ❑ Adaptou-se e executou-se alguns ACOs da literatura (*Multi-Objective Ant Colony Optimization Algorithm* (MOACS) e *Multiobjective Evolutionary Algorithm Based on Decomposition ACO* (MOEA/D-ACO)) para os problemas do roteamento multicast (PRM) e da mochila (PMM). Nessa adaptação, também foi proposto um modelo mais eficiente para a construção de soluções no PRM.

- Através da execução de vários algoritmos multi e *many-objective* sobre 2 problemas diferentes em diversos níveis de complexidade, foi feita uma análise sobre o comportamento desses algoritmos à medida em que se aumenta o número de objetivos e a complexidade do espaço de busca, possibilitando uma comparação entre os principais algoritmos da literatura.

- A principal contribuição desta dissertação está na proposição de um novo algoritmo denominado *Many-objective Ant Colony Optimization based on Non-Dominated Sets* (MACO/NDS). O MACO/NDS é uma estratégia baseada em colônia de formigas e decomposição de objetivos que resolve problemas com muitos objetivos de forma rápida e eficiente. O novo método foi aplicado aos problemas PMM e PRM e comparado com os demais algoritmos. Durante o processo de desenvolvimento desse algoritmo, diferentes estratégias para a construção de soluções em ambos os problemas (PMM e PRM) foram avaliadas e diversos experimentos foram realizados, os quais serão apresentados no decorrer do texto.

1.3 Organização do texto

Este trabalho está dividido em capítulos organizados da seguinte forma:

- **Capítulo 2, otimização bio-inspirada**: nesse capítulo apresenta-se uma visão geral sobre os algoritmos genéticos e a otimização por colônia de formigas.

- **Capítulo 3, otimização multiobjetivo**: apresenta-se a definição de problemas multiobjetivos e descreve-se cada algoritmo usado neste trabalho.

- **Capítulo 4, problemas de teste**: introduz os dois problemas explorados nesta dissertação, o Problema da Mochila Multiobjetivo (PMM) e o Problema do Roteamento Multicast (PRM), assim como as estratégias evolutivas para cada um deles.

- **Capítulo 5, algoritmo proposto**: descreve o algoritmo MACO/NDS proposto nesta dissertação e as estratégias para a construção de soluções em ACO para ambos os problemas tratados neste trabalho (PMM e PRM).

- **Capítulo 6, experimentos**: apresenta e discute todos os experimentos realizados no decorrer deste trabalho com o objetivo de avaliar a eficiência do algoritmo proposto, bem como analisar comparativamente o desempenho das abordagens investigadas na resolução de diferentes configurações de dois problemas discretos distintos.

- **Capítulo 7, conclusão**: resume as principais conclusões obtidas a partir dos experimentos e apresenta ideias para trabalhos futuros.

Capítulo 2

Otimização bio-inspirada

Grande parte dos problemas de otimização envolvem encontrar a melhor opção em um conjunto de possibilidades que cresce de maneira exponencial (KANN, 1992). Tais problemas são impossíveis de serem resolvidos de modo satisfatório com a tecnologia atual e necessitam de estratégias inteligentes para se aproximarem da solução ótima em tempo hábil. Dentre essas estratégias, destacam-se os algoritmos gulosos e os algoritmos de busca bio-inspirados.

Os algoritmos gulosos (CORMEN et al., 2009) são estratégias relativamente simples que, apesar de nem sempre obterem a solução ótima, normalmente aproximam-se bem dela. Tais métodos são geralmente utilizados quando apenas um critério é envolvido na otimização. Quando mais de um objetivo deve ser analisado, o problema fica bem mais complexo e essa abordagem deixa de ser indicada.

A otimização bio-inspirada (RAI; TYAGI, 2013) lança mão de estratégias baseadas na natureza para se encontrar boas soluções de forma eficiente. Assim como os algoritmos gulosos, não é garantida a obtenção da solução ótima, mas com uma boa modelagem do problema, é possível encontrar soluções suficientemente próximas. Na natureza, o processo de obter uma boa solução é usado a todo momento, desde a forma como as espécies evoluem, até a maneira como uma simples formiga encontra o caminho mais curto entre a colônia e uma fonte de comida. Ao observar processos comuns da natureza, estudiosos encontraram maneiras simples e eficientes de se resolver diversos problemas de otimização. Os principais métodos de otimização bio-inspirados na literatura são os algoritmos evolutivos e os algoritmos de inteligência coletiva (*swarm intelligence*). Dentre os evolutivos, destacam-se os algoritmos genéticos (AGs), e dentre os coletivos, destacam-se as colônias de formigas (ACOs) e os enxames de partículas (PSOs).

Os algoritmos genéticos se inspiram na teoria da evolução de Darwin (DARWIN, 1859). Na evolução das espécies, cada indivíduo possui um material genético que é cruzado com os genes de outro representante da espécie para gerar um novo indivíduo. Durante tal cruzamento, alguns indivíduos sofrem mutações aleatórias que podem alterar parte de sua carga genética e, com isso, suas características (fenótipo). O ambiente determina a

parte da população que sobrevive e a parte que perece. Os indivíduos sobreviventes, ou seja, aqueles bem adaptados ao ambiente, possuem maiores chances de se reproduzir e espalhar suas boas características genéticas. Desta forma, a evolução das espécies nada mais é que um processo otimizador, no qual indivíduos são gerados e os melhores são selecionados. Esse processo é repetido ao longo das gerações, até que se obtenha indivíduos bem adaptados ao ambiente em questão.

Os ACOs são inspirados no comportamento das formigas, organismos simples que quando analisados em conjunto (colônia) apresentam comportamento complexo (DORIGO; MANIEZZO; COLORNI, 1996). O interesse nas formigas vem da observação de que, ao buscarem por comida, acabam por encontrar o caminho mais rápido entre a fonte de alimento e o formigueiro. Como podem seres tão simples resolverem eficientemente um problema de otimização? Estudos revelaram que as formigas se baseiam no depósito de feromônios para se guiarem. Quanto mais forte o feromônio em um caminho, maior as chances dele ser utilizado pelas formigas. Tal comportamento serviu de inspiração para os algoritmos de otimização bio-inspirados, dando origem ao ACO.

Os algoritmos baseados em enxames de partículas (PSOs), assim como os ACOs são inspirados no comportamento emergente de populações de animais simples. Os PSOs se inspiram na navegação de pássaros, onde cada elemento da formação se guia através dos pássaros à frente (KENNEDY; EBERHART, 1995). O algoritmo se baseia na direção e velocidade de cada elemento do enxame, determinando a exploração do espaço de busca através de operações vetoriais. Portanto, é uma estratégia indicada para problemas contínuos, não sendo adequada para os problemas explorados nesta dissertação (embora existam exemplos de discretização dos PSOs na literatura).

Em geral, todo algoritmo de otimização bio-inspirado inicia sua busca por meio da geração aleatória de soluções. Após essa geração inicial, inicia-se uma etapa recorrente, na qual as melhores soluções guiam a construção de novas soluções que serão submetidas ao mesmo processo até que uma condição de parada seja atingida. Como não é necessário gerar todas as soluções possíveis, são métodos eficazes que, quando bem modelados, encontram um conjunto de boas soluções que resolvem o problema. Uma das principais diferenças entre os algoritmos bio-inspirados e as demais estratégias de otimização é o fato de que os primeiros produzem um conjunto de soluções aproximadas até o último passo do processo, quando, retorna a solução melhor avaliada. Essa é uma característica interessante para a otimização multiobjetivo, pois nela poder-se-ia retornar todas as soluções não-dominadas ao invés de apenas uma.

2.1 Algoritmos Genéticos

Os algoritmos genéticos (AGs) são métodos de busca baseados na ideia da seleção natural proposta por Charles Darwin (DARWIN, 1859). A teoria de Darwin, hoje já endossada

por diversas observações no campo da biologia, parte do princípio de que os organismos se adaptam ao ambiente em que vivem através de seleção natural, cruzamento e mutação. Mudanças nos genes (genótipo) de um indivíduo da população afetam suas características genéticas (fenótipo) e podem ajudá-lo a sobreviver em seu habitat ou atrapalhá-lo. Os indivíduos com características favoráveis têm maiores chances de sobreviver e reproduzir, passando essas características para a geração seguinte. Dessa forma, ao longo de milhões de anos, organismos simples se tornam complexos e extremamente adaptados ao meio.

A ideia dos algoritmos genéticos é aplicar o mesmo conceito da evolução natural na computação. O algoritmo parte de um conjunto de soluções aleatórias (população inicial) e, após várias iterações de seleção, cruzamento (ou *crossover*) e mutação, obtém um conjunto de soluções (população final) que espera-se que resolvam bem o problema. Nesse processo, o indivíduo na população representa uma solução, o meio representa o problema e as operações de cruzamento e mutação devem ser definidas, respectivamente, de forma a permitir a combinação de duas soluções e uma alteração aleatória em uma solução. A seleção dos pais e a sobrevivência dos mais aptos representam a metáfora da seleção natural.
Conforme ilustrado na Figura 1, o funcionamento básico de um AG inicia-se com a geração da população inicial. Nessa etapa, os indivíduos da população são normalmente gerados de forma aleatória e, em seguida, avaliados quanto à sua aptidão ao meio. No caso de um problema de otimização para uma função objetivo f, a aptidão de um indivíduo x é dada por $f(x)$. Após esse processo de inicialização, parte-se para a evolução das gerações que consiste na repetição das seguintes etapas:

1. Sortear os pares de pais;
2. Aplicar o cruzamento em cada par e gerar os filhos;
3. Aplicar a mutação aos filhos, de acordo com uma taxa de mutação pré-estabelecida;
4. Avaliar os filhos;
5. Selecionar entre pais e filhos quais indivíduos formarão a população da iteração seguinte (reinserção).

A evolução das gerações do AG termina quando uma condição de término estabelecida pelo usuário é atingida, por exemplo, encontrar a solução ótima do problema (quando conhecida) ou atingir um número máximo de gerações. Cada etapa de execução do AG é descrita com mais detalhes a seguir.

2.1.1 Representação do indivíduo

A principal dificuldade ao se elaborar um algoritmo genético é definir a representação do indivíduo. Cada indivíduo representa uma possível solução para o problema investi-

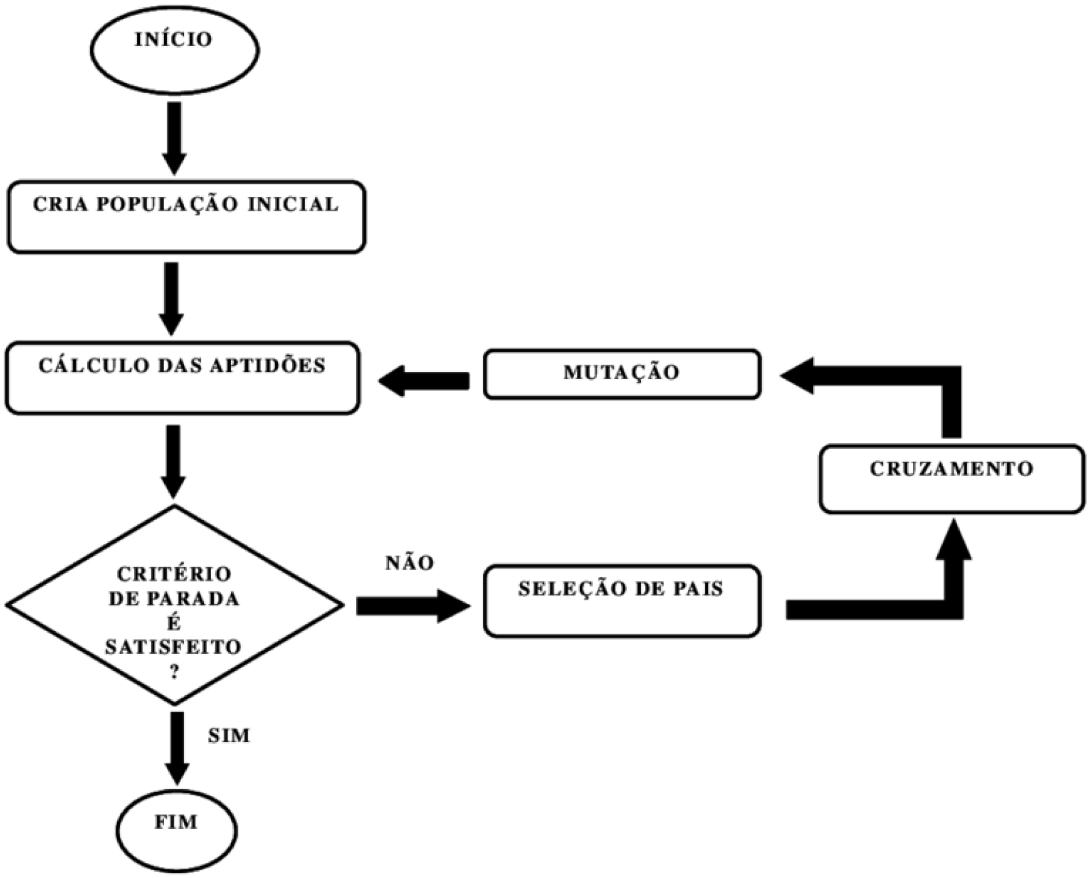

Figura 1 – Fluxograma de um AG. Retirado de (NÉIA et al., 2013)

gado e deve ser codificado em uma estrutura que possibilite e/ou favoreça a realização das operações genéticas de mutação e cruzamento. Na proposição original do AG (GOLDBERG, 1989), o indivíduo é representado de forma binária, ou seja, a solução para o problema é codificada em uma cadeia de bits, na qual cada posição pode facilmente ser invertida (mutação) ou copiada de um cromossomo para outro (cruzamento).

No problema da mochila 0/1, por exemplo, existe um conjunto de itens I com pesos e valores e uma mochila com capacidade limitada. Deve-se descobrir qual a melhor forma de se arranjar os itens de maneira que a soma dos valores de cada um seja máxima e que a capacidade da mochila não seja excedida. Para representar uma solução deste problema em um AG, basta assumir um vetor binário de tamanho igual a $|I|$, onde diz-se que o item está na mochila se sua posição correspondente no vetor binário é 1 e não está, caso contrário.

Outras representações de indivíduos também são possíveis, mas apresentam novos desafios. Por exemplo, em problemas de menores caminhos, normalmente trabalha-se com árvores e caminhos. No PRM, o indivíduo é uma árvore e tanto a mutação quanto o cruzamento devem ser operações em árvores.

Para elaborar a representação do indivíduo no AG, deve-se levar em consideração a

facilidade de manipulação da estrutura, a possibilidade de se introduzir um fator aleatório (mutação) e, principalmente, a representação das características de ambos os pais nos filhos. Se a estrutura não permite a herança de características, não é uma boa escolha para se utilizar em um algoritmo genético. Além disso, é preciso garantir a unicidade da forma de representação, ou seja, cada solução pode ser representada de uma única maneira e cada indivíduo deve representar uma única solução (relação um-para-um).

2.1.2 Seleção de pais

A estratégia para selecionar os pais que contribuirão para a composição genética da geração seguinte do AG deve ser escolhida de acordo com o propósito do algoritmo. Dependendo do problema, pode ser mais desejável uma convergência rápida que uma exploração mais profunda do espaço de busca, por exemplo. Os três métodos abaixo estão entre os mais empregados na etapa de seleção.

1. Seleção elitista: os $t_e\%$ indivíduos da população com melhor aptidão formam um grupo de onde se sorteiam todos os pares de pais necessários para gerar a população de filhos. t_e representa a taxa de elitismo, que é um parâmetro do AG. Dessa forma, não há chance de indivíduos muito ruins propagarem suas características. Portanto, a população converge para indivíduos iguais (ou muito parecidos) rapidamente, sem explorar partes do espaço de busca aparentemente ruins. Isso pode prejudicar o algoritmo em problemas com muitos mínimos locais, onde o fato de existir uma solução ruim não quer dizer que as soluções próximas também o são.

2. Roleta: é um método menos rigoroso que o anterior. Cada indivíduo recebe uma probabilidade de ser selecionado de acordo com sua aptidão. Os indivíduos com maior aptidão receberão probabilidade alta e dificilmente não serão selecionados, enquanto indivíduos muito ruins raramente se tornarão pais. Apesar disso, toda solução tem chance de ser escolhida, o que melhora a exploração do espaço de busca em relação a estratégia anterior, mas pode diminuir a velocidade de convergência. Esta estratégia é um meio-termo entre a seleção elitista e a seleção por torneio, explicada a seguir.

3. Seleção por torneio: esta é a estratégia com a menor pressão seletiva dentre os três métodos, ou seja, aquele que dá a maior chance aos indivíduos ruins de se tornarem pais e propagarem suas características. O torneio consiste em selecionar dois indivíduos aleatoriamente e escolher como primeiro pai aquele com a melhor aptidão, da mesma forma, sorteiam-se dois outros indivíduos na população, diferentes do primeiro pai, e define-se como segundo pai a solução com melhor aptidão. O torneio básico é o torneio de dois, mas para aumentar a pressão seletiva, é possível realizar o torneio com um maior número de representantes da população. Dentre as

três estratégias, esta é a que melhor explora o espaço de busca. Por outro lado, é possível que, ao explorar demasiadamente o espaço, o AG não consiga convergir.

2.1.3 Operadores genéticos

Nos algoritmos genéticos, destacam-se duas operações principais que atuam sobre o código do indivíduo: cruzamento e mutação. O cruzamento e a mutação estão diretamente ligados com a forma de representação do indivíduo.

Num algoritmo genético, cada iteração do laço principal é chamada de geração. No início de cada geração, sorteiam-se pares de pais de acordo com suas aptidões para que seja gerada uma nova população de filhos. A quantidade de pares de pais sorteados é determinada pela taxa de crossover, a qual é um parâmetro de configuração do AG. Cada par é composto de duas cadeias genéticas (cromossomos), uma correspondente a cada indivíduo do par. Para gerar o filho, cruzam-se os dois cromossomos a fim de se obter dois novos indivíduos que compartilhem características de ambos genitores. Esse processo é conhecido como cruzamento, ou *crossover*. Existem várias maneiras de se cruzar cadeias genéticas. A escolha depende principalmente da estrutura de dados usada para representar o cromossomo. A estrutura mais comum é a cadeia binária (GOLDBERG, 1989), onde uma solução é codificada em uma cadeia de 0's e 1's. Nesse caso, a forma mais simples de efetuar o cruzamento é gerar duas novas cadeias de bits (filhos), onde parte do material genético (posições na cadeia) pertence a um pai e o restante do material é proveniente do outro. Os filhos gerados em um cruzamento adotam materiais genéticos complementares de cada pai. Por exemplo, se o filho 1 herda os n primeiros genes do pai 1 e o restante do pai 2, o filho 2 herdará os n primeiros genes do pai 2 e o restante do pai 1. A Figura 2 ilustra este tipo de cruzamento. Existem diversas formas de se combinar duas cadeias de bits, dentre elas podem-se destacar:

- Ponto de cruzamento único: uma posição i de uma das cadeias é sorteada. O filho 1 herda os i primeiros genes do pai 1 e restante do pai 2, enquanto que o filho 2 adota o material genético complementar. Na Figura 2, é ilustrado um exemplo onde o ponto de cruzamento é estabelecido na metade.

- Dois pontos de cruzamento: ao invés de se utilizar apenas um ponto para dividir o material genético, este método divide a cadeia binária em três partes. Nesse caso, um filho herda a primeira e terceira partes de um pai e a segunda do outro.

- Cruzamento uniforme: cada bit do filho é obtido de forma aleatória, pode vir tanto do pai 1 quanto do pai 2. Em termos de implementação, sorteia-se uma máscara binária e para cada bit 0 da máscara, copia-se o gene do pai 1 para o filho. Para cada bit 1, copia-se o gene do pai 2.

- ❏ Cruzamento aritmético: realiza-se uma operação binária (ex: AND, OR, XOR, etc.) entre os cromossomos do pai 1 e do pai 2.

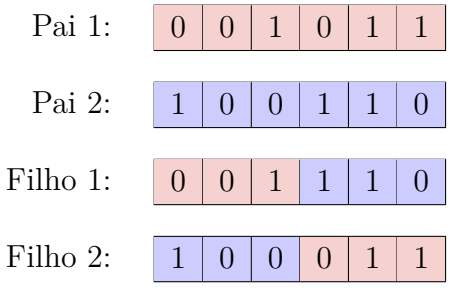

Figura 2 – Exemplo de *crossover* com cruzamento de ponto único, onde o ponto de cruzamento está na metade do material genético.

Esses operadores genéticos foram propostos inicialmente para indivíduos binários. Para outros tipos de representação de indivíduos (não binários), foram necessários outros tipos de cruzamento mais apropriados a essas representações (número reais, números inteiros, árvores, etc).

A fim de melhorar a diversidade entre as soluções avaliadas, uma pertubação nos genes dos indivíduos é gerada durante a busca, possibilitando a exploração de elementos ausentes na população atual. Para isso, após gerar o cromossomo de cada filho, é preciso permitir que ocorra uma mutação, ou seja, uma mudança aleatória no material genético. A chance de uma mutação ocorrer depende de um parâmetro do AG chamado de "taxa de mutação". O processo de alteração genética aleatória depende da representação do indivíduo. No caso de uma cadeia binária, por exemplo, é simples, basta sortear um bit e invertê-lo. Em árvores, a mutação pode consistir na eliminação de algum vértice e na reconexão aleatória. Em grafos ponderados, alterar os pesos de uma aresta pode ser uma boa ideia. A definição do processo de mutação dependerá do problema e pode ser feita de várias maneiras diferentes, desde que produza uma pequena diferença no indivíduo e que ele continue representando uma solução válida.

O cruzamento normalmente gera um ou dois filhos para cada pai. Em todos os métodos descritos anteriormente é possível obter um segundo filho com o material genético não utilizado.

2.1.4 Reinserção da População

Independente do número de filhos gerados, após os processos de *crossover* e mutação, a população será maior que o limite máximo permitido, exigindo a eliminação de alguns indivíduos (seleção natural). A reinserção opera sobre a aptidão (função de avaliação ou *fitness*) de cada indivíduo da população. Diversas estratégias podem ser aplicadas nesse processo de seleção, também conhecido como reinserção. Por exemplo, a seleção

natural sem elitismo, simplesmente elimina a população mais velha (pais). A seleção natural com elitismo mantém um percentual da população de pais (elite), completando-a com os melhores filhos. Outro tipo de elitismo é a seleção dos melhores indivíduos. Essa estratégia é normalmente mais interessante, pois permite a sobrevivência dos indivíduos mais aptos considerando a totalidade da população, sendo assim, avaliam-se todos os pais e filhos e mantêm-se aqueles com melhor aptidão.

2.2 Otimização por Colônia de formigas

A otimização por colônia de formigas (ACO), proposta por Dorigo, Maniezzo e Colorni (1996), é um modelo de busca bio-inspirado que parte da ideia de que estruturas simples, com alguma espécie de comunicação, podem gerar um comportamento complexo quando operam em conjunto, de forma cooperativa. A inspiração do ACO é o forrageamento em uma colônia de formigas. Na natureza, observa-se que as formigas, mesmo sendo seres vivos simples, conseguem encontrar o melhor caminho entre o formigueiro e a fonte de comida. A partir do estudo desse comportamento, descobriu-se que tal faceta é possível através de uma comunicação indireta entre os animais. Ao fazer um caminho, as formigas depositam uma substância chamada feromônio, a qual pode ser percebida por outros membros da espécie. Uma formiga ao decidir qual caminho percorrer, tem maior chance de escolher aquele com a maior quantidade de feromônios. Além disso, a substância evapora com o tempo. Dessa forma, quanto menor o caminho, maior a frequência com a qual as formigas depositarão feromônio e, portanto, maior será a chance de ser escolhido.

Ao trazer o conceito de colônia de formigas para a computação, observa-se um grande potencial para se resolver problemas de busca em grafo. Por exemplo, para descobrir o menor caminho entre os vértices A e B em um grafo não ponderado G, basta simular várias formigas que partem de A e chegam em B, fazendo um caminho baseado na quantidade de feromônios das arestas. Ao final de cada iteração, atualiza-se o valor do feromônio em cada aresta de acordo com a evaporação e com as arestas percorridas pelas formigas. Dessa forma, espera-se que, após várias iterações, a quantidade de feromônios seja suficiente para guiar uma formiga pelo melhor caminho entre A e B. O fluxograma de um ACO é mostrado na Figura 3.

O algoritmo inicia com a limpeza dos feromônios das arestas, ou seja, é atribuído zero para a quantidade de feromônio depositado em cada aresta do grafo. Após essa limpeza, inicia-se o processo iterativo do algoritmo onde, a cada iteração, as formigas na colônia percorrem o caminho do vértice de partida (formigueiro) ao nó destino (fonte de comida). Dependendo da formulação do problema, o caminho de volta também pode ser realizado. Ao final de cada iteração, atualizam-se os feromônios em cada aresta. O algoritmo termina quando uma condição de parada pré-determinada é atingida, normalmente um número máximo de iterações. As etapas principais da execução de um ACO são descritas a seguir.

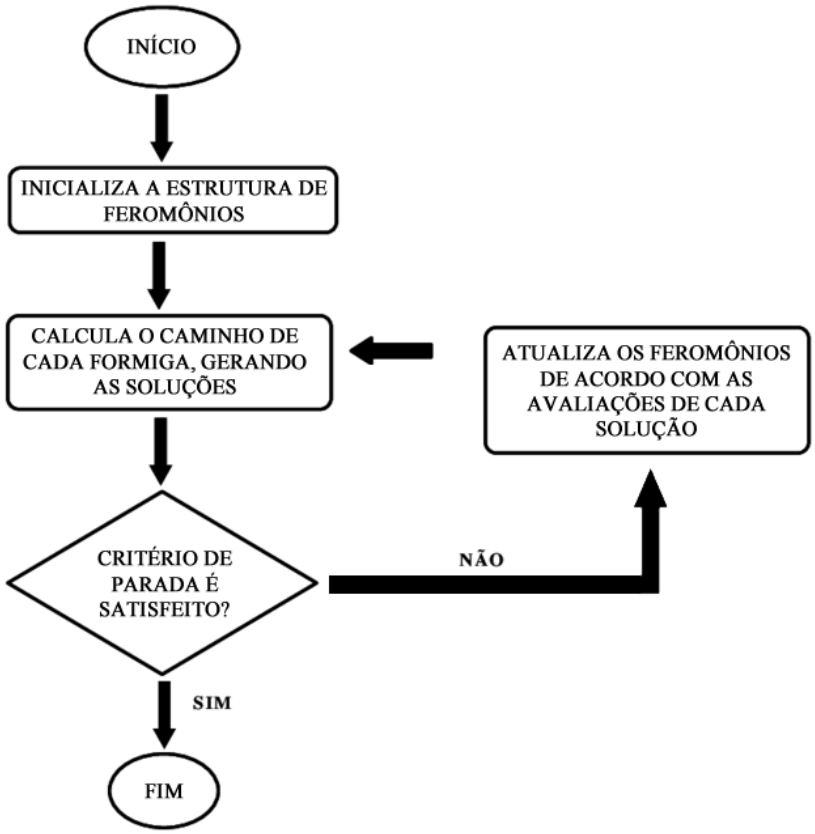

Figura 3 – Fluxograma de um ACO.

2.2.1 Representação da solução

No ACO, a solução é representada pelo caminho feito pela formiga no grafo, normalmente uma lista de vértices, uma árvore ou um subgrafo do grafo de entrada. Por exemplo, no problema de encontrar o menor caminho, a solução pode ser uma lista de vértices que representa o percurso desse caminho. Caso seja necessário encontrar caminhos entre a raiz e diversos destinos, pode-se representar a solução como uma árvore. Nem sempre é fácil decidir a codificação de uma solução. No problema da mochila, por exemplo, não é trivial a visualização de um grafo no processo da construção da solução. Nesse caso, uma possível representação é a associação de feromônios diretamente aos itens, através de um vetor de feromônios ao invés de uma matriz.

2.2.2 Construção da solução

Independentemente da representação escolhida, deve ser possível relacionar cada parte da solução com uma quantidade de feromônios na estrutura principal. Por exemplo, no caso de grafos, as arestas escolhidas para montar a solução devem ter seus feromônios incrementados a fim de guiar as próximas formigas. Em cada época (iteração do algoritmo) um número pré-determinado de soluções é construído, onde cada formiga decide quais

partes serão adotadas em sua solução, com base nos respectivos valores de feromônio.

Além dos feromônios, as formigas ainda utilizam as informações de heurística para decidir o próximo passo. Uma heurística é uma função que estima a qualidade do caminho e normalmente representa o peso de uma aresta. Estando em um vértice i de um grafo G, uma formiga tem probabilidade $p(i, j)$ de escolher a aresta que leva ao vértice adjacente j. Essa probabilidade é calculada pela seguinte equação:

$$p(i,j) = \frac{\tau_{i,j}^{\alpha} \times \eta_{i,j}^{\beta}}{\sum_{v \in adj(i)} \tau_{i,v}^{\alpha} \times \eta_{i,v}^{\beta}} \tag{1}$$

Sendo:

- $\tau_{i,j}^{\alpha}$: feromônio na aresta (i, j) elevado à constante α, que representa a importância atribuída ao valor do feromônio. Representa o conhecimento sobre o ambiente (busca global).
- $\eta_{i,j}^{\beta}$: heurística da aresta (i, j) elevada à constante β, que representa a importância atribuída à heurística. A heurística de uma aresta é dada em função do peso, como, por exemplo, $1/peso$, normalmente usado em problemas de minimização. Representa a visibilidade da formiga (busca local).
- $adj(i)$: são todos os vértices adjacentes a i, ou seja, todo vértice em G para o qual é possível construir um caminho a partir de i com apenas uma aresta.

O número de iterações (épocas), a quantidade de soluções geradas por iteração e as constantes α e β são parâmetros de configuração de um algoritmo ACO. De forma geral, dado um grafo G e um nó inicial, o processo de construção da solução sempre verifica todos os movimentos possíveis para a formiga, tomando sua decisão de acordo com os feromônios e as heurísticas de cada uma das possibilidades.

2.2.3 Atualização dos feromônios

Existem duas ocasiões onde o feromônio de uma aresta pode ser atualizado: no momento em que a formiga passa pela aresta e no fim de cada iteração. A maioria das implementações considera apenas o segundo caso, pois assim, é possível avaliar as soluções e incrementar os feromônios de acordo com os desempenhos obtidos. Ao fim de cada época, a quantidade de feromônio existente na aresta que liga os vértices i e j ($\tau_{i,j}$) é atualizada por:

$$\tau_{i,j} = (1 - \rho) \times \tau_{i,j} + \sum_{k \in formigas} \Delta\tau_{i,j}(k) \tag{2}$$

Sendo:

- ρ: parâmetro de configuração do ACO que corresponde ao coeficiente de evaporação. Determina o quão rápido o feromônio deve desaparecer das arestas após depositado.
- $formigas$: conjunto de todas as formigas na iteração.
- $\Delta\tau_{i,j}(k)$: quantidade de feromônio depositada pela formiga k na aresta entre os vértices i e j.

A quantidade de feromônio depositada por uma formiga k em uma aresta entre os vértices i e j é dada por:

$$\Delta\tau_{i,j}(k) = \begin{cases} \frac{Q}{L_k}, & \text{se } x \geq 1 \\ 0, & \text{caso contrário} \end{cases} \tag{3}$$

Sendo:

- Q: Quantidade máxima de feromônio que pode ser depositada por uma formiga.
- L_k: Custo da solução gerada pela formiga k.

Considerando essas características, nossa pesquisa propõe uma nova versão de ACO para tratar problemas discretos com enfoque em otimização multiobjetivo.

Capítulo 3

Otimização multiobjetivo

A otimização multiobjetivo consiste em selecionar as melhores soluções de acordo com múltiplos critérios. Por exemplo, ao estabelecer o melhor caminho entre duas cidades pode-se não estar interessado apenas na menor distância, mas também no tráfego, segurança das vias, quantidade de pedágios, etc. Na otimização com um único objetivo, a comparação entre soluções é relativamente simples. Para que uma solução seja considerada melhor que a outra, basta que ela tenha uma melhor avaliação segundo o critério/métrica considerada. Por outro lado, quando se trabalha com mais de uma função de otimização, é preciso adotar alguma estratégia mais elaborada. Por exemplo, pode ser possível aplicar algum tipo de ponderação entre os valores dos diferentes objetivos, combinando-se as funções em um único objetivo. De forma alternativa, existem diversas abordagens que mantêm os objetivos separados e trabalham com o conceito de dominância de Pareto, como os algoritmos NSGA-II (DEB et al., 2002a) e SPEA2 (ZITZLER; LAUMANNS; THIELE, 2001).

A dominância de Pareto estabelece que uma solução A é melhor que uma solução B, ou A domina B ($A \prec B$), se, e somente se:

- A é melhor avaliado que B em pelo menos um dos objetivos;
- A não tem avaliação pior que B em nenhum dos objetivos.

Considerando-se um problema de minimização e F como o conjunto de funções objetivo, tem-se, matematicamente:

$$A \prec B \Leftrightarrow (\forall (f \in F) f(A) \leq f(B)) \wedge (\exists (f \in F) f(A) < f(B)) \tag{4}$$

Em problemas de otimização multiobjetivo, geralmente o objetivo está em encontrar o conjunto Pareto ótimo, ou seja, o conjunto de todas as soluções do espaço de busca que não são dominadas por nenhuma outra e cujo mapeamento para o espaço de objetivos forma a Fronteira de Pareto. Graficamente, a fronteira de Pareto representa a linha formada pelas soluções não-dominadas existentes para o problema. Na Figura 4 apresenta-se um

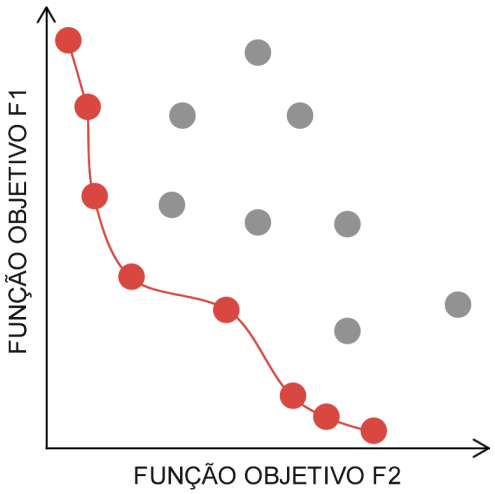

Figura 4 – Fronteira de Pareto

exemplo da fronteira de Pareto para um problema de minimização com dois objetivos ($F1$ e $F2$). Nesse exemplo, os pontos pertencentes à fronteira de Pareto estão destacados em vermelho. Observe que, todas as soluções pertencentes à fronteira de Pareto são não dominadas, ou seja, nenhum ponto da fronteira possui ambos valores menores que alguma outra solução em ambas as funções ($F1$ e $F2$). Em contrapartida, toda solução acima da fronteira (pontos em cinza) é dominada, pois existe alguma solução em vermelho que possui ambos valores de F1 e F2 menores.

Não existe limite para o número de funções objetivo em um problema de otimização, mas quanto maior a quantidade de objetivos, mais complexa é a busca (DEB; JAIN, 2014). Os algoritmos clássicos de otimização multiobjetivo *Non-Dominated Sorting Genetic Algorithm II* (NSGA-II) e *Strength Pareto Evolutionary Algorithm 2* (SPEA2) lidam bem com até três objetivos, mas a partir de quatro critérios de otimização, ambos os métodos sofrem para encontrar soluções que fazem parte do conjunto Pareto ótimo do problema. Desta forma, criou-se a classificação "*many-objective*". Problemas *many-objective* (4 ou mais objetivos) apresentam um maior grau de dificuldade e demandam novas técnicas para que sejam resolvidos eficientemente. Analisando as abordagens baseadas em algoritmos evolutivos multiobjetivos (AEMOs), Deb e Jain (2014) observaram os seguintes os problemas trazidos pelo alto número de objetivos:

1. **Grande parte da população é não dominada:** a maioria dos algoritmos multiobjetivos classifica a população de acordo com a dominância de Pareto. Se existem muitas funções objetivo, é muito comum que uma solução seja melhor que outra em pelo menos uma delas. Desta forma, a maior parte das soluções se torna não-dominada, o que impede os algoritmos de evoluírem a população, já que todos os indivíduos são considerados igualmente bons.

2. **Avaliar a diversidade da população se torna computacionalmente caro:**

a fim de garantir uma boa diversidade populacional, os algoritmos adotam alguma medida de distância entre as soluções e removem aquelas consideradas mais similares. O aumento na dimensionalidade traz consequentemente um maior impacto (complexidade e custo) no cálculo da proximidade entre os indivíduos.

3. ***Crossover* ineficiente:** a alta dimensionalidade do espaço de busca faz com que os indivíduos na população sejam muito distantes uns dos outros e, normalmente, o cruzamento entre duas soluções muito diferentes resultam num filho muito distante dos pais, o que prejudica a convergência da busca. Portanto, pode ser necessário redefinir os operadores de recombinação a fim de restringir as possibilidades de pareamento.

4. **População demasiadamente grande:** quanto maior o número de objetivos, maior o número de soluções na fronteira de Pareto. Portanto, para se obter bons resultados, é necessário que se manipule grandes populações de indivíduos, o que é computacionalmente caro e dificulta a análise do usuário que deverá escolher uma única solução ao final do processo.

5. **Métricas de análise de desempenho do algoritmo se tornam difíceis de calcular:** a avaliação do resultado do algoritmo (taxa de erro, distância para o Pareto, hiper-volume, etc.) está diretamente relacionada ao número de objetivos. Quanto maior ele for, maior será o esforço computacional necessário. A complexidade do hiper-volume, por exemplo, cresce exponencialmente com o número de objetivos.

6. **Dificuldade de visualização:** é fácil representar graficamente as soluções e a fronteira de Pareto em problemas de até três objetivos. Entretanto, não existe uma forma eficiente de visualizar os resultados para problemas que lidam com maior dimensionalidade (4 objetivos em diante).

A maior parte dos algoritmos *many-objectives* mencionados neste trabalho buscam lidar com os quatro primeiros problemas. As duas últimas não se relacionam diretamente aos algoritmos de otimização em si, mas sim à análise da qualidade das soluções encontradas.

3.1 Algoritmos Bio-Inspirados Multiobjetivos

A maior parte dos métodos de busca multiobjetivo é baseada em algoritmos inspirados na biologia. Esses algoritmos são adaptações das estratégias evolutivas, colônias de formigas e enxames de partículas tradicionais para lidar com problemas que envolvam mais de um objetivo. A principal diferença entre um algoritmo de otimização tradicional e sua

variação multiobjetivo está na forma de cálculo da aptidão dos indivíduos da população. No caso dos algoritmos multiobjetivos, geralmente o conceito de dominância é usado de diferentes formas (BUENO; OLIVEIRA, 2010).

A seguir são apresentados alguns algoritmos bio-inspirados multiobjetivo encontrados na literatura. Os dois primeiros (NSGA-II e SPEA2) são algoritmos bem conhecidos e largamente utilizados em problemas multiobjetivos. Apesar de sua eficiência na resolução de problemas com até 3 objetivos, o desempenho desses AEMOs costuma cair consideravelmente com o aumento no número de objetivos (FRANÇA et al., 2017). Os demais algoritmos apresentados, são mais recentes e foram concebidos especificamente para tratar de problemas *many-objective*, ou seja, que envolvam 4 ou mais objetivos.

3.1.1 NSGA-II

O *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002a) é o algoritmo evolutivo multiobjetivo mais frequente na literatura. O processo desse algoritmo é semelhante ao do algoritmo genético comum, com diferença no cálculo de aptidão, que é feito por *ranks*, e no cálculo de distâncias, que é inexistente na proposta original do AG. A atribuição de aptidão (*fitness*) se dá pela classificação da população em categorias/classes (*rankings*) de dominância (fronteiras), de forma que o primeiro contenha todas as soluções não dominadas, o segundo todos os indivíduos não-dominados excluindo aqueles presentes na primeira fronteira, e assim por diante. Quanto melhor o *ranking* de uma solução, melhor sua aptidão e maior sua chance de sobreviver para a próxima geração. Várias soluções podem pertencer a uma mesma fronteira. A fim de diferenciá-las entre si, é utilizado um cálculo de distância (*crowding distance*), o qual confere melhor avaliação às soluções mais esparsas de uma mesma fronteira (maior distância em relação às outras soluções da fronteira), garantindo assim a diversidade da população.

O fluxo de funcionamento do NSGA-II inicia-se com a geração aleatória dos indivíduos. Em seguida, a população é classificada em fronteiras de dominância (ou *ranks*) e inicia-se o processo iterativo, o qual termina assim que a condição de parada é atingida. As etapas que compõem cada iteração do NSGA-II são descritas no pseudo-código do algoritmo 1.

Algoritmo 1 Processo iterativo do NSGA-II

1: **enquanto** critério de parada não for atingido **faça**
2: $Pais \leftarrow selecao(Pop_{atual}, TX_{cross})$
3: $Filhos \leftarrow crossover(Pais)$
4: $Pop_{nova} \leftarrow Pop_{atual} \cup Filhos$
5: $ranks \leftarrow ranking(P_{nova})$
6: $crowdingDistance(P_{nova})$
7: $Pop_{atual} \leftarrow reinsercao(ranks, tam_{populacao})$
8: **fim enquanto**

O processo iterativo do algoritmo inicia-se com a seleção dos pares de pais para o

cruzamento (linha 2). Na implementação considerada, a seleção de pais utiliza torneio simples com *tour* de 2, ou seja, dois elementos da população são escolhidos de forma aleatória e o indivíduo com melhor avaliação é selecionado como um dos pais. Então, esse processo é repetido para a escolha do segundo pai.

Na linha 3 do pseudo-código, através dos pares de pais, geram-se os filhos com o *crossover* e a mutação. Após a geração dos filhos, a população corrente e o conjunto de filhos são concatenados (linha 4) e submetidos à classificação em fronteiras de dominância (*ranks*) na linha 5).

A classificação em fronteiras de dominância recebe um conjunto de soluções e verifica quais dentre elas não são dominadas. O conjunto de soluções não-dominadas forma a primeira fronteira de dominância. Do conjunto restante (excluindo a primeira fronteira), retiram-se as soluções não dominadas para formar a segunda fronteira. Esse processo se repete até que todos os indivíduos tenham sido classificados.

Após toda a população ter sido classificada em fronteiras, na linha 6, o algoritmo calcula a distância de aglomeração (*crowding distance*) dos indivíduos em cada fronteira de dominância. O cálculo da distância, considerando cada objetivo avaliado, ordena o conjunto de soluções e faz uma relação entre as distâncias de cada indivíduo para os vizinhos adjacentes (que estão imediatamente antes ou depois). A distância de aglomeração de cada indivíduo é resultado da somatória das distâncias resultantes em cada objetivo. Quanto maior o valor, maior é a diferença entre o indivíduo e seus pares dentro do *rank*. Essa métrica é usada para manter a diversidade das soluções entre as gerações do algoritmo.

Com toda a população classificada em fronteiras (ou *ranks*) e todas as distâncias calculadas, na linha 7, uma nova população é escolhida com base nos melhores indivíduos entre pais e filhos. Para isso, analisa-se fronteira a fronteira, da melhor para a pior, até que o tamanho máximo da população seja atingido. Para cada fronteira aplica-se o seguinte processo de decisão:

- Se $tamanho(fronteira) + tamanho(novaPopulação) < tamPop$: adicionam-se todos os membros da fronteira à nova população.
- Caso contrário: adicionam-se à nova população os $(tamPop - tamanho(novaPopulação))$ elementos da fronteira com os maiores valores de distância.

Desta forma, ao final do algoritmo obtém-se a fronteira de Pareto aproximada na primeira fronteira da população gerada na última iteração do algoritmo.

3.1.2 SPEA2

O *Strength Pareto evolutionary algorithm 2* (SPEA2) (ZITZLER; LAUMANNS; THIELE, 2001) é um AEMO que calcula, para cada membro da população, sua força (*strength*)

e densidade. A força de uma solução é dada pelo número de indivíduos que ela domina, enquanto a densidade é uma medida de distância para os vizinhos mais próximos, quanto maior a densidade mais próximo o indivíduo está das demais soluções. A aptidão (*fitness*) de uma solução é definida por sua densidade mais a soma das forças de todo indivíduo que a domina.

Os indivíduos mais aptos no SPEA2 são aqueles dominados pela menor quantidade de soluções e que possuem maior variabilidade genética. O algoritmo calcula a aptidão em três etapas: cálculo de força (*strength*), da aptidão crua (*raw fitness*) e da densidade (ZITZLER; LAUMANNS; THIELE, 2001).

A força de um indivíduo i ($s(i)$) é o número de soluções que ele domina, ou seja, considerando A o arquivo e P a população:

$$s(i) = |j| : j \in P \cup A \wedge i \prec j \tag{5}$$

Uma vez calculada a força de cada indivíduo, parte-se para o cálculo do *raw fitness*. O *raw fitness* de um indivíduo i ($r(i)$) é dado pela soma das forças de cada elemento que o domina, como segue:

$$r(i) = \sum_{j \in A \cup P | j \prec i} s(j) \tag{6}$$

Observe que, caso o indivíduo seja não-dominado, seu *raw fitness* será o menor possível (zero). Para finalizar o cálculo de aptidão, é definida a densidade de cada indivíduo ($d(i)$). Essa densidade é computada de acordo com a distância da solução para seus vizinhos e é dada por:

$$d(i) = \frac{1}{\sigma_i^k + 2} \tag{7}$$

Sendo, σ_i^k a k-ésima menor distância entre o indivíduo i e o restante da população; e k a raiz quadrada do tamanho do conjunto de soluções em avaliação, i.e. $k = \sqrt{|P \cup A|}$. O valor de $d(i)$ sempre está no intervalo (0,1).

Finalmente, a aptidão do indivíduo ($f(i)$) é dada pela soma do *raw fitness* e a densidade:

$$f(i) = r(i) + d(i) \tag{8}$$

Note que, se a solução i é não-dominada, $f(i) < 1$. Isso ocorre dado que $d(i) < 1$ para qualquer solução e, quando i é não-dominada, $r(i) = 0$.

Além do cálculo de aptidão, outra diferença importante entre o SPEA2 e um AG tradicional é a utilização de uma população extra, denominada arquivo. O arquivo é responsável por guardar as melhores soluções já encontradas até o momento, funciona como uma espécie de elitismo. Na seleção para o cruzamento, os pais são sempre escolhidos do arquivo e os filhos substituem 100% da população corrente. A cada iteração, os melhores

indivíduos entre a população e o arquivo compõem o arquivo da geração seguinte. A quantidade de indivíduos no repositório de soluções não-dominadas é limitada e, quando esse tamanho máximo (tam_{arq}) é excedido, deve-se executar um processo de truncamento.

O processo de truncamento de um arquivo ocorre na seleção natural (reinserção), que é a última função executada na iteração do laço principal de um AG. No SPEA2, a reinserção consiste em definir o arquivo para a próxima geração a partir das populações. Nesse processo, extraem-se do conjunto total de soluções (população e arquivo) aquelas que não são dominadas por nenhuma outra, e com esse subconjunto (n_d) constrói-se o novo arquivo através do seguinte processo de decisão:

- Se $tamanho(n_d) = tam_{arq}$, o novo arquivo é formado por n_d;
- Se $tamanho(n_d) < tam_{arq}$, o novo arquivo é formado pela união de n_d com os $tam_{arq} - tamanho(n_d)$ indivíduos restantes com melhor aptidão;
- Caso contrário, ($tamanho(n_d) > tam_{arq}$), o novo arquivo é formado por n_d e deve-se truncá-lo em ($tamanho(n_d) - tam_{arq}$) passos, onde em cada passo, elimina-se o indivíduo com menor variabilidade genética em relação aos demais.

O algoritmo retorna como resposta para o problema o arquivo resultante da última geração computada. Espera-se que, após as diversas iterações, o algoritmo tenha conseguido uma boa aproximação da fronteira de Pareto.

3.1.3 MOEA/D

O *Multiobjective evolutionary algorithm based on decomposition* (MOEA/D) (ZHANG; LI, 2007) é um algoritmo que avalia os objetivos através de uma função escalarizadora, baseando-se na dominância de Pareto apenas para atualizar o conjunto de soluções não dominadas geradas em cada iteração. A esse conjunto, dá-se o nome de arquivo. No MOEA/D, um problema multiobjetivo é decomposto em múltiplos problemas de um único objetivo. Cada problema mono-objetivo é associado a uma célula da estrutura que armazena a população. Cada célula é definida por um vetor de pesos gerado aleatoriamente e representa um indivíduo, ou seja, o número de células é igual ao tamanho da população. Além dos pesos, a célula é composta de uma solução e uma vizinhança. A vizinhança é formada pelos k indivíduos mais próximos de acordo com o vetor de pesos, onde k é um parâmetro do algoritmo que representa o tamanho das vizinhanças. A aptidão (*fitness*) de uma solução é calculada de acordo com sua avaliação em cada objetivo, a função escalarizadora, e o vetor de pesos da célula. Em toda geração, uma nova solução é gerada para cada célula, onde a vizinhança é considerada nas etapas de escolha dos pais e seleção natural.

O início do algoritmo consiste na geração da estrutura de células e na definição das vizinhanças. Para isso, sorteiam-se os vetores de pesos (a soma de cada vetor deve ser

igual a um) e, para cada um deles, calculam-se os k vetores mais próximos (vizinhança). Essa estrutura é imutável e é utilizada no decorrer de todo o algoritmo. A geração dos vetores de pesos pode ser tanto aleatória quanto seguir uma distribuição pré-definida (por exemplo, uma distribuição uniforme).

Uma parte fundamental do MOEA/D é a escolha da função escalarizadora, a qual é a principal responsável pelo cálculo de aptidão. Em todos experimentos realizados neste trabalho, foi utilizada a soma ponderada, mas outras estratégias, como *Penalty-Based Boundary Intersection* (PBI) e decomposição de Tchebycheff também podem ser utilizadas (ZHANG; LI, 2007). A escolha da soma ponderada se baseou em experimentos preliminares que mostraram sua maior eficiência em relação à decomposição de Tchebycheff para os problemas avaliados. Além disso, é mostrado em Ishibuchi, Akedo e Nojima (2013) que, para muitos objetivos, as funções de soma ponderada e PBI apresentam melhores resultados para o problema da mochila.

A aptidão de uma solução é calculada através da função escalarizadora e do vetor de pesos. Por exemplo, se os valores $[2; 9; 5]$ representam a solução s no espaço de objetivos, $[0,3; 0,2; 0,5]$ é o vetor de pesos da célula c, e a soma ponderada é a função escalarizadora, então a aptidão de s em c é dada por $2 \times 0,3 + 9 \times 0,2 + 5 \times 0,5 = 4,9$.

Antes de começar o processo iterativo (evolução das gerações), gera-se aleatoriamente uma solução para cada célula e calcula-se a sua Aptidão, de acordo com o vetor de pesos associado a cada célula. Além disso, todas as soluções geradas aleatoriamente para as células (a estrutura completa) são analisadas de acordo com o critério da Dominância de Pareto para identificar-se o conjunto de soluções não-dominadas. Essas soluções são utilizadas para iniciar o conjunto de soluções não dominadas armazenadas no arquivo externo. Esse arquivo pode sofrer alterações a cada geração, dependendo se as novas soluções criadas dominam aquelas no arquivo. Após a última geração, esse arquivo externo com as soluções não dominadas é retornado como solução do algoritmo.

Como em qualquer outro AG, o processo evolutivo do MOEA/D consiste basicamente da seleção de pais, da geração dos filhos e sua reinserção na população. Para cada célula c_i, dois pais são selecionados aleatoriamente em sua vizinhança. Quando um filho é gerado para uma célula c_i, sua aptidão é calculada para cada uma das células na vizinhança v de c_i, substituindo a solução atual da primeira célula em v onde o *fitness* do novo indivíduo (filho) é melhor que o da solução corrente. Ao final de cada iteração (geração), atualiza-se o arquivo com as novas soluções não-dominadas. As Figuras 5, 6 e 7 ilustram um exemplo de aplicação simples do MOEA/D em um problema com 2 objetivos. Na Figura 5 são criados 6 vetores de pesos $(v_1, v_2, ..., v_6)$ distribuídos uniformemente que representam cada um uma célula. Nesse exemplo, o tamanho da população é 6. Na Figura 6, é gerada uma solução aleatória para cada célula $(s_1, s_2, ..., s_6)$. Uma vez gerada a população, inicia-se o processo evolutivo (iterativo), no qual o algoritmo passa por todas as células em cada geração. O retângulo menor (mais interno) indica a célula que está

sendo tratada (v_1). O retângulo maior (mais externo) representa a vizinhança da célula corrente. Nesse caso, a vizinhança possui tamanho 2 e, portanto, os vizinhos de v_1 são v_2 e v_3, que são os vetores de pesos mais próximos de v_1. Duas soluções são sorteadas aleatoriamente na vizinhança para serem os pais e, através do *crossover*, geram uma nova solução filha (s_c). Na Figura 6, foram sorteadas como pais as soluções s_1 e s_3. A Figura 7 mostra o processo de seleção, onde a nova solução é comparada com cada membro da vizinhança através da função escalarizadora (média ponderada) e do vetor de pesos das células. s_c é submetida à média ponderada (f) de seus valores no espaço de objetivos três vezes, uma para os pesos v_1, outra para v_2 e outra para v_3. A primeira célula em que a média de s_c é menor (problema de minimização) que a média do indivíduo que já está na célula é atualizada com s_c. No exemplo, v_2 é a primeira célula onde o filho tem melhor desempenho que a solução corrente. Portanto a solução em v_2 deixa de ser s_2 e passa a ser s_c. Após o tratamento da primeira célula v1 (primeira iteração), o conjunto de células tem a configuração mostrada na Figura 8. A segunda iteração trabalha com a célula v_2, que possui a mesma vizinhança anterior, v_1, v_2 e v_3. A iteração seguinte trabalha com os vetores v_2, v_3 e v_4, e assim por diante, até a última iteração que trabalha com a vizinhança v_4, v_5 e v_6. Após passar por todas as células, a geração termina atualizando o arquivo externo com todas as soluções não dominadas obtidas na geração corrente. Ao final do processo, quando todas as gerações foram calculadas, espera-se que o arquivo contenha uma boa aproximação da fronteira de Pareto.

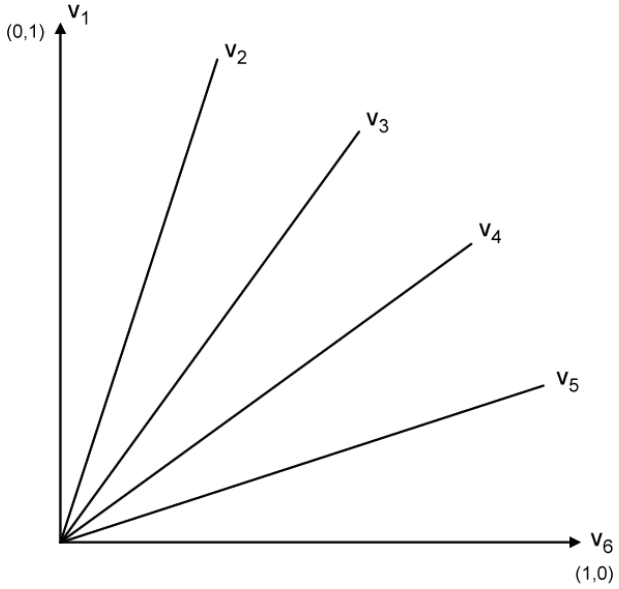

Figura 5 – Exemplo de aplicação do MOEA/D: geração dos vetores de peso

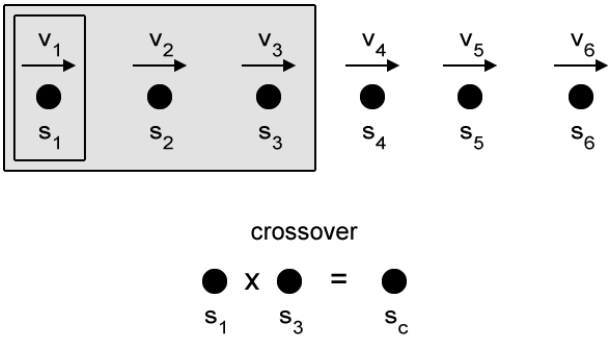

Figura 6 – Exemplo de aplicação do MOEA/D: geração de soluções

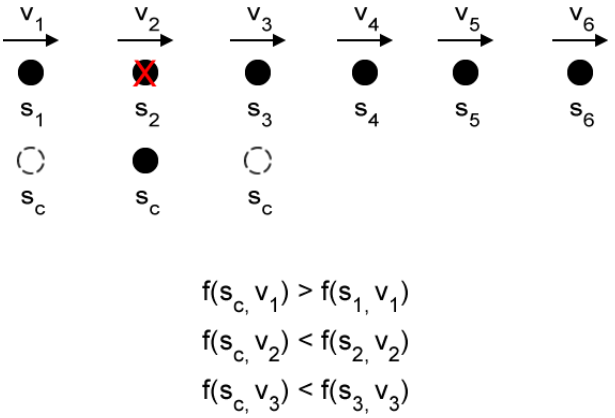

Figura 7 – Exemplo de aplicação do MOEA/D: seleção de soluções

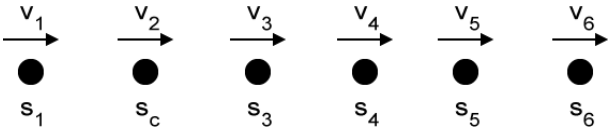

Figura 8 – Exemplo de aplicação do MOEA/D: configuração da estrutura após a primeira iteração

3.1.4 NSGA-III

O *Non-dominated Sorting Genetic Algorithm III* (NSGA-III) (DEB; JAIN, 2014) é uma extensão do NSGA-II, elaborada com o objetivo de prover um melhor desempenho do *framework* ao lidar com problemas de mais de três objetivos (*many-objective*). Ele se diferencia da versão anterior principalmente na fase de reinserção ou seleção natural, onde, ao invés de usar a distância de aglomeração (*crowding distance*) para diferenciar soluções em uma mesma fronteira, utiliza um método de agrupamento, no qual os indivíduos são divididos em nichos de acordo com suas proximidades em relação aos pontos de referência.

O NSGA-III é caracterizado pelo processo de atribuição de nicho chamado de classificação não-dominada baseada em pontos de referência. Sua ideia é traçar uma figura geométrica de uma dimensão a menos que o número de objetivos nos pontos extremos da primeira fronteira. Um número pré-definido de pontos de referência equidistantes são

distribuídos ao longo da figura, cada qual representando um respectivo nicho.

Os pontos de referência nada mais são do que um conjunto de pontos no espaço de objetivos que busca guiar o processo de busca, melhorando tanto a diversidade quanto a convergência do conjunto de soluções do algoritmo. Esses pontos representam as regiões nas quais se deseja encontrar soluções não dominadas, direcionando a busca evolutiva. Os pontos de referência podem ser gerados a partir de uma abordagem estruturada ou previamente fornecidos ao algoritmo. Por exemplo, se o usuário tem preferências sobre as regiões do espaço de busca que devem ser exploradas.

Na abordagem estruturada, é gerado um simplex unitário na dimensão imediatamente inferior à quantidade de objetivos. Por exemplo, se a quantidade de objetivos é 3, o simplex gerado tem dimensão 2. O simplex unitário é definido como a generalização do triângulo para uma determinada quantidade de dimensões. Assim, um simplex bi-dimensional é um triângulo, um simplex tridimensional é um tetraedro, e assim sucessivamente. O simplex gerado intercepta cada um dos eixos no ponto unitário, ou seja, para um simplex de dimensão 1 em um espaço de dois objetivos, os seus dois vértices são $v_1 = (1; 0)$ e $v_2 = (0; 1)$. Para cada eixo, são feitas d subdivisões entre 0 e 1, onde d é um parâmetro do NSGA-III. Por exemplo, para 4 subdivisões ($d = 4$), temos os pontos 0; 0,25; 0,5; 0,75 e 1 sobre cada eixo. Por exemplo, em um problema de 2 objetivos, pode-se definir uma estrutura similar à apresentada na Figura 9, para se estabelecer pontos de referência próximos aos quais se deseja encontrar soluções não dominadas, direcionando a busca evolutiva.

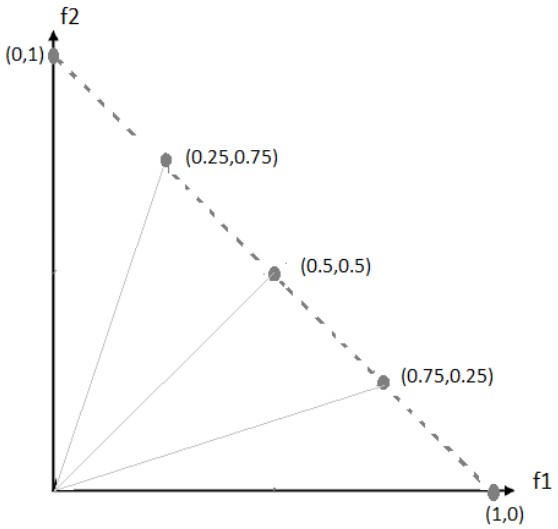

Figura 9 – Pontos de referência para um problema com 2 objetivos (normalizados) e 4 subdivisões.

Para classificar uma solução da população corrente, encontra-se a reta mais próxima formada a partir dos pontos de referência. Ao final, são selecionadas (sobrevivem) as

soluções localizados nas regiões menos lotadas do espaço de busca, com um menor número de indivíduos.

O NSGA-III segue os mesmos passos do NSGA-II até a classificação das soluções em fronteiras não dominadas, onde na primeira fronteira (F_1) estão contidas todas as soluções não dominadas da população, na segunda fronteira (F_2) estão as soluções que são dominadas apenas pela primeira fronteira, e assim por diante. A população para a próxima iteração é iniciada como um conjunto vazio ($P_{t+1} = \emptyset$), e então incluem-se as soluções de cada fronteira, começando pela primeira, até que se atinja o tamanho máximo da população (tam_pop). Se incluir a próxima fronteira faz com que o tamanho de P_{t+1} atinja ou ultrapasse o valor (tam_pop), deve-se tomar uma de duas decisões em relação à inclusão dos indivíduos dessa fronteira (F_{last}):

1. Se $|P_{t+1}| + |F_{last}| = tam\_pop$, faz-se $P_{t+1} = P_{t+1} \cup F_{last}$ e parte-se para a próxima iteração.

2. Se $|P_{t+1}| + |F_{last}| > tam\_pop$, acrescentar os elementos de F_{last} a P_{t+1} ultrapassa o limite da população. Portanto, é necessário selecionar quais soluções de F_{last} devem ser incluídas em P_{t+1}. No NSGA-III, essa seleção é realizada por meio da classificação não-dominada baseada em pontos de referência.

O NSGA-III se difere do seu precursor (NSGA-II) justamente nesse processo de selecionar os indivíduos de F_{last}. O primeiro passo da classificação não-dominada baseada em pontos de referência é a definição de um conjunto de soluções referência (S_r). Geralmente, o conjunto escolhido é a primeira fronteira não dominada, ou seja, F_1. Isso só não acontece quando o número de soluções na primeira fronteira é menor que o número de objetivos (m) e existe outra fronteira com pelo menos m elementos. Nesse caso, a primeira fronteira com m ou mais elementos é escolhida (F_{best}). Se a fronteira escolhida foi F_{last}, então $S_r = F_{last}$, caso contrário, $S_r = F_{last} \cup F_{best}$.

O conjunto de soluções referências (S_r) é então normalizado. Os maiores valores em S_r para cada objetivo se tornam 1, enquanto os menores se tornam 0. A seguir, calcula-se um ponto extremo para cada objetivo de acordo com as soluções em F_{best}. Para determiná-los, basta encontrar a solução em F_{best} que mais se aproxima do eixo correspondente ao objetivo (distância euclidiana baseada no valor normalizado). Para isso, ao calcular a distância da solução s ao eixo do objetivo, determina-se a distância entre s e um ponto imaginário p com todas as coordenadas valendo zero, exceto para o objetivo em questão, cujo valor é o mesmo de s. Por exemplo, para um problema de 6 objetivos, o cálculo da distância entre a solução $s = [0,8; 0,4; 0,5; 0,7; 0,1; 0,4]$ e o eixo do quarto objetivo é dada pela distância euclidiana entre s e o ponto $p = [0; 0; 0; 0,7; 0; 0]$, que vale $\sqrt{0,8^2 + 0,4^2 + 0,5^2 + 0 + 0,1^2 + 0,4^2} = 1,22$.

O passo seguinte, é distribuir as soluções em F_{last} em nichos. Os nichos são criados de acordo com os pontos extremos (um por objetivo) e o número de subdivisões (d, parâmetro

do NSGA-III). Inicialmente, cria-se uma sequência de retas que ligam os pontos extremos uns aos outros. Para 6 objetivos, por exemplo, liga-se o primeiro ponto extremo aos outros 5, criando 5 retas. O segundo ponto é ligado a todos os outros com exceção do primeiro que já está ligado, formando mais 4 retas. Com o terceiro ponto extremo, criam-se mais 3 retas. Com o quarto ponto formam-se duas novas retas e a última reta é criada pelos quinto e sexto pontos, gerando ao todo 15 retas. Em cada uma delas distribuem-se d pontos igualmente espaçados chamados de pontos de referência (P_{ref}).

Com o conjunto de pontos de referência (P_{ref}) em mãos, basta associar um nicho (inicialmente vazio) a cada ponto e então distribuir as soluções em F_{last}. Para cada ponto $p \in F_{last}$, verifica-se a distância entre ele e todos os pontos de referência $r \in P_{ref}$. Em seguida, p é incluído ao nicho referente ao ponto r com a menor distância para p. Após distribuir todas soluções entre os nichos, ordena-se cada nicho de acordo com as distâncias da solução para o ponto de referência. A solução mais distante do ponto de referência deve estar na primeira posição do nicho e a mais próxima, na última.

Como todas as soluções de F_{last} estão classificadas em nichos, basta adicionar uma por uma em P_{t+1} até que o limite no tamanho da população (*tam_pop*) seja atingido. Cada nicho possui um contador que inicia em 0. Cada vez que uma solução do nicho é escolhida, seu contador é incrementado em 1. Sempre, ao selecionar uma solução, escolhe-se o nicho com o menor valor de contador, e então extrai-se dele a solução na última posição (mais próxima do ponto de referência).

Com a população P_{t+1} formada de exatamente *tam_pop* indivíduos, volta a se realizar o processo idêntico ao NSGA-II, até que novamente, na próxima iteração, após a geração dos filhos, seja necessário selecionar os indivíduos sobreviventes. Mais detalhes sobre o processo de agrupamento podem ser obtidos no artigo original do algoritmo (DEB; JAIN, 2014).

3.1.5 SPEA2-SDE

O *SPEA2 with Shift-Based Density Estimation* (SPEA2-SDE) (LI; YANG; LIU, 2014) é uma pequena alteração no algoritmo SPEA2 que possibilita lidar com problemas que envolvam 4 ou mais objetivos de uma forma muito mais eficiente. A única alteração está no cálculo de distância entre duas soluções. Suponha que s_1 e s_2 sejam dois indivíduos na população e que seus valores no espaço de objetivo sejam, respectivamente, $[5, 10, 351, 7, 15]$ e $[6, 8, 15, 9, 14]$. No SPEA2, a distância entre s_1 e s_2 é dada pela distância euclidiana dos dois vetores ($dist$), ou seja:

$$dist(s_1, s_2) = \sqrt{(5-6)^2 + (10-8)^2 + (351-15)^2 + (7-9)^2 + (15-14)^2} = 336,0148 \tag{9}$$

As duas soluções são bem próximas, pois só estão distantes em uma das 5 coordenadas. Entretanto, o cálculo de distância do SPEA2 não reflete isso, o que é prejudicial em problemas com muitos objetivos. A fim de resolver esse problema, Li, Yang e Liu (2014) introduziram o cálculo de distância *Shift-Based Density Estimation* (SDE), que nada mais faz do que transladar a coordenada mais distante do segundo ponto para o mesmo valor no primeiro ponto. Isto é, antes de calcular a distância euclidiana, identifica-se a coordenada que exibe maior diferença entre os dois pontos. No caso de s_1 e s_2, a terceira coordenada é a que possui esse comportamento. Em seguida, é realizada a translação do segundo ponto na coordenada identificada para que seja igual ao primeiro ponto, (no exemplo, $s'_2 = [6, 8, \mathbf{351}, 9, 14]$). A distância SDE é, então, determinada pela distância euclidiana entre o primeiro ponto (s_1) e o segundo ponto transladado (s'_2). No exemplo, a distância SDE obtida é:

$$dist_{SDE}(s_1, s'_2) = \sqrt{(5-6)^2 + (10-8)^2 + (351-351)^2 + (7-9)^2 + (15-14)^2} = 3,1622 \tag{10}$$

A distância SDE descarta a coordenada mais distante ao calcular as distâncias, permitindo assim que pontos distantes em apenas uma coordenada ainda sejam considerados próximos. Todo o restante do processo do SPEA2-SDE segue os mesmos passos do algoritmo original.

3.1.6 MEAMT (AEMMT)

O *Multi-Objective Evolutionary Algorithm with Many Tables* (MEANDS) (BRASIL; DELBEM; SILVA, 2013) foi proposto originalmente com o nome em português Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas (AEMMT) e será referenciado como tal no decorrer desta dissertação. Assim como o MOEA/D, o AEMMT decompõe o problema multiobjetivo em subproblemas menores de um único objetivo. Entretanto, o AEMMT utiliza um esquema de tabelas, onde cada tabela representa uma combinação diferente dos objetivos investigados. A função que transforma os múltiplos objetivos em um valor escalar é a média aritmética e cada tabela mantém os melhores indivíduos, considerando o valor médio dos objetivos que ela representa. A cada geração, duas tabelas são selecionadas para o cruzamento. Dois pais, um de cada tabela, são sorteados aleatoriamente para gerarem um único filho. Essa nova solução (filho) é avaliada em todas as tabelas, entrando naquelas em que representar uma melhor aptidão em relação aos indivíduos atuais. Naturalmente, como um único *crossover* é realizado e um único filho é gerado a cada iteração, o AEMMT precisa de mais gerações para alcançar o mesmo número de soluções avaliadas que os algoritmos citados anteriormente.

A quantidade de tabelas é determinada pelo número de combinações possíveis entre os objetivos. Para quatro objetivos (f_1, f_2, f_3, f_4), por exemplo, serão criadas 15 tabelas

de combinações mais uma tabela extra, usada para guardar os indivíduos não-dominados, como ilustrado na Figura 10. Cada tabela possui um limite máximo de indivíduos. No início do algoritmo, gera-se um número pré-definido de soluções aleatórias de forma a preencher todas as tabelas com soluções diferenciadas umas das outras. Por exemplo, nos experimentos com o PRM descritos no Capítulo 6, foi gerado um número de soluções equivalente à quantidade de tabelas criadas multiplicada pelo tamanho máximo das tabelas (30). Assim, no caso do PRM com 4 objetivos, foram criadas $16 \times 30 = 480$ soluções e extraídas as 30 melhores para cada tabela, de acordo com a combinação de objetivos de cada uma.

No laço principal, um indivíduo só entra em uma tabela t, se for melhor que a pior solução na população atual de t. Com relação à tabela de dominância, sempre que um filho é gerado e ele não é dominado por nenhum outro indivíduo da tabela, ele é incluído. A restrição no tamanho da tabela de dominância pode ser diferente das demais e, sempre que o limite for atingido, é feito um truncamento priorizando a permanência das soluções com maior valor de média aritmética entre todos os objetivos.

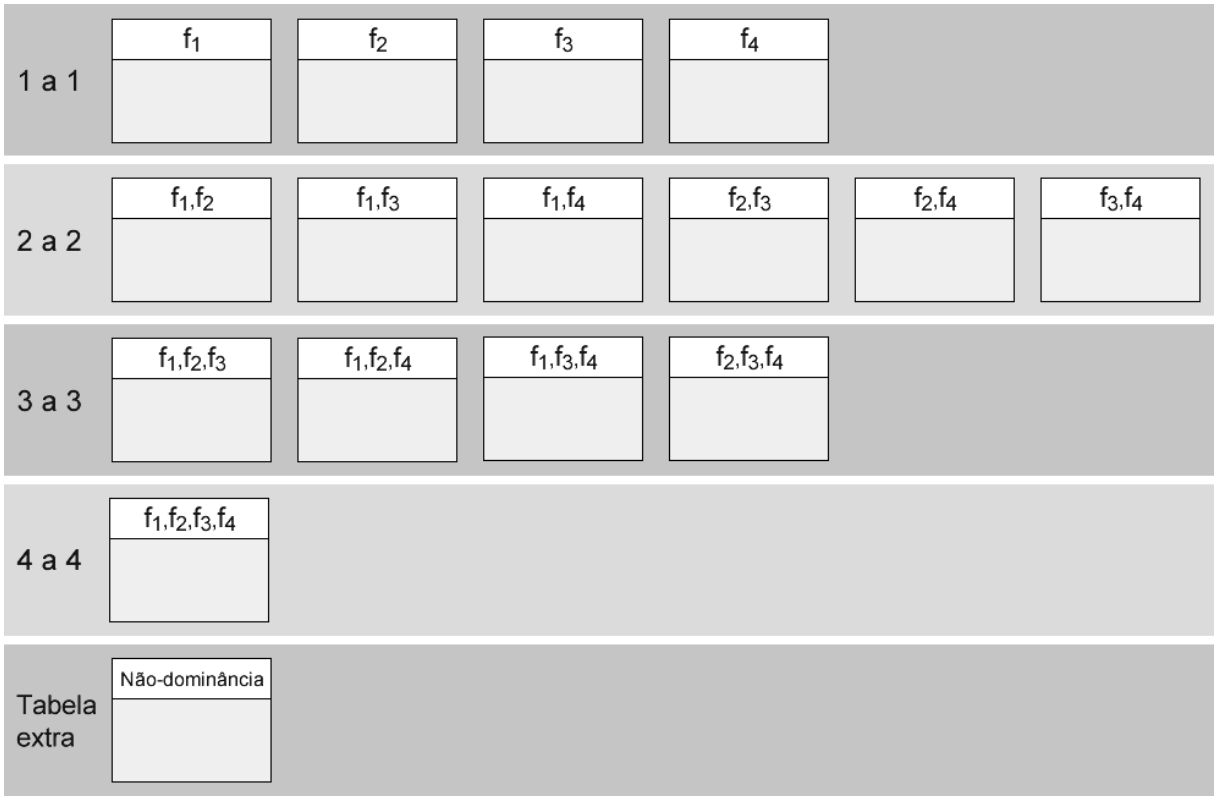

Figura 10 – Exemplo da quantidade de tabelas usadas pelo AEMMT

Após a geração e preenchimento das tabelas, inicia-se o laço principal. A cada iteração são escolhidas duas tabelas via torneio duplo, de acordo com suas pontuações. A pontuação de uma tabela tem valor inicial zero e sempre que essa tabela gera um filho que sobrevive para a geração seguinte, sua pontuação é incrementada. De acordo com

as observações dos autores da pesquisa, após um determinado número de gerações, as pontuações das tabelas eram tais que sempre as mesmas eram escolhidas para o *crossover*, assim, na versão final do algoritmo, as pontuações são zeradas a cada 100 gerações. Considerando-se as duas tabelas selecionadas, sorteia-se um indivíduo de cada e efetua-se o cruzamento entre eles. O filho gerado é, então, comparado tabela a tabela, sendo incluído naquelas em que representar uma melhoria, ou seja, o novo indivíduo supera pelo menos o pior indivíduo da tabela corrente. Após a execução de todas as gerações, dá-se como resultado o conjunto não dominado, considerando-se as soluções de todas as tabelas.

3.1.7 MEANDS (AEMMD)

O *Multi-Objective Evolutionary Algorithm based on Non-dominated Decomposed Sets* (MEAMT) (LAFETÁ et al., 2016) foi proposto originalmente com o nome em português Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias (AEMMD) e será referenciado como tal no decorrer desta dissertação. O AEMMD é uma modificação do AEMMT que, apesar de usar o mesmo processo de divisão do problema multiobjetivo, abandona a ideia de escalarização e adota o conceito de dominância também usado nos métodos clássicos (e.g. NSGA-II e SPEA2). No AEMMD, ao invés de se utilizar a média dos objetivos da tabela para avaliar o indivíduo, lança-se mão da relação de dominância de Pareto. Como o conceito de tabela do AEMMT e dos métodos que o inspiraram remete ao conceito de média e ordenação, essa estrutura, no AEMMD, é chamada apenas de conjunto de não-dominância. Um indivíduo novo s só entra no conjunto C_{nd}, se s não for dominado por nenhuma solução em C_{nd} (considerando apenas os objetivos de C_{nd}). Além disso, se s entra em C_{nd}, todas as soluções em C_{nd} dominadas por s são removidas.

O primeiro passo do AEMMD é gerar a lista de conjuntos de não-dominância, considerando todas as combinações de objetivos possíveis a partir de dois a dois. Combinações de um único objetivo não são criadas, pois o conceito de dominância é válido a partir de dois valores. Diferentemente do AEMMT, os conjuntos não possuem limite de tamanho e podem crescer indefinidamente. Por exemplo, para um problema de quatro objetivos (f_1, f_2, f_3, f_4), 11 conjuntos seriam gerados, como pode ser observado na Figura 3.1.7.

Com os conjunto de não-dominância criados, gera-se um número pré-definido de soluções aleatórias, distribuindo-as pelos conjuntos de acordo com a relação de dominância de Pareto. Assim como no AEMMT, a cada iteração do algoritmo, é utilizado um torneio duplo para selecionar dois conjuntos de acordo com suas pontuações. Entretanto, a pontuação dos conjuntos no AEMMD é diferente da pontuação nas tabelas do AEMMT. Ao invés de conceder um ponto sempre que o conjunto contribui com um pai que gera um indivíduo sobrevivente, pontua-se quando ele recebe um indivíduo novo. Uma vez selecionados os dois conjuntos, sorteia-se um representante de cada e gera-se um único filho, o qual é comparado conjunto a conjunto, sendo inserido naqueles onde representa uma solução não dominada. Outra diferença observada pelo autor em relação ao AEMMT é que

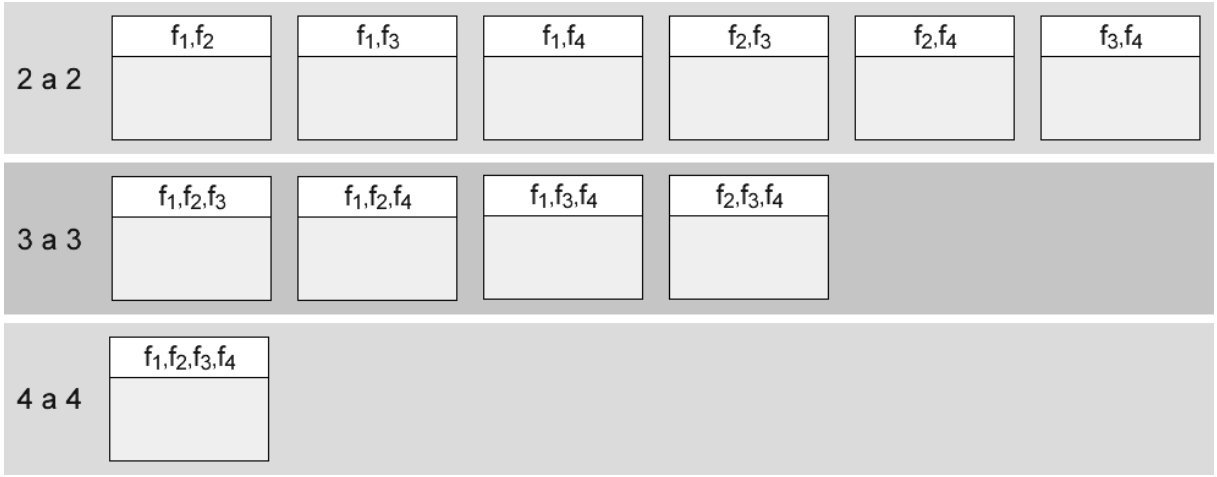

Figura 11 – Exemplo da quantidade de conjuntos usados pelo AEMMD

o novo modelo de pontuação não degrada no decorrer das gerações, ou seja, mesmo após várias gerações, ainda existe grande diversidade na escolha dos conjuntos pais, tornando desnecessária a reinicialização das pontuações associadas aos conjuntos (LAFETÁ et al., 2016). Espera-se que ao final das gerações, a população do conjunto que considera todos os objetivos no seu critério de dominância tenha convergido para a fronteira de Pareto.

3.2 Algoritmos multiobjetivos baseados em colônias de formigas

A maior parte dos métodos de busca multiobjetivo são baseados em algoritmos genéticos. Entretanto, uma alternativa que ainda é pouco explorada são as técnicas inspiradas em inteligência coletiva. Dentre elas, optou-se por investigar métodos de otimização baseados em colônias de formigas, os quais são particularmente adequados para lidar com problemas discretos, como aqueles utilizados neste trabalho (problema da mochila multiobjetivo e problema do roteamento multicast). Dentre os ACOs multiobjetivos encontrados na literatura, destacam-se o MOACS (BARÁN; SCHAERER, 2003) e o MOEA/D-ACO (KE; ZHANG; BATTITI, 2013).

3.2.1 MOACS

O *Multi-Objective Ant Colony Optimization Algorithm* (MOACS) (BARÁN; SCHAERER, 2003) é uma adaptação do ACO original que torna possível a otimização de múltiplos objetivos utilizando uma única estrutura de feromônios, múltiplas heurísticas e um arquivo de soluções não dominadas. Esse algoritmo foi proposto originalmente para o problema de roteamento de veículos com janelas de tempo e, posteriormente, foi apli-

cado no problema do roteamento multicast (PINTO; BARAN, 2005). Uma variação desse algoritmo, mais voltada à solução de problemas *many-objective*, foi apresentada em (RIVEROS et al., 2016), a qual é utilizada neste trabalho. A diferença básica para o original está na geração dos pesos para as heurísticas. No modelo original são geradas todas as combinações possíveis, o que é inviável em problemas com muitos objetivos. Na versão *many-objective*, o problema é solucionado através de uma amostragem desses vetores de peso. O funcionamento básico do MOACS é descrito no Algoritmo 2.

Algoritmo 2 Algoritmo MOACS

1: Inicialize a estrutura de feromônios τ_{ij} com τ_0 /* τ_0 é o valor inicial */
2: Crie um conjunto vazio de soluções não-dominadas ND
3: **enquanto** Número máximo de iterações não for atingido **faça**
4: **para** $i \leftarrow 0$ até *tamanho_populacao* **faça**
5: Sorteie valores no intervalo $[0, w_{max}[$ para formar um vetor de pesos W com $|H|$ posições
6: Construa uma solução de acordo com a tabela de feromônios τ_{ij}, as heurísticas (H) e os pesos W
7: Atualize ND com a nova solução
8: **fim para**
9: **se** ND foi modificado **então**
10: Reinicie a estrutura de feromônios fazendo $\tau_{ij} = \tau_0 \forall (i, j)$
11: **senão**
12: Atualize a estrutura de feromônios com todas as soluções em ND
13: **fim se**
14: **fim enquanto**
15: **retorne** ND

O processo de construção de uma solução depende do problema investigado. No MOACS, os principais componentes utilizados nesse processo são:

- Feromônios (τ_{ij}): estrutura que guarda a quantidade de feromônios em cada partícula que pode formar a solução. No caso de problemas em grafos, representa a quantidade da substância em cada uma das arestas, indicando a frequência de suas utilizações;

- Heurísticas (H): conjunto de funções que estimam a qualidade de uma dada partícula que pode formar a solução. No caso de problemas em grafos, representa os vários pesos em uma aresta. Por exemplo, num grafo que representa uma rede de computadores com informações de custo, distância e tráfego, H poderia ser formado de três funções que recebem uma aresta (i, j) e devolvem, respectivamente, os valores de suas métricas custo, distância e tráfego.

- Peso máximo de uma heurística (w_{max}): representa a granularidade da escala de pesos que cada heurística pode assumir. Em (RIVEROS et al., 2016), propõe-se

$w_{max} = 3$, de forma que cada função possa ser classificada como 0 (não importante), 1 (importante), 2 (muito importante).

- Vetor de pesos (W): O vetor de pesos atribui a importância de cada heurística e normalmente é gerado de forma aleatória em cada iteração. Cada função de heurística recebe um valor inteiro representando o peso e variando no intervalo $[0, w_{max}[$.

Ao construir uma solução, utiliza-se o mesmo processo de decisão do ACO original, explicado na seção 2.2.2. A única diferença é que as múltiplas heurísticas do MOACS (H) são unificadas em uma única função $h(x)$, por meio de uma média ponderada. Nesse processo, utiliza-se o vetor de pesos W para definir o fator de ponderação de cada heurística. Dessa forma, a função h(x) é dada por:

$$h(x) = \frac{\sum_{i \leftarrow 0}^{size(H)} H_i(x) \times W_i}{\sum_{w \in W} w} \tag{11}$$

Em cada época (iteração do ACO), atualiza-se o arquivo de soluções não dominadas com as novas soluções geradas. Se o arquivo foi atualizado após a criação de todas as soluções, reiniciam-se as informações de feromônio, redefinindo todos os valores na estrutura τ_{ij} para o valor inicial de feromônio τ_0. Caso o arquivo tenha se mantido estável, ou seja, nenhuma das novas soluções é não-dominada, atualizam-se as quantidades de feromônio na estrutura de acordo com as soluções no arquivo.

Considerando-se um problema em grafos, para atualizar a estrutura τ_{ij} com uma solução s, faz-se:

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \rho \times \Delta\tau(s) \qquad \forall (i,j) \in s \tag{12}$$

Sendo:

- ρ: coeficiente de evaporação;
- $\Delta\tau(s)$: Quantidade de feromônios depositados pela solução s, que por sua vez é definida por:

$$\Delta\tau(s) = \frac{1}{performance(s)} \tag{13}$$

Na fórmula anterior, $performance(s)$ é dado pela soma dos valores de s no espaço de objetivos. Neste caso, considera-se um problema de minimização. Por exemplo, se os objetivos são reduzir o custo, o tráfego e o atraso (*delay*) de uma rede, então, $performance(s) = custo(s) + trafego(s) + delay(s)$. Para problemas de maximização, basta inverter a equação.

Após todas as iterações do MOACS, espera-se obter no arquivo uma boa aproximação da fronteira de Pareto.

3.2.2 MOEA/D-ACO

O *Multiobjective evolutionary algorithm based on decomposition ACO* (MOEA/D-ACO) (KE; ZHANG; BATTITI, 2013) é um método que adapta o algoritmo MOEA/D (seção 3.1.3) ao *framework* de otimização por colônia de formigas (ACO). Os conceitos de células e vizinhanças são reutilizados, enquanto que a reprodução local, inexistente no ACO, é substituída por um processo de construção da solução levemente modificado. Além disso esse método introduz um novo conceito de grupos que é utilizado para agrupar as formigas de acordo com seus vetores de peso. Nesse algoritmo, o termo formiga corresponde ao conceito de célula do MOEA/D.

O primeiro passo do MOEA/D-ACO consiste em gerar os vetores de peso que, assim como no MOEA/D, são arranjos de valores entre 0 e 1, cujas somas valem 1. A geração pode ser aleatória ou seguir alguma distribuição pré-definida. No caso deste trabalho, utilizou-se a distribuição uniforme proposta no artigo original (KE; ZHANG; BATTITI, 2013). Atribui-se cada vetor a uma formiga e cria-se a estrutura de vizinhanças, onde cada formiga é associada a uma vizinhança contendo as v_{size} formigas mais próximas de acordo com os vetores de peso (incluindo ela mesma). O segundo passo do algoritmo determina os grupos, sendo que cada grupo recebe um novo vetor de pesos gerado da mesma maneira que os pesos anteriores. As formigas são então distribuídas entre os grupos de acordo com a proximidade entre seu vetor de pesos e o vetor de pesos do grupo (*agrupamento*).

Com respeito às duas principais estruturas de um ACO, heurística e feromônios, cada formiga possui uma função heurística e cada grupo mantém uma estrutura de feromônios. Quando os grupos são criados, cada um recebe uma estrutura de feromônios com o valor máximo de feromônio em cada uma das posições. Por sua vez, quando a formiga é criada, recebe uma função heurística baseada na combinação de todas as funções heurísticas do problema e no vetor de pesos da própria formiga. O cálculo específico da heurística depende do problema.

O processo iterativo (evolução) do MOEA/D-ACO consiste em gerar as soluções, atualizar o conjunto de soluções não-dominadas ND, distribuir as novas soluções entre as formigas e atualizar as estruturas de feromônio. Primeiramente, para cada grupo G, geram-se as soluções para cada formiga $f \in G$. A solução é gerada com base nos feromônios de G, na heurística de f e na solução atual de f. Tendo gerado todas as soluções para um grupo, atualiza-se o arquivo de soluções não-dominadas ND. Para cada nova solução S que entrou no arquivo, modifica-se a estrutura de feromônios τ de G. Assim, o cálculo do feromônio de uma partícula e na iteração $i+1$ ($\tau_{i+1}(e)$) é obtido a partir do seu feromônio atual ($\tau_i(e)$), como segue:

$$\tau_{i+1}(e) = \begin{cases} \rho \times \tau_i(e) + \delta, & \text{se } e \in S \\ \rho \times \tau_i(e), & \text{caso contrário} \end{cases} \tag{14}$$

Onde e é uma possível partícula de uma solução para o problema investigado (qualquer aresta, no caso do PRM, ou item, para o PMM) da tabela de feromônios; ρ é a taxa de evaporação; e δ é dado pelo inverso da soma dos valores da função de escalarização aplicada sobre cada solução não dominada de G em relação ao vetor de pesos de G.

$$\delta = \frac{1}{\sum_{s \in S_{nd} f_{esc}(s,W)}} \tag{15}$$

Sendo, S_{nd} o conjunto de soluções não dominadas de G, f_{esc} a função de escalarização, e W o vetor pesos de G. Caso o problema seja de maximização, δ seria a soma dos valores, ao invés do inverso dela.

Cada formiga armazena duas soluções, a solução atual, que de início é nula, e a nova solução, criada em cada iteração do algoritmo. Após a geração de novas soluções a atualização de feromônios para todos os grupos e suas formigas, deve-se decidir a solução atual de cada formiga com base nas novas soluções da vizinhança. Nesse processo, para cada formiga f analisam-se as novas soluções presentes na vizinhança de f. Se alguma nova solução sn, que ainda não substituiu nenhuma outra, possui um *fitness* melhor que o da solução corrente de f_{esc} (sc), substitui-se sc por sn. O *fitness* é calculado de acordo com a média ponderada dos valores da solução em cada objetivo baseadas no vetor de pesos da formiga em questão. Neste trabalho, utilizou-se a soma ponderada como função *escalarizadora*, mas qualquer uma das funções propostas em (ZHANG; LI, 2007) pode ser usada.

O processo de construção da solução depende do problema investigado. Entretanto, o MOEA/D-ACO inclui duas novas características no processo, independente do problema. São elas:

- **Influência da solução atual:** no MOEA/D-ACO, cada formiga mantém uma solução atual que influencia a construção da próxima solução com base em um parâmetro δ. Isso acontece devido à introdução de um novo termo no cálculo de feromônio na construção da solução. Ao calcular a probabilidade de uma partícula p (aresta ou item) fazer parte da solução, ao invés de adotar $feromonio(p)^{\alpha}$, considera-se $(\delta * x + feromonio(p))^{\alpha}$, sendo $x = 1$ se p pertence à solução atual e $x = 0$ caso contrário.

- **Taxa de elitismo:** a maioria dos ACOs utilizam uma espécie de roleta para decidir qual partícula fará parte da solução. Aquelas com maior probabilidade têm maior chance de serem escolhidas, mas não necessariamente serão. Em uma estratégia elitista, não se utiliza a roleta. A partícula escolhida será aquela que receber o maior valor de probabilidade. No MOEA/D-ACO, utiliza-se uma taxa de elitismo que estipula a chance de uma partícula ser escolhida por elitismo ao invés de roleta.

O processo iterativo do algoritmo é executado até que uma condição de parada pré-definida seja atingida. Normalmente, adota-se um número máximo de iterações. Espera-se que, nesse momento, as soluções no conjunto ND tenham convergido para a fronteira de Pareto.

3.3 Outros algoritmos multiobjetivos

Outros algoritmos multiobjetivos mais citados na literatura, mas que não foram analisados neste trabalho, são mencionados a seguir.

Vector Evaluated Genetic Algorithm (VEGA) (SCHAFFER, 1984) e *Multi-Objective Genetic Algorithms* (MOGA) (MURATA; ISHIBUCHI, 1995) foram os primeiros algoritmos multiobjetivos a aparecerem na literatura. Posteriormente, os algoritmos *Non-Dominated Sorting Genetic Algorithm* (NSGA) (SRINIVAS; DEB, 1994) e *Strength Pareto Evolutionary Algorithm* (SPEA) (ZITZLER; THIELE, 1999) foram propostos e são considerados os mais eficientes da primeira geração de Algoritmos Evolutivos Multiobjetivos (AEMOs). A segunda geração de AEMOs é caracterizada por métodos elitistas, sendo que NSGA-II e o SPEA2 apresentados na 3.1 são considerados os mais eficientes dessa geração e até hoje são os mais utilizados na literatura, especialmente com 2 e 3 objetivos. Em problemas *many-objective*, uma das principais dificuldades encontradas é o excesso de soluções não-dominadas, o que dificulta a identificação de melhores indivíduos e prejudica a convergência. Para lidar com esse problema, Aguirre e Tanaka (2009) apresentam um conceito menos rigoroso de dominância, dando origem ao algoritmo ϵ-MOEA. Por sua vez, a fim de reclassificar as soluções não dominadas e resolver o mesmo problema, Beume, Naujoks e Emmerich (2007) propõem o algoritmo SMS-EMOA, no qual a evolução da população ocorre de acordo com indicadores de qualidade do conjunto de soluções. Nesse algoritmo, os autores usam o hiper-volume para avaliar soluções consideradas igualmente boas pela dominância de Pareto. Bader e Zitzler (2011) utilizam uma ideia parecida ao propor o algoritmo Hype. Esse algoritmo também utiliza o hiper-volume como parâmetro para guiar a evolução da população.

Em relação aos algoritmos de inteligência coletiva, o algoritmo MOEA/D-BACO foi proposto por Souza e Pozo (2015) e aplica uma variação do MOEA/D-ACO no Problema de Programação Binária Quadrática Irrestrito (UBQP). Como para o UBQP as estruturas de feromônio crescem de forma exponencial em relação ao tamanho da entrada, o método MOEA/D-ACO original se torna inviável. A variação proposta pelos autores, algoritmo MOEA/D-BACO, visa solucionar esse problema através da substituição do ACO, no MOEA/D-ACO, pelo *Binary Ant Colony Optimization* (BACO) (KONG; TIAN, 2006), que é um algoritmo mais simples e promete menor tempo de execução em relação ao algoritmo original. Em 2001, Iredi, Merkle e Middendorf propuseram o algoritmo *Bi-Criterion Optimization with Multi Colony Ant Algorithms* para o problema

Single Machine Total Tardiness Problem (SMTTP) com dois objetivos (IREDI; MERKLE; MIDDENDORF, 2001). O algoritmo usa duas tabelas de feromônios (uma para cada objetivo) e cada formiga recebe dois valores de peso que são distribuídos uniformemente entre a colônia, esses valores representam o peso de cada objetivo nas decisões da formiga. A atualização dos feromônios é realizada com base nas formigas que geraram soluções não dominadas na geração corrente. Iredi, Merkle e Middendorf (2001) também propõem um método com múltiplas colônias, onde cada colônia trabalha com duas tabelas de feromônios, mas, através da atribuição de pesos aos objetivos, pesquisam em regiões diferentes do espaço de busca. Segundo Lopez-Ibanez e Stutzle (2012), os algoritmos os algoritmos ACO multiobjetivos (MACOs) seguem todos uma mesma formulação e que podem ser modelados como um único *framework*, diferenciando apenas em algumas decisões de projeto, que seriam parâmetros desse *framework* geral capaz de gerar qualquer MACO. Nesse trabalho, os autores classificam diversos MACOs na literatura e destacam onde se diferem, mostrando as estratégias que podem ser adotadas ao elaborar um algoritmo ACO multiobjetivo. Além disso, o artigo propõe uma forma de gerar automaticamente os parâmetros de um MACO a fim de obter os melhores valores possíveis para um determinado problema sem que se seja necessário testar manualmente diversas configurações.

Capítulo 4

Problemas de teste

A fim de avaliar o desempenho de algoritmos de otimização, é comum que sejam estabelecidos diferentes problemas de teste. São vários os problemas em que se pode aplicar os algoritmos multiobjetivos, os quais podem ser divididos em duas categorias principais: contínuos ou discretos. Os problemas contínuos são representados por funções contínuas e muitos dos exemplos na literatura não representam um problema real. Alguns exemplos de problemas contínuos (SCH, FON, POL, KUR e ZDT) podem ser encontrados no artigo original do NSGA-II (DEB et al., 2002a). Os problemas discretos, por outro lado, possuem um espaço de busca discreto e nem todas as soluções possíveis são válidas, ou seja, existem lacunas no contradomínio das funções. A maioria desses problemas está relacionada à área de otimização combinatória, na qual se tem um conjunto de objetos e deseja-se encontrar a melhor (ou mais viável) combinação ou permutação desses objetos. Exemplos de problemas discretos comumente usados na literatura multiobjetivo são: caixeiro viajante (LUST; TEGHEM, 2010), roteamento de veículos com janelas de tempo (OMBUKI; ROSS; HANSHAR, 2006), problema da mochila (BAZGAN; HUGOT; VANDERPOOTEN, 2009), sequenciamento de proteínas (BRASIL; DELBEM; SILVA, 2013) e problemas de roteamento em redes (LAFETÁ et al., 2018). O comportamento e a adequação ao uso de um determinado algoritmo multiobjetivo são influenciados pelo tipo do problema. Neste trabalho, dois problemas discretos foram investigados: o problema da mochila multiobjetivo (PMM) e o problema do roteamento multicast (PRM). Esses problemas são interessantes, pois, ao mesmo tempo que se assemelham em suas características discretas, diferem em suas naturezas. O PMM representa um problema de combinação multimodal, no qual várias configurações (indivíduos) podem representar uma mesma solução. O PRM é um problema de permutação, no qual a ordem dos componentes é importante e pode afetar a qualidade da solução. Ambos os problemas são descritos em detalhes nas subseções a seguir.

4.1 Problema da mochila multiobjetivo

4.1.1 Definição do problema

O Problema da Mochila (PM) é um problema muito comum na computação e possui um caráter teórico. Entretanto, existem problemas reais equivalentes que podem ser resolvidos com as mesmas técnicas, como o escalonamento de tarefas entre as várias linhas de produção de uma fábrica.

O problema consiste em arranjar um conjunto de itens em uma mochila de forma a não exceder sua capacidade e, ao mesmo tempo, maximizar o valor (lucro) dos objetos carregados. Matematicamente, dada uma mochila de capacidade C e um conjunto de itens O, onde cada $o_i \in O$ possui um peso $peso(o_i)$ e um lucro $lucro(o_i)$, deseja-se encontrar o conjunto $S \subset O$, tal que $\sum_{o \in S} peso(o) \leq C$ e $\sum_{o \in S} lucro(o)$ seja o maior possível. A Figura 12 mostra uma instância do problema da mochila mono-objetivo e três soluções possíveis.

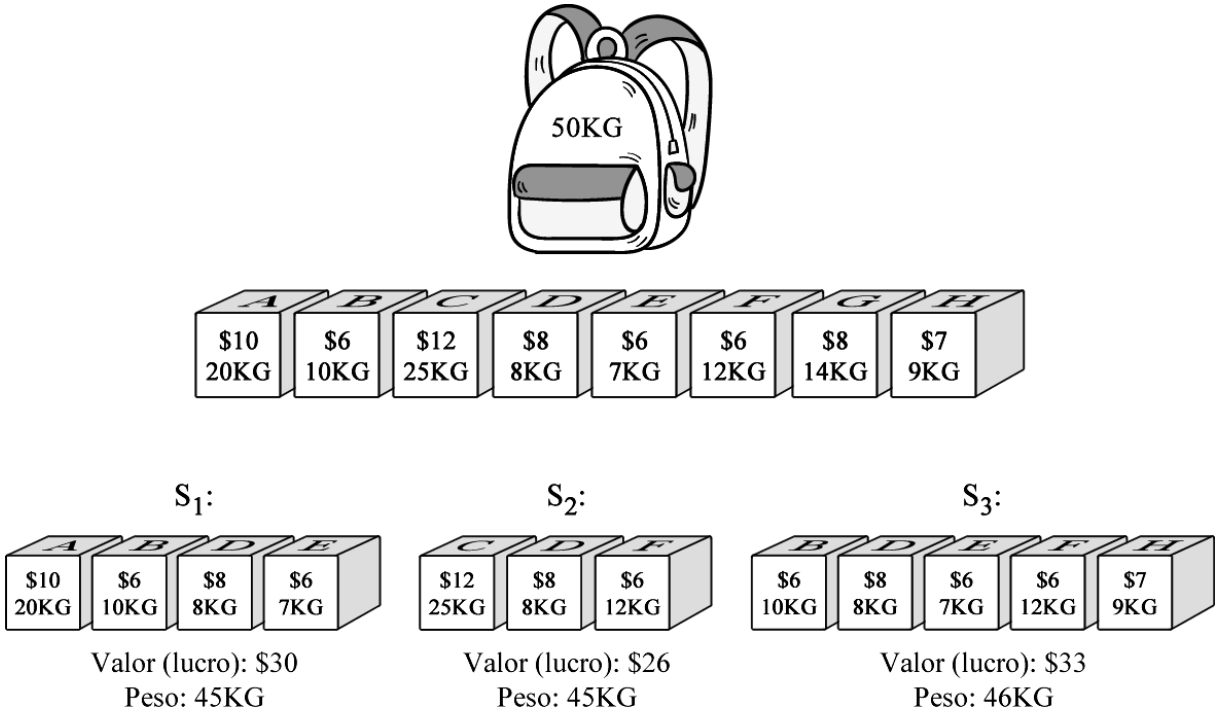

Figura 12 – Exemplo para o problema da mochila mono-objetivo.

Na Figura 12, a capacidade máxima da mochila é 50Kg e existem oito itens disponíveis para se escolher. Três soluções possíveis são mostradas (S_1, S_2 e S_3), sendo que a melhor delas é S_3, pois possui o maior valor de lucro e ao mesmo tempo é válida, ou seja, respeita a restrição de capacidade (peso menor que 50Kg). No exemplo, todas as soluções estão saturadas, em outras palavras, a adição de qualquer outro item as tornam inválidas.

Existem diversas estratégias para resolver o PM. Dentre elas, as mais usadas são os algoritmos gulosos (HRISTAKEVA; SHRESTHA, 2005), a programação dinâmica (TOTH,

1980) e os algoritmos genéticos (USLU, 2015). Os algoritmos gulosos e a programação dinâmica são os mais rápidos e eficientes para resolver o PM mono-objetivo. Em contrapartida, a complexidade adicionada ao considerar mais de um objetivo inviabiliza a utilização desse tipo de algoritmo, tornando os algoritmos genéticos e demais métodos bio-inspirados as melhores opções.

O problema da mochila multiobjetivo (PMM) é similar ao mono-objetivo (PM). Sua única diferença está no fato de que cada item, ao invés de possuir um único valor (lucro), é composto de múltiplos valores. No PMM, a função $lucro(o_i)$ retorna um vetor ao invés de um único valor escalar. Cada componente (elemento) do vetor representa o valor do item o_i considerando um dos objetivos. Por exemplo, considerando um PMM de 3 objetivos, cada $o_i \in O$ possui um vetor formado por três valores de lucros (um para cada objetivo). O objetivo do problema passa a ser maximizar todos os lucros ao invés de um único valor. A definição do problema da mochila utilizada neste trabalho, também utilizada em (BAZGAN; HUGOT; VANDERPOOTEN, 2009), considera apenas um peso por item e uma única restrição, referente à capacidade da mochila. Outra definição mais comum do PMM envolve a consideração de múltiplos pesos por item e múltiplas restrições para a mochila, o número de pesos e de restrições é o mesmo que o número de objetivos. Essa definição alternativa é bastante comum na literatura e foi utilizada nos trabalhos de (ALAYA; SOLNON; GHÉDIRA, 2004; CHU; BEASLEY, 1998; ISHIBUCHI; AKEDO; NOJIMA, 2015).

O PMM já foi utilizado várias vezes para avaliar algoritmos multiobjetivos, podendo-se destacar os trabalhos de (ZITZLER; THIELE, 1999), (ZITZLER; LAUMANNS; THIELE, 2001) e (ZHANG; LI, 2007). As instâncias do PMM utilizadas no decorrer deste trabalho foram geradas de forma aleatória e compreendem problemas da mochila com 30, 40, 50, 100 e 200 itens. Para gerar as instâncias, sortearam-se valores no intervalo (0, 1000) para os lucros e para os pesos. A capacidade da mochila foi sempre definida como 60% da soma dos pesos.

Uma das partes mais importantes de um algoritmo de busca bio-inspirado é a modelagem da solução. Para resolver o PMM através de algoritmos genéticos, é necessário definir a representação da solução, a geração aleatória de indivíduos, o cruzamento e a mutação. A representação de uma solução no AG para o problema da mochila mono-objetivo é simples, pois a solução é representada por um vetor binário e a literatura está repleta de exemplos que podem ser resolvidos dessa maneira (USLU, 2015; HRISTAKEVA; SHRESTHA, 2004). A versão *many-objective* do problema não requer modificação no modelo, fazendo com que os mesmos processos de cruzamento e mutação possam ser utilizados. Nas seções a seguir, detalhe-se a representação da solução e os operadores genéticos para o PMM usados neste trabalho. A construção das soluções nos algoritmos ACO é mais elaborada e será definida no próximo capítulo.

4.1.2 Representação da solução e geração da população inicial

Uma solução para o PMM é representada por um vetor binário. Se a instância do problema da mochila apresenta 10 itens ao total, por exemplo, um vetor com 10 bits representa a solução. As posições do vetor representam a ausência ou a presença de cada item. Se a posição i do vetor é 0, então o i-ésimo item não faz parte da solução. Caso contrário, se o vetor na posição i vale 1, então o i-ésimo item está presente na mochila. Por exemplo, em uma solução representada pelo vetor [1,0,0,1,0,1,1,0,0,0], apenas os itens 0, 3, 5 e 6 são colocados na mochila, os outros itens (1, 2, 4, 7, 8 e 9) são descartados. Essa abordagem garante que cada solução possua uma única forma de representação, eliminando a característica multimodal do problema.

Nesse cenário, a geração aleatória de uma solução consiste em sortear os valores 0 ou 1 para cada posição do vetor. Tanto a geração aleatória de um indivíduo na formação da população inicial, quanto a combinação de vetores (cruzamento) não garantem a formação de uma solução válida, ou seja, é possível criar um vetor, cuja soma dos pesos dos itens ultrapasse a capacidade da mochila. Portanto, após a geração de qualquer solução, seja aleatória na população inicial ou proveniente de um *crossover*, é necessário executar um processo de validação da solução. Para validar um vetor binário, basta verificar a soma dos pesos associados aos itens representados por 1. Caso esse valor seja maior que a capacidade da mochila, remove-se um item aleatório. Esse processo é repetido sucessivamente, até que a solução se torne válida, ou seja, até que a soma dos pesos dos itens respeite a capacidade da mochila (ISHIBUCHI; AKEDO; NOJIMA, 2015).

4.1.3 Cruzamento e mutação

O cruzamento entre duas soluções binárias, como explicado na Seção 2.1, pode ser efetuado de diversas maneiras. Neste trabalho, foi utilizado *crossover* uniforme, ou seja, o filho herda de forma aleatória os bits do pai 1 ou do pai 2. Esse processo é ilustrado na Figura 13.

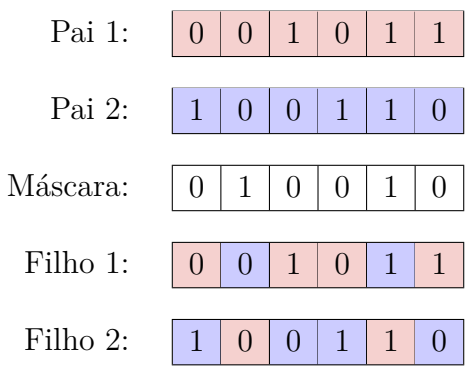

Figura 13 – Exemplo de crossover uniforme

Como pode ser visto na Figura 13, o *crossover* uniforme pode ser implementado com uma máscara, que é um vetor aleatório de bits que controla os genes herdados de cada pai. Se o bit na posição i da máscara vale 0, então o gene do filho na posição i herda o valor de um dos pais (ex: pai 1), caso contrário, herda o valor do outro (ex: pai 2). Dessa forma, ainda é possível gerar 2 filhos: um deles usando a regra na qual o bit 0 da máscara representa o pai 1 e o bit 1 representa o pai 2 (filho 1 na figura); e o outro usando a máscara com os valores complementares (filho 2).

Após o *crossover*, existe uma chance, determinada pela taxa de mutação definida para o AG, do indivíduo sofrer alguma mutação genética. Neste trabalho, foi utilizado o método mais simples de mutação para vetores binários, ou seja, a inversão de bit. Esse processo consiste em sortear uma posição aleatória do vetor e se o valor for 0, troca-se para 1, caso contrário, troca-se para 0.

4.2 Problema do roteamento multicast

4.2.1 Definição do problema

O problema do roteamento multicast (PRM) aparece na engenharia de tráfego em redes de computadores e consiste em escolher a forma "mais eficiente" de transmitir uma mensagem multicast (LAFETÁ et al., 2016). Uma transmissão de rede pode ser do tipo unicast, multicast ou broadcast. Em transmissões unicast, conecta-se um ponto da rede a outro ponto qualquer (transmissão ponto-a-ponto). Para fazer isso de forma eficiente, é possível usar algum algoritmo capaz de encontrar o melhor caminho entre os dois pontos (e.g. (DIJKSTRA, 1959)), se apenas uma métrica for utilizada. As comunicações broadcast caracterizam-se pelo fato de um nó da rede (servidor) enviar o conteúdo a todos os demais. Para obter as melhores rotas para trafegar os dados, é possível utilizar algum algoritmo capaz de determinar a árvore geradora de custo mínimo (e.g. (PRIM, 1957)), se apenas uma métrica for empregada na otimização. Em uma comunicação multicast, o objetivo é transmitir o conteúdo de um nó da rede, chamado nó de origem ou transmissor, para alguns outros nós, denominados receptores. Essa tarefa apresenta maior complexidade, mesmo no caso de uma única métrica, pois é necessário obter uma árvore de Steiner de custo mínimo, o que é mais difícil que calcular uma única rota ou construir a árvore geradora de custo mínimo (BUENO; OLIVEIRA, 2010).

O PRM é um problema prático muito importante. Seu estudo visa o desenvolvimento de algoritmos eficientes e eficazes, proporcionando avanços na geração de rotas em redes de computadores. O objetivo desses algoritmos é encontrar soluções (rotas multicast) que permitam uma comunicação mais rápida, menos custosa e mais confiável entre dispositivos, o que é essencial em uma era onde a maioria das pessoas consomem informação e entretenimento pela Internet.

Uma vez que se deseja transmitir um conteúdo via uma rede de computadores, o problema, em sua versão com um único objetivo de custo, consiste em encontrar a melhor rota possível entre a fonte de dados e os vários destinos. Dado um grafo $G = (V, E)$, que representa a rede de comunicação, um nó raiz $r \in V$ (nó transmissor) e um conjunto de nós destinos $D \subset V$ (nós receptores), o PRM consiste em determinar a subárvore T de G enraizada em r que inclui todos os vértices em D e apresenta o menor custo possível. A Figura 14 ilustra uma instância do problema de roteamento multicast. Nesse exemplo de rede de comunicação, o nó transmissor é definido por um círculo duplo (vértice 0), os nós receptores são destacados em cinza escuro (vértices 1, 8, 12 e 13) e os nós intermediários estão em cinza claro. Os números em cada aresta determinam o custo da comunicação entre os nós que ela conecta.

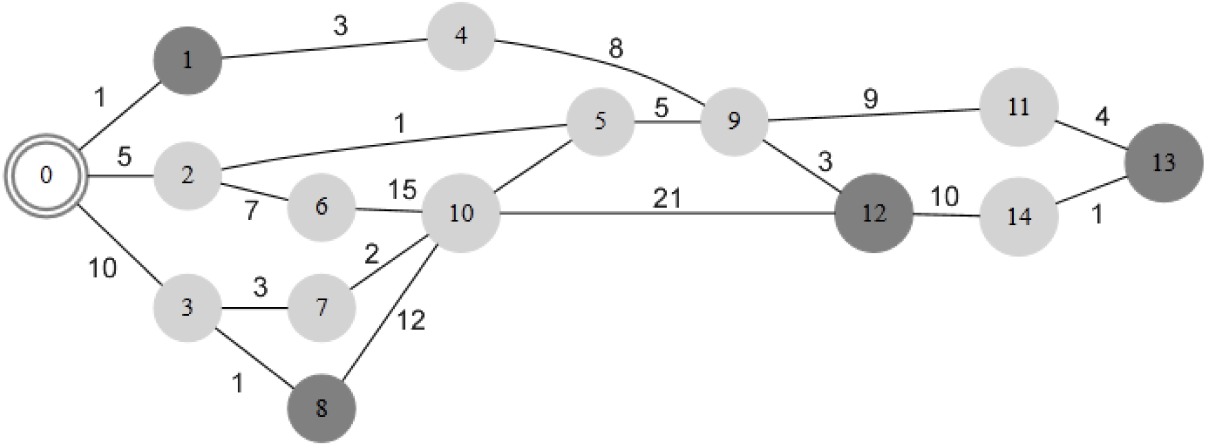

Figura 14 – Exemplo de uma rede de comunicação. Retirado de (BUENO, 2010).

Na Figura 15, são apresentados alguns exemplos de árvores multicast criadas a partir do grafo mostrado na figura Figura 14. O custo de cada árvore é dado pela soma dos custos de suas arestas. Dentre os exemplos, a árvore mais à direita possui o menor custo total: 65.

O PRM original é proposto com apenas um objetivo a se otimizar e utiliza uma métrica por enlace, que é o custo. Essa métrica pode representar uma combinação de outras métricas. Entretanto, áreas que tratam da melhoria do desempenho em redes de computadores, como a engenharia de tráfego e a Qualidade de Serviço (QoS) trouxeram uma nova necessidade para a transmissão em redes, onde a utilização de métricas distintas é mais adequada ao gerenciamento da rede. Características como distância, atraso (*delay*), capacidade de tráfego e uso do tráfego são boas métricas da qualidade de serviço (QoS) em uma comunicação (LAFETÁ, 2016). Nesse contexto, este trabalho investiga uma versão multiobjetivo do problema de roteamento multicast. Nesta versão do problema, as árvores apresentadas como solução devem representar o melhor compromisso entre as métricas utilizadas.

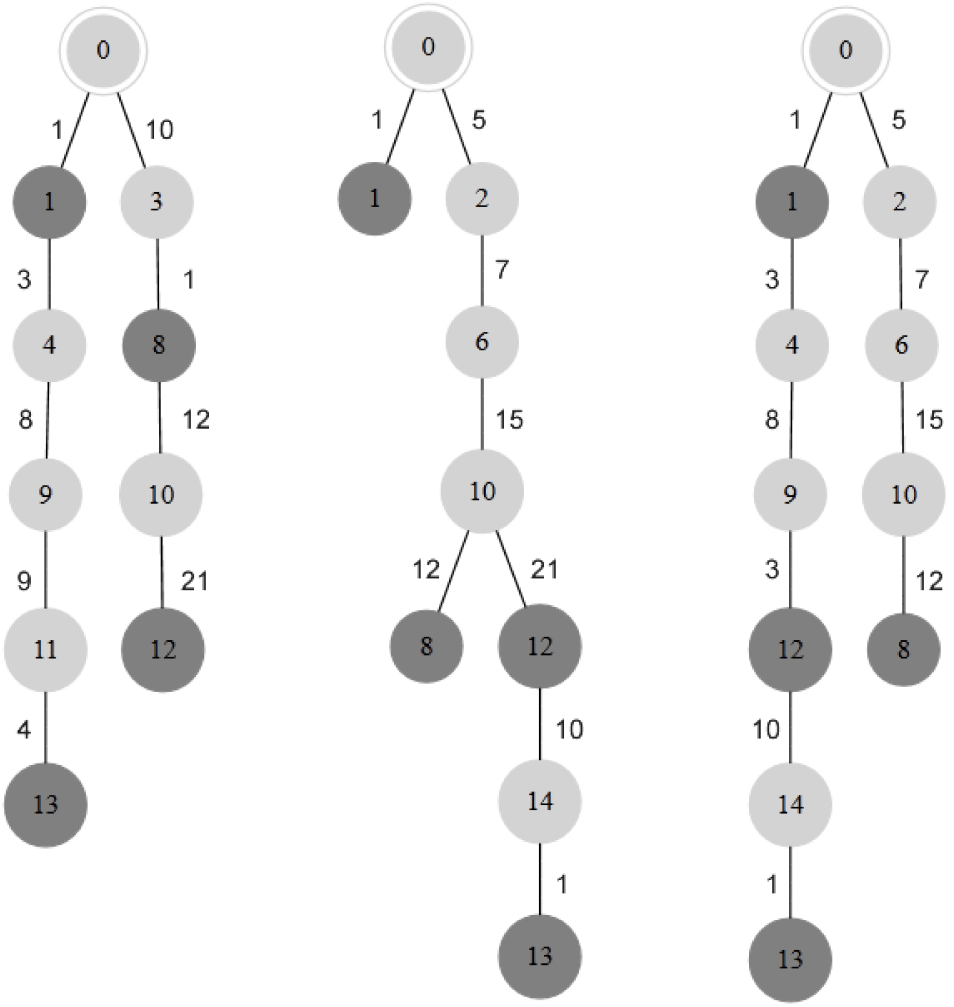

Figura 15 – Exemplos de árvores multicast relativos ao grafo da figura Figura 14. Retirado do trabalho de (LAFETÁ, 2016)

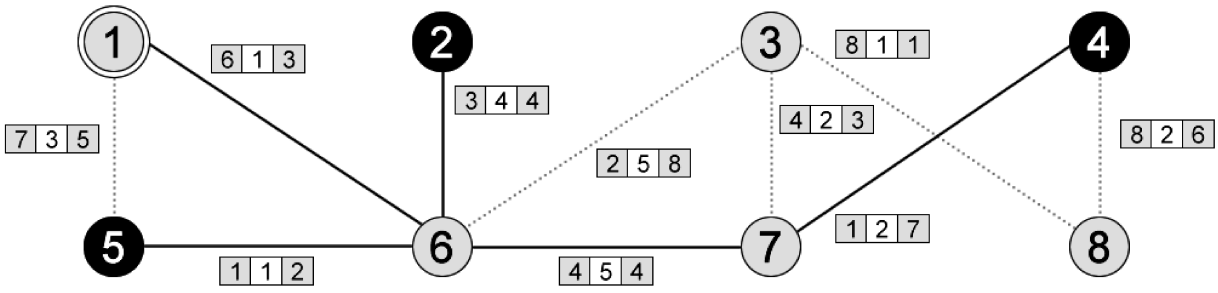

Figura 16 – Exemplo de árvore multicast no PRM multiobjetivo.

Na Figura 16 é apresentado um exemplo de rede com as métricas "custo" (primeiro valor), "delay" (segundo valor) e "tráfego corrente" (terceiro valor) nas arestas. As arestas representadas por linhas contínuas identificam uma árvore multicast ótima (não-dominada) para os objetivos "custo total", "delay-fim-a-fim médio" e "utilização média dos enlaces", descritos mais a frente no texto. Na figura, o nó 1 é a raiz e os nós com fundo escuro representam os nós destinos. Para esse exemplo, no objetivo "utilização média dos enlaces", considerou-se a mesma capacidade de tráfego para todos os enlaces, de forma que apenas o tráfego corrente fosse relevante.

Neste trabalho consideram-se até quatro valores de peso para um enlace da rede: custo, *delay*, capacidade de tráfego e tráfego corrente, representados respectivamente pelas funções: $c()$, $d()$, $z()$ e $t()$. Por meio dessas medidas são formulados os seguintes objetivos:

1. **Custo total:** soma dos valores de custo de todas as arestas da árvore.
2. **Delay fim-a-fim médio:** média da soma dos *delays* em cada ramo da árvore. Em outras palavras, média do atraso em cada uma das comunicações cliente-servidor.
3. **Delay fim-a-fim máximo:** maior valor para a soma de *delays* dentre todos os ramos da árvore, ou seja, o maior atraso dentre todas as comunicações cliente-servidor.
4. ***Hops count***: número de vértices na árvore.
5. **Utilização máxima de enlaces:** considerando todas as arestas na árvore, qual delas atinge a maior utilização de banda? Matematicamente, considerando E o conjunto de arestas da árvore e ϕ o tamanho da mensagem, essa métrica é dada por:
$$\max_{e \in E} \frac{t(e) + \phi}{z(e)} \tag{16}$$
6. **Utilização média dos enlaces:** média da utilização de banda entre todas as arestas da árvore. Seu cálculo é similar ao da métrica anterior (utilização máxima de enlaces) e é dado por:
$$\frac{\sum_{e \in E} \frac{t(e)+\phi}{z(e)}}{|E|} \tag{17}$$

A fim de possibilitar diversos cenários de teste para o PRM, os objetivos acima podem ser combinados de diversas maneiras, criando várias instâncias multiobjetivo do problema. Neste trabalho foram utilizadas 5 formulações (combinações) desses objetivos:

1. P_2: formado pelos objetivos 1 e 3.
2. P_3: formado pelos objetivos 1, 3 e 4.
3. P_4: formado pelos objetivos 1, 3, 4 e 5.

4. P_5: formado pelos objetivos 1, 3, 4, 5 e 6.

5. P_6: formado pelos objetivos 1, 2, 3, 4, 5 e 6.

Essas combinações de objetivo foram formuladas em trabalhos anteriores (LAFETÁ, 2016; BUENO, 2010) e, segundo esses estudos prévios, apresentam baixo grau de correlação. Ou seja, um problema de 6 objetivos realmente representa 6 variáveis distintas, onde uma não pode ser calculada como função da outra.

O PRM foi trabalhado sobre 5 redes diferentes variando a complexidade em termos de quantidade de nós destinos, vértices e arestas. Essas redes foram retiradas do trabalho de (LAFETÁ et al., 2016) e suas caraterísticas são apresentadas na Tabela 1.

Tabela 1 – Definições das redes utilizadas no PRM

Nome	Destinos	Vértices	Arestas
Rede 1 (R1)	10	33	106
Rede 2 (R2)	18	75	188
Rede 3 (R3)	37	75	188
Rede 4 (R5)	12	75	300
Rede 5 (R5)	16	100	250

Para resolver o PRM através de estratégias evolutivas, é necessário, em primeiro lugar, definir a modelagem do indivíduo. Esse processo não é muito simples, pois a solução não pode ser representada por um vetor. De fato, como deve representar os caminhos entre o servidor e os múltiplos destinos em uma rede de computadores, a solução para o PRM é melhor representada por uma árvore. Dessa forma, é preciso desenvolver o processo de crossover e de mutação de acordo com essa estrutura. O operador de cruzamento deve receber duas árvores pais e gerar uma nova árvore (filha) que compartilha características de ambos os pais. O operador de mutação precisa criar uma pequena alteração na árvore que permita explorar diferentes regiões do espaço de busca, mas que não a descaracterize completamente. As seções a seguir apresentam o modelo adotado nas abordagens evolutivas para o PRM.

Em relação à proposta utilizada em (LAFETÁ, 2016), destacam-se como diferenças a geração da população inicial e o cruzamento. Com relação à população inicial, os indivíduos gerados através do método descrito neste trabalho apresentam diversidade um pouco maior. No que diz repeito ao cruzamento, aqui se utiliza apenas o *crossover* por caminho, enquanto no trabalho anterior se utilizava também um outro tipo de cruzamento, o *crossover* por similaridade.

4.2.2 Representação da solução

Como mostrado na seção 4.2, considerando que em cada nó da rede a mensagem pode ser replicada e enviada aos próximos nós conectados, o PRM deseja encontrar a árvore

que representa o processo de transmissão de menor custo que parte do nó fonte (servidor) e atinge todos os destinos. Existem duas maneiras de se representar uma solução:

1. **Representação em árvore:** (BUENO; OLIVEIRA, 2010) o AG evolui a própria árvore que se deseja encontrar como solução. É um processo mais complicado que elimina a necessidade de pós-processamento. A Figura 15, mostra alguns exemplos de árvores multicasts.

2. **Representação em conjunto:** (CRICHIGNO; BARÁN, 2004) o AG evolui um conjunto de caminhos C, ou seja, para cada nó destino d, deve existir uma sequência de nós $L \in C$ que contém o caminho que leva do nó servidor (emissor) até o nó d. A representação em conjuntos de caminhos é mais fácil de se gerenciar, mas exige a transformação (conjunção) dos caminhos pertencentes ao conjunto em uma árvore ao final do processo. Como diferentes árvores podem ser formadas a partir de um único conjunto de caminhos, essa representação não é tão eficiente quanto a anterior ao explorar o espaço de busca.

Neste trabalho, optou-se por utilizar a representação em árvores.

4.2.3 Geração da população inicial

Considere um grafo da rede G, um nó raiz (também chamado de servidor ou emissor) r e um conjunto de nós de destino D. Para criar uma solução aleatória no PRM, inicia-se um grafo S apenas com o vértice r. O passo seguinte é extrair um destino aleatório $d \in D$ e, com base no grafo G, criar um caminho em S entre qualquer um de seus vértices atuais e d. Após construir o caminho, d com certeza será atingível a partir de qualquer vértice de S. O processo se repete até que todos os nós destinos estejam no grafo S. Por fim, o grafo S é transformado em uma árvore a partir da remoção dos ciclos presentes em S e da aplicação de podas dos ramos desnecessários.

Para criar o caminho aleatório entre um vértice de S e um nó destino d, considera-se um vetor de exploração Exp que, inicialmente, contém todos os nós de S. Enquanto o destino d não é encontrado, algum nó $exp_i \in Exp$ é escolhido e todos os vértices adjacentes (conectados) a exp_i são incluídos em Exp. Quando d é encontrado, adiciona-se a S o caminho necessário para sair de algum vértice de S e chegar ao destino d.

4.2.4 Cruzamento

A estratégia de cruzamento utilizada neste trabalho para o PRM é chamada de cruzamento por caminho e foi proposta em (LAFETÁ et al., 2016) como alternativa ao cruzamento por similaridade utilizado em trabalhos anteriores (BUENO; OLIVEIRA, 2010). Esse tipo de cruzamento é realizado entre duas árvores P_1 e P_2 e produz um único filho F.

O processo consiste em separar cada um dos pais em ramos e, então, para cada nó destino d, acrescentar a F o ramo de P_1 ou de P_2 que leva a d. A escolha entre P_1 e P_2 é feita de forma aleatória. Assim que todos os nós destinos são atingíveis em F, a etapa de seleção de ramos é interrompida, os ciclos que possivelmente foram criados são removidos, e um processo de poda é realizado a fim de remover qualquer nó folha que não seja destino.

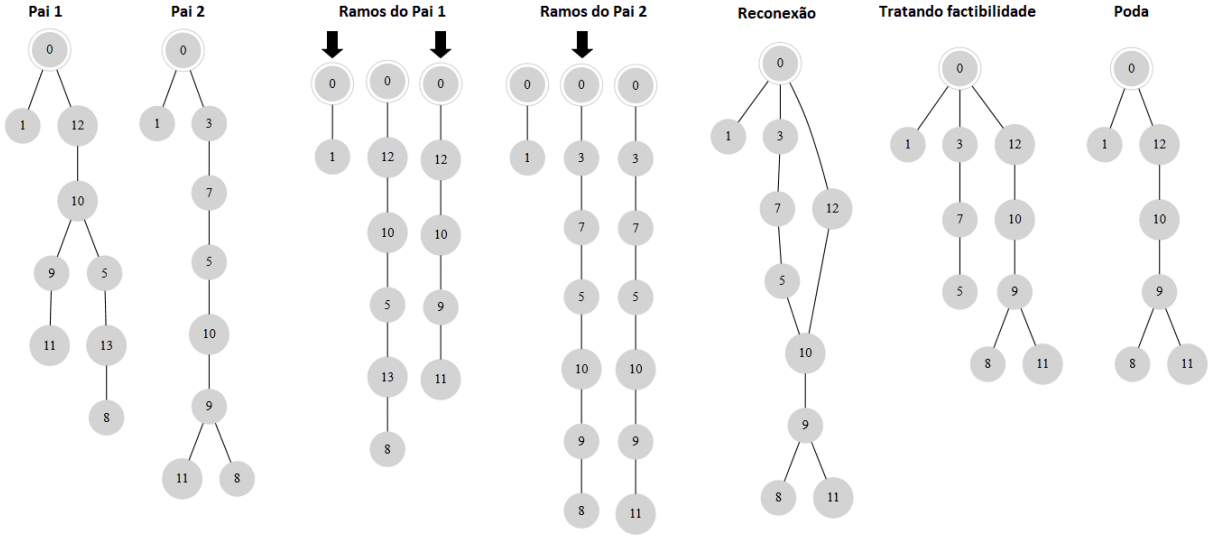

Figura 17 – Exemplo de cruzamento por caminho, onde o nó raiz é 0 e os destinos são {1, 8, 11}. Retirado de (LAFETÁ, 2016)

A Figura 17 representa o processo de cruzamento por caminho entre duas árvores (Pai 1 e Pai 2). No exemplo, o nó raiz é o vértice 0 e os nós destinos são o conjunto $\{1, 8, 11\}$. As setas em "ramos do pai 1" e "ramos do pai 2" representam os caminhos escolhidos em cada um dos pais para compor a árvore filha. O grafo nomeado "reconexão" representa o filho após a inclusão de todos os ramos. Como foi gerado um ciclo (existem dois caminhos da origem até o nó 10), o mesmo deve ser removido a fim de obter uma árvore válida. Supondo que o caminho $0 \rightarrow 12 \rightarrow 10$ tem menor custo que o caminho $0 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 10$, o nó 10 será acessado apenas a partir do nó 12. Em "tratando factibilidade", apresenta-se o filho após a remoção dos ciclos. Como os nós 3, 7 e 5 não são destinos, eles são removidos durante o processo de poda, resultando na última árvore ("poda") que é o filho.

Para remover os ciclos, percorre-se a árvore em largura removendo qualquer aresta que adicione ciclo. No processo de poda, verificam-se todas as folhas. Se algum nó folha não for um destino, ele é removido e o processo repetido até que todos os nós folhas sejam destinos.

Após o cruzamento, realiza-se um processo de filtragem, onde todas as soluções repetidas são substituídas por novos indivíduos gerados aleatoriamente.

4.2.5 Mutação

A mutação em uma árvore que representa uma solução para o PRM consiste em remover parte dos nós da árvore e então reconectá-los de maneira aleatória utilizando o grafo correspondente à rede em questão. As etapas desse processo estão descritas no algoritmo 3.

Algoritmo 3 Mutação para uma árvore $(A, G, qte_{arestas}, r, D)$

```
1: desconectar(A, qte_arestas)
2: C ← extraiComponente(A, r)
3: M ← novoGrafo()
4: M ← mesclar(M, C)
5: enquanto |D| > 0 faça
6:     d ← removeAleatorio(D)
7:     C ← extraiComponente(A, d)
8:     se d ∉ V(M) então
9:         P ← caminhoAlearorio(C, M, G)
10:        M ← mesclar(M, P)
11:    fim se
12:    M ← mesclar(M, C)
13: fim enquanto
14: removerCiclos(M)
15: podarArvore(M)
16: retorne M
```

O algoritmo recebe como entrada a árvore que sofrerá mutação (A), o grafo da rede (G), a quantidade de arestas a se remover na mutação ($qte_{arestas}$), o vértice raiz (r), e o conjunto de destinos (D). Na linha 1, desconecta-se a árvore através da remoção aleatória de $qte_{arestas}$ arestas. Entre as linhas 2 e 4, cria-se a estrutura para a construção da árvore (M). A partir da linha 5 começa o laço para reconectar todos os nós destinos à arvore. A linha 6 seleciona um destino aleatório d para reconectar. Na linha 7, a componente conexa contendo d é extraída de A e chamada de C. Se não existe componente conexa com o vértice d, C será um grafo com um único nó (d) e nenhuma aresta. Caso o vértice d não exista na componente conexa M, cria-se um caminho aleatório em M (com base no grafo G) que a conecta a C (linhas 8 a 10). A construção do caminho aleatório é feita nó a nó até se encontrar uma sequência de arestas entre as duas componentes. Na linha 12, incluem-se em M todas as arestas de C. Ao final, o mesmo pós-processamento do cruzamento por caminho é realizado: remoção de ciclos e poda da árvore (linhas 14 e 15).

Capítulo 5

Algoritmo proposto

O algoritmo proposto neste trabalho é baseado em colônia de formigas (ACO) e foi chamado de *Many-objective Ant Colony Optimization based on Non-Dominated Sets* (MACO/NDS), em português, otimização em colônia de formigas para muitos objetivos baseada em decomposição de feromônios (FRANçA; MARTINS; OLIVEIRA, 2018). A proposição envolve os seguintes módulos:

- Modelo PMM: construção da solução para o problema da mochila multiobjetivo, apresentado na seção 5.1.
- Modelo PRM: construção da solução para o problema do roteamento multicast, apresentado na seção 5.2.
- *Framework* ACO: algoritmo geral voltado à manipulação de problemas *many-objective* desenvolvido para se trabalhar com qualquer problema discreto, apresentado na seção 5.3.

Este capítulo apresenta a construção da solução no ACO para ambos os problemas (PMM e PRM). Os modelos de construção, apesar de terem sido propostos para o MACO/NDS, podem ser utilizados em qualquer algoritmo ACO. O *framework* MACO/NDS, componente principal do algoritmo, é descrito em seguida na seção 5.3. Nela são discutidos alguns conceitos gerais sobre a implementação multiobjetivo do ACO, bem como uma descrição mais detalhada das etapas de construção das soluções e atualização dos feromônios.

5.1 Construção da solução para o PMM

Como mostrado na seção 4.1.2, os AGs utilizam vetores binários para representar as soluções do PMM. Nos ACOs, é possível utilizar a mesma representação, entretanto, deve-se desenvolver um algoritmo que constrói a solução a partir de uma estrutura de feromônios e uma heurística. O processo de gerar as soluções é mais simples nos AGs,

pois basta efetuar as operações de *crossover* e mutação sobre os vetores binários. No entanto, a implementação de um ACO para o PMM pode ser desafiadora, já que as colônias de formigas foram inicialmente propostas para lidar com grafos e não vetores de bits. Diversos trabalhos que adotam ACOs para a resolução do problema da mochila são encontrados na literatura (KE et al., 2010; KONG; TIAN; KAO, 2008; CHANGDAR; MAHAPATRA; PAL, 2013; ALAYA; SOLNON; GHÉDIRA, 2004; ALAYA; SOLNON; GHEDIRA, 2007).

As colônias de formigas foram propostas inicialmente para problemas em grafos, e, portanto, não é intuitivo sua adaptação para outros tipos de problemas, como o problema do mochila. Entretanto, como mostrado em (KE et al., 2010), não é necessário ter um grafo para se utilizar a técnica, sendo possível manipular os feromônios de diversas outras formas. Foram encontradas na literatura três diferentes formas de lidar com os feromônios para o problema da mochila multiobjetivo (PMM). A primeira abordagem consiste em depositar feromônios em cada um dos itens (LEGUIZAMON; MICHALEWICZ, 1999). Sempre que um item é escolhido para compor a solução, incrementa-se a quantidade de feromônios presente no item. Dessa forma, a quantidade de feromônio em cada item representa sua preferência em relação aos demais. A segunda abordagem propõe a criação de um grafo direcionado que representa a preferência em incluir um item b após a inclusão do item a (FIDANOVA, 2003). Dessa forma, sempre que se escolher um item a e posteriormente um b, deposita-se feromônio na aresta (a, b) do grafo. Ao construir uma solução, analisa-se o último item selecionado e todas as arestas no grafo que partem dele. O destino com a maior quantidade de feromônios representa o item preferível. A terceira estratégia foi proposta por (ALAYA; SOLNON; GHÉDIRA, 2004) e utiliza um conceito de aresta semelhante à abordagem anterior. Entretanto, ao invés de depositar feromônios apenas em pares consecutivos, eles são depositados para todos os pares de objetos existentes na solução. Por exemplo, se na solução os itens a, d e f estão na mochila e no passo atual adiciona-se o item c, o depósito de feromônios é feito nas arestas (a, c), (d, c) e (f, c), de forma que o grafo represente a preferência de se escolher um item dado que algum outro item já tenha sido selecionado. Assim, ao construir a solução, deve-se analisar todas as arestas com origem em algum item já presente na solução, o destino da aresta com maior quantidade de feromônios representa o item mais desejável. Neste trabalho utilizou-se a primeira estratégia que deposita os feromônios diretamente nos itens.

Além dos feromônios, outra importante fonte de informação para se construir uma solução no ACO é a heurística. Ela é responsável por estimar o quão vantajoso é escolher um item em relação a outro. Tomando como inspiração o estudo de (KE et al., 2010), a implementação do ACO para o PMM utiliza como heurística o seguinte modelo de funções:

- Para cada objetivo k (lucro do item), cria-se uma função de heurística h_k que recebe dois parâmetros, a capacidade restante (cr) e o item que se deseja incluir. A

capacidade restante é a diferença entre a capacidade da mochila e a soma dos pesos dos itens que já foram incluídos. A heurística é então dada por:

$$h_k(item, cr) = lucro_k(item) \times (1 - \frac{peso(item)}{cr}) \tag{18}$$

- Uma heurística extra é usada exclusivamente para se referir ao peso do item, a qual é dada por:

$$h_{peso}(item) = 1 - \frac{peso(item)}{peso\_maximo} \tag{19}$$

Logo, o número de heurísticas no PMM, por item, é sempre o número de objetivos + 1. Tendo definido a estrutura de feromônios e as heurísticas, para construir uma solução, o algoritmo recebe a função heurística (h) e um vetor de feromônios (τ). A heurística h é normalmente uma combinação de todas as heurísticas. No caso do MACO/NDS, h é uma média ponderada das heurísticas $h_1, h_2, .., h_k$ e h_{peso}, esse processo é detalhado na Seção 5.3.1.

Para escolher o próximo item da solução, uma formiga analisa todas as possibilidades e então toma uma decisão com base nas probabilidades indicadas pelos feromônios e pelas heurísticas. Para avaliar as possibilidades, utiliza-se uma lista de exploração Exp. A lista de exploração Exp contém todos os itens possíveis, a probabilidade p de cada item $exp_i \in Exp$ é calculada de acordo com a heurística de exp_i ($h(exp_i)$) e a quantidade de feromônios no item τ_i, como mostrado na fórmula a seguir:

$$p(exp_i) = \frac{\tau_i^\alpha + h(exp_i)^\beta}{\sum_{j \leftarrow 0}^{tamanho(Exp)} \tau_j^\alpha + h(exp_j)^\beta} \tag{20}$$

O próximo item na solução é escolhido de acordo com as probabilidades calculadas em uma espécie de roleta, onde, quanto maior o valor de probabilidade ($p(exp_i)$), maior a chance do item ser escolhido. Após ter selecionado um item, recalcula-se a lista Exp para que contenha apenas itens válidos para a solução, isto é, remove-se qualquer elemento de Exp que, se incluído na solução, excede a capacidade da mochila. O processo é repetido até que Exp seja uma lista vazia.

A fim de reduzir a complexidade do algoritmo de construção da solução para o PMM, ao invés de se utilizar a lista completa de itens possíveis (Exp), utiliza-se uma amostra desse conjunto (Exp'). Exp é calculado normalmente, mas ao submeter-se o conjunto para o cálculo de probabilidades e roleta, no lugar de Exp, é utilizado Exp', uma amostra aleatória de tamanho fixo Exp'_{size} (parâmetro do algoritmo) da lista completa Exp.

5.2 Construção da solução para o PRM

A modelagem de um algoritmo genético para o problema do roteamento multicast não é trivial, pois deve-se representar os caminhos entre o servidor e os múltiplos destinos em

uma rede de computadores, ou seja, uma árvore. Em contrapartida, o PRM se aproxima da definição original do ACO, pois trabalha com grafos. Dessa forma, a modelagem para o ACO do PRM é mais natural que a modelagem para o AG.

O processo de construção da solução para um ACO do PRM deve gerar a árvore *multicast* com base no grafo da rede, da estrutura de feromônios e da heurística. Existem diversas maneiras de se criar tal árvore. A fim de estudar o comportamento das diferentes estratégias possíveis, aprimorá-las e determinar o melhor modelo para se utilizar no PRM, analisou-se o comportamento de cada ideia no caso mais básico possível: o PRM mono-objetivo. As ideias consideradas neste trabalho são listadas a seguir:

1. A primeira estratégia, apresentada em (PINTO; BARAN, 2005), pode ser vista como uma formiga que caminha pelo grafo até encontrar todos os destinos. A análise é feita passo a passo, considerando apenas a vizinhança do vértice corrente como possibilidades para compor a solução. O processo inicia com uma lista de exploração de vértices (Exp_v) que contém um único elemento: a raiz. Sorteia-se um vértice $i \in Exp_v$ e atribui-se probabilidades a todas as arestas (i, j), onde j é qualquer vértice não-visitado na adjacência de i, sendo que i é removido de Exp_v caso não possua vizinhos factíveis. Um vértice v é escolhido de acordo com as probabilidades calculadas (conforme à seção 2.2.2), a aresta (i, v) é adicionada à solução e v é inserido na lista de exploração Exp_v. O processo é repetido até que todos os destinos tenham sido alcançados. Como etapa final, poda-se a árvore, eliminando as folhas que não representam destinos.

2. A segunda estratégia, até onde se sabe, proposta neste trabalho, manipula de forma independente os caminhos entre a raiz e cada um dos destinos. Portanto, nessa estratégia, $|D|$ formigas partem da raiz e, com base nas heurísticas e feromônios, encontram, cada uma, um vértice $d \in D$ diferente, sendo D o conjunto de nós destinos. Com os caminhos para todos os nós destinos definidos, monta-se a árvore evitando a ocorrência de ciclos. Por fim, realiza-se a poda da árvore para excluir qualquer vértice folha que não seja um destino.

3. Uma terceira estratégia é proposta neste trabalho. Ela adota uma representação de formiga fictícia que pode estar em vários locais do grafo ao mesmo tempo (formiga com sobreposição quântica). Dessa forma, é possível analisar as probabilidades de todas as arestas factíveis ao mesmo tempo, ao invés de sempre escolher a composição da solução de acordo com uma vizinhança local. Ao considerar todas as possibilidades ao mesmo tempo, obtém-se um processo melhor de decisão que não apresenta qualquer tendência (viés) para algum dos ramos da árvore, já que, a todo momento, qualquer vértice factível pode ser incluído no resultado. O processo é iniciado a partir de uma lista de exploração de arestas (Exp_a) que contém todas as arestas conectadas ao nó raiz. A cada passo, calcula-se as probabilidades para

toda aresta de Exp_a e, de acordo com os valores obtidos, escolhe-se $exp_i \in Exp_a$ para compor a solução. exp_i é então removida de Exp_a, adicionada à solução e tem todas suas arestas adicionadas à lista de exploração, desde que ainda não tenham sido incluídas. O processo é repetido até que todos os destinos sejam atingidos. Uma poda é realizada ao final do processo. Uma outra analogia para esse processo seria imaginar que a formiga é clonada sempre que visita um vértice. Dessa forma, haverá uma formiga em cada vértice já visitado e então a vizinhança de todos eles será considerada ao mesmo tempo.

4. A última estratégia, também proposta e avaliada neste trabalho, consiste em construir a solução de forma inversa. Ao invés de partir da raiz e chegar aos destinos, este modelo propõe que se utilizem $|D|$ formigas, onde cada uma tem como posição inicial um nó destino $d \in D$ diferente. A ideia é que as formigas escolham seus caminhos localmente com base nas probabilidades das arestas em suas vizinhanças. Sempre que uma formiga encontrar o caminho que já foi explorado por outra formiga, ela para de explorar e segue os mesmos passos realizados pelo outro agente. O processo termina quando o nó raiz foi encontrado por alguma formiga e todas as formigas tiverem se encontrado em algum vértice. Em outras palavras, o processo inicia com uma componente conexa para cada $d \in D$. Em todo passo do algoritmo, cada componente conexa é explorada a partir do último nó adicionado. Nesse processo, calculam-se as probabilidades das arestas na vizinhança do nó explorado e escolhe-se, de acordo com os valores obtidos, um novo vértice v para ser adicionado à componente. Se não há arestas factíveis na vizinhança, deve-se retroceder para um vértice anterior. Caso o nó incluído v pertença a uma outra componente conexa, elas se unem. O processo termina quando existe apenas uma componente conexa e o vértice raiz foi encontrado. Por fim, uma poda é realizada como pós-processamento da árvore.

Cada uma das estratégias mencionadas foi implementada e avaliada a fim de determinar aquela que produz as soluções de melhor qualidade para o PRM. Para obter um valor como referência de qualidade, também foi implementada uma versão modificada do algoritmo de Prim (PRIM, 1957) que aproxima a árvore *multicast* de menor custo. Como esperado, o algoritmo de Prim modificado, apesar de nem sempre conseguir os melhores resultados, é a opção mais rápida. Entretanto, a intenção deste estudo não é produzir um algoritmo melhor para o problema mono-objetivo, mas avaliar as estratégias de construção de solução com o objetivo de escolher aquela que será empregada em versões mais complexas (multiobjetivo) do PRM.

Os resultados dos experimentos, que podem ser encontrados na seção 6.3, mostram que nossa estratégia (formiga com sobreposição quântica) representa a melhor relação entre

qualidade do resultado e tempo de execução. Portanto, para todos os demais experimentos no PRM, foi o método de construção da solução utilizado.

Durante a implementação da estratégia, foram adotadas algumas técnicas de amostragem a fim de agilizar o algoritmo, sem afetar significativamente a qualidade das soluções. O fluxograma apresentado na Figura 18 mostra, de maneira geral, o funcionamento do algoritmo. A descrição detalhada dessa implementação é apresentada no Algoritmo 4, que recebe como parâmetros de entrada:

- G: o grafo que representa a rede;
- r: o nó raiz, servidor de onde parte a mensagem;
- D: conjunto de nós destinos;
- τ: estrutura de feromônios;
- h: função heurística;
- Exp'_{size}: tamanho da amostra.

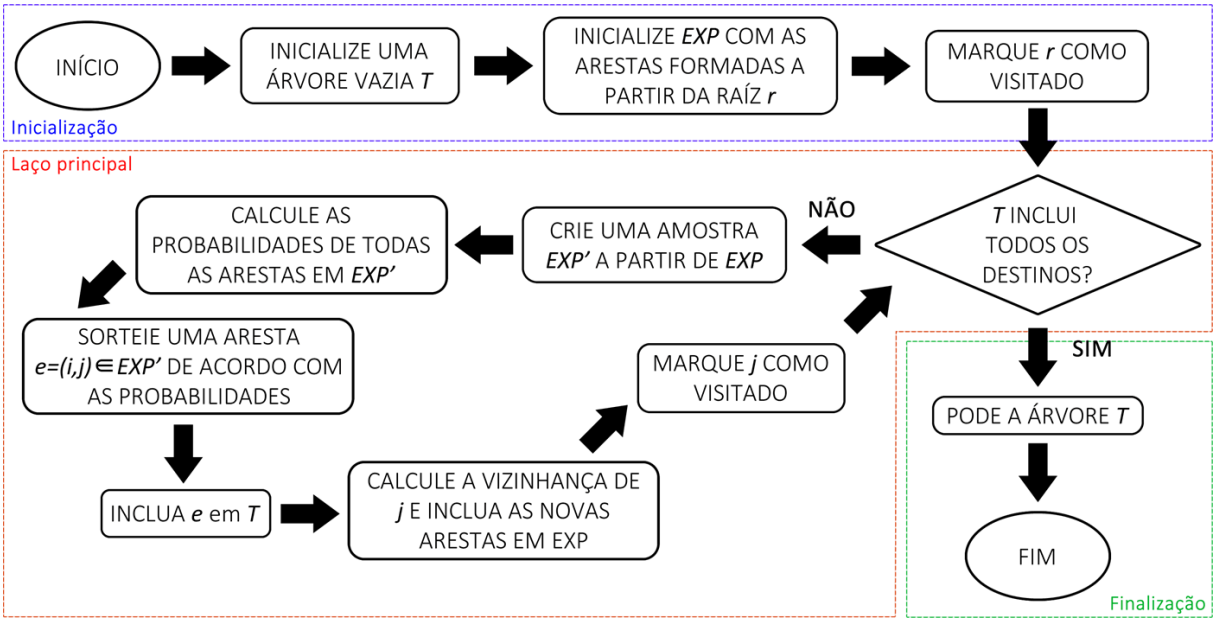

Figura 18 – Método de construção de soluções do ACO para o PRM

Um exemplo gráfico considerando um PRM simples é mostrado na Figura 19. No grafo apresentado na figura, A é o vértice raiz, enquanto $\{B, D, F, G\}$ são os destinos. No passo 1, a formiga é colocada na raiz (A) e os vértices na vizinhança de A se tornam candidatos a fazerem parte da árvore. Em cada passo a formiga escolhe o caminho a perseguir de acordo com os feromônios, a heurística e uma função de probabilidade. Em nosso exemplo, esse processo não é ilustrado, sendo apresentado apenas o caminho escolhido. No passo 2,

Algoritmo 4 Geração de solução no ACO $(G, r, D, \tau, h, Exp'_{size})$

1: Inicie uma árvore vazia T
2: Inicie Exp_a com todas as arestas que possuem alguma extremidade em r
3: Marque r como visitado
4: **enquanto** T não incluir todos os vértices em D **faça**
5: Crie uma amostra aleatória Exp' a partir da lista Exp_a com Exp'_{size} elementos
6: Calcule as probabilidades de todas as arestas em Exp' de acordo com τ e h
7: Escolha uma aresta $e = (i, j) \in Exp'$ de acordo com as probabilidades
8: Inclua e em T
9: Marque j como visitado
10: Calcule a vizinhança V do vértice j
11: **para** $v \in V$ **faça**
12: **se** já existe aresta a em Exp_a que leva a v **então**
13: Remova a de Exp_a
14: Calcule as probabilidades de a e de (j, v) de acordo com τ e h
15: Sorteie uma das duas arestas de acordo com as probabilidades e adicione a vencedora em Exp_a
16: **senão se** v não tiver sido visitado **então**
17: Inclua (j, v) em Exp_a
18: **fim se**
19: **fim para**
20: **fim enquanto**
21: Pode a árvore T
22: **retorne** T

a formiga escolhe o vértice B e passa a enxergar a vizinhança de B como candidata, mas diferentemente de métodos tradicionais, não deixa de considerar os vértices descobertos em A. Ainda no passo 2, verifica-se que o vértice C pode ser atingido tanto por A quanto por B. Este algoritmo não mantém duas arestas que levam ao mesmo destino, portanto uma delas deve ser retirada da lista de candidatos. Tal escolha é feita de acordo com o cálculo de probabilidade. Por exemplo, considerando que a aresta (A, C) tem uma probabilidade menor que a aresta (B, C), ela é removida. No passo 3, a formiga escolhe ir para C e agora enxerga toda a vizinhança de A, de B e de C. No passo 4, dentre os vértices $\{D, E, F, G\}$ a formiga escolhe partir para D. No passo 5, a formiga adiciona a aresta (A, G) à solução e, como no passo 2, a aresta (G, F) é escolhida em detrimento da (C, F), que é removida da exploração. No último passo, a formiga, que neste momento enxerga os vértices $\{E, F\}$ escolhe ir para F. Como todos os destinos foram adicionados à árvore, o processo termina. Em problemas mais complexos, pode ser necessário realizar uma poda, mas neste caso, a solução está completa.

Trabalhar com todas as arestas possíveis é demasiadamente caro e inviável para um algoritmo em que se deseja bom desempenho quanto ao tempo de execução. Por isso, trabalha-se com uma amostra da lista de exploração (linha 5 do algoritmo). Outro processo de simplificação também é empregado para evitar o crescimento exagerado de Exp_a durante a etapa de exploração, nas linhas 12 a 16. Nesse processo, caso um novo vértice

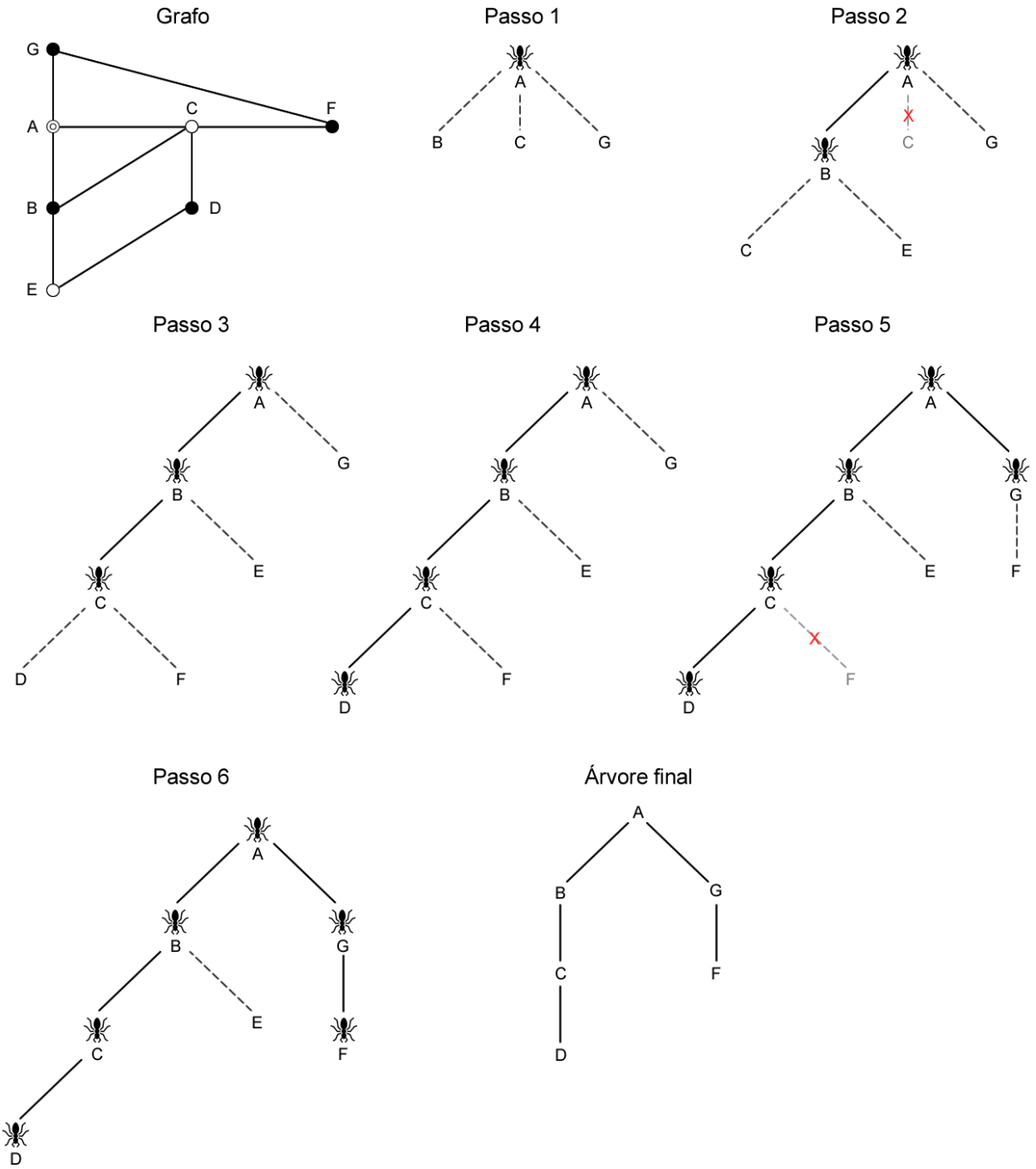

Figura 19 – Exemplo de como construir uma solução para o PRM

descoberto já seja atingível a partir de alguma aresta em Exp_a, mantém-se em Exp_a apenas uma das arestas. Para escolher qual das arestas manter, calculam-se as probabilidades de acordo com os feromônios (τ) e a heurística (h).

Com relação às heurísticas no PRM, uma função é construída para cada métrica de rede presente na formulação de objetivos. Por exemplo, no problema P_4, cujos objetivos envolvem as métricas custo, *delay*, tráfego e capacidade, quatro heurísticas são criadas:

$$\begin{cases} h_1(e) = 1 - custo(e) \\ h_2(e) = 1 - delay(e) \\ h_3(e) = 1 - trafego(e) \\ h_4(e) = capacidade(e) \end{cases}$$

Onde $custo(e)$, $delay(e)$, $trafego(e)$ e $capacidade(e)$ correspondem, respectivamente, ao valor normalizado entre 0 e 1 para o custo, *delay*, tráfego e capacidade da aresta e. Na heurística h_4 não é feito o complemento da função, pois essa é a única métrica que deve ser maximizada.

5.3 *Framework* MACO/NDS

O MACO/NDS se baseia na ideia do AEMMD de se criar conjuntos de soluções não dominadas para diferentes combinações possíveis de objetivos, adaptando-a para um modelo de colônia de formigas.

A multiplicidade de objetivos impõe que algumas mudanças sejam feitas na ideia original do ACO. Isso envolve responder as seguintes perguntas:

- Como representar os diferentes objetivos no depósito de feromônio?
- Como compor as múltiplas heurísticas em uma única função?

De acordo com (ALAYA; SOLNON; GHEDIRA, 2007), essas questões podem ser resolvidas através de uma das seguintes opções:

- **Várias colônias e múltiplos feromônios** $(m+1, m)$**:** considerando m o número de objetivos, esse modelo utiliza $m+1$ colônias de formigas e m estruturas de feromônios. Cada colônia é associada a um único objetivo e otimiza apenas esse objetivo através de sua própria estrutura de feromônios. Uma colônia adicional, que representa o conjunto de todos os objetivos, utiliza os feromônios das outras colônias de forma aleatória. Ao criar uma solução, sorteia-se alguma das estruturas de feromônios para se utilizar a cada passo. Outra proposta, no modelo $(m+1, m)$ é utilizar a soma de todas as estruturas de feromônios ao criar uma solução na colônia

extra. Quanto às heurísticas, cada colônia utiliza a função referente a seu objetivo, enquanto a colônia extra utiliza o somatório de todas as funções de heurística.

- **Colônia e feromônio únicos** $(1, 1)$**:** este modelo utiliza apenas uma colônia de formigas e uma estrutura de feromônios. Assim como no modelo anterior, as heurísticas são somadas para montar uma solução. A estrutura única de feromônios representa todos os objetivos. A atualização dos feromônios se dá através de um arquivo que mantém todas as soluções não-dominadas encontradas. Toda solução não-dominada contribui com a mesma quantidade de feromônios, já que, de fato, são indiferenciáveis em termos de qualidade.

- **Única colônia e vários feromônios** $(1, m)$**:** considerando m o número de objetivos, esse modelo utiliza uma colônia de formigas e m estruturas de feromônios. Assim como nos demais modelos, as heurísticas são somadas para montar uma solução. A cada passo da construção da solução, sorteia-se aleatoriamente a estrutura de feromônios a se utilizar. Cada estrutura de feromônios representa um objetivo e sua atualização é feita a cada iteração do algoritmo, de acordo com a melhor solução encontrada considerando aquele objetivo.

Em (ALAYA; SOLNON; GHEDIRA, 2007), os experimentos mostraram que o terceiro modelo $(1, m)$ gera os melhores resultados quando aplicado ao problema da mochila multiobjetivo. Dentre os ACOs multiobjetivos investigados, o MOACS se encaixa na segunda abordagem (1,1), adaptando apenas o modo de lidar com a heurística (soma ponderada com pesos aleatórios), enquanto o MOEA/D-ACO adota a primeira estratégia, pois trabalha com múltiplas formigas e várias estruturas de feromônios, apesar da quantidade exata de formigas e tabelas de feromônios não ser ditada somente pelo número de objetivos (depende também do parâmetro H, ou número de divisões, passado ao algoritmo).

O algoritmo proposto nesta dissertação (MACO/NDS), não pertence a nenhuma das categorias apresentadas por (ALAYA; SOLNON; GHEDIRA, 2007), mas se aproxima da estratégia (1,m), pois lida com uma única colônia e múltiplos feromônios. A grande diferença é que o número de estruturas de feromônio não é igual à quantidade de objetivos, mas ao número de possíveis combinações entre os objetivos, de forma semelhante ao que ocorre no AEMMD (LAFETÁ et al., 2018). Na verdade o modelo é (1, $|P|$), onde P é o conjunto de subproblemas e seu tamanho é definido por:

$$|P| = \sum_{i=2}^{m} C_m^i \tag{21}$$

Sendo, m a quantidade de objetivos considerados, i a quantidade de objetivos considerados na combinação e C a quantidade de possibilidades combinando os objetivos i a i.

Além disso, o processo de cálculo da heurística e de atualização dos feromônios também se difere substancialmente da ideia original do modelo $(1, m)$. O MACO/NDS, se baseia na ideia de decomposição, ou seja, ao invés de trabalhar diretamente com todos os objetivos, o problema é atacado em várias frentes com quantidades reduzidas de funções. Desta forma, evita-se o problema de classificação da população devido ao alto número de soluções não-dominadas em espaços de dimensionalidade alta (DEB; JAIN, 2014). Os feromônios no MACO/NDS são encapsulados por uma estrutura chamada "subproblema", que contém as seguintes propriedades:

- **Objetivos:** determina os objetivos do subproblema em questão. É um vetor binário de tamanho m (número de objetivos), onde o bit 1 representa um objetivo que faz parte do problema e o bit 0 indica aqueles que não pertencem ao problema.
- **Feromônios:** corresponde aos valores dos feromônios em si, os quais são inicializados com o menor valor possível (parâmetro de entrada).
- **Arquivo:** conjunto de soluções não-dominadas que apareceram até o momento para o subproblema em questão.
- **Convergência:** indica a convergência do arquivo. Começa em 0 e incrementa em 1 sempre que o arquivo sofre alterações durante uma iteração (época).
- β**:** importância da heurística no cálculo de probabilidades ao construir uma solução. Um valor para β é informado como parâmetro de entrada do MACO/NDS e representa o valor inicial desse parâmetro (β) em cada subproblema.

Cada subproblema é responsável por uma combinação de objetivos. O funcionamento básico do MACO/NDS é ilustrado na Figura 20 e descrito no 5.

Algoritmo 5 Algoritmo geral do MACO/NDS

```
1:  P ← criarSubproblemas()
2:  P_ativo ← {P[0], P[1], ..., P[4]}
3:  sgi ← tamanho(P) − 1 // índice do subproblema geral
4:  enquanto condição de parada não for atingida faça
5:      S ← criarSolucoes(P_ativo)
6:      atualizarArquivo(P[sgi], S)
7:      S_nd ← P[sgi].arquivo
8:      atualizarSubproblemas(P_ativo, S_nd)
9:  fim enquanto
10: retorne S_nd
```

O primeiro passo do MACO/NDS (linha 1 do Algoritmo 5) é criar as estruturas de dados para os subproblemas $P = \{p_1, p_2, ..., p_i, ..., p_n\}$, uma para cada combinação possível de objetivos (análogo ao processo do AEMMD apresentado na seção 3.1.7).

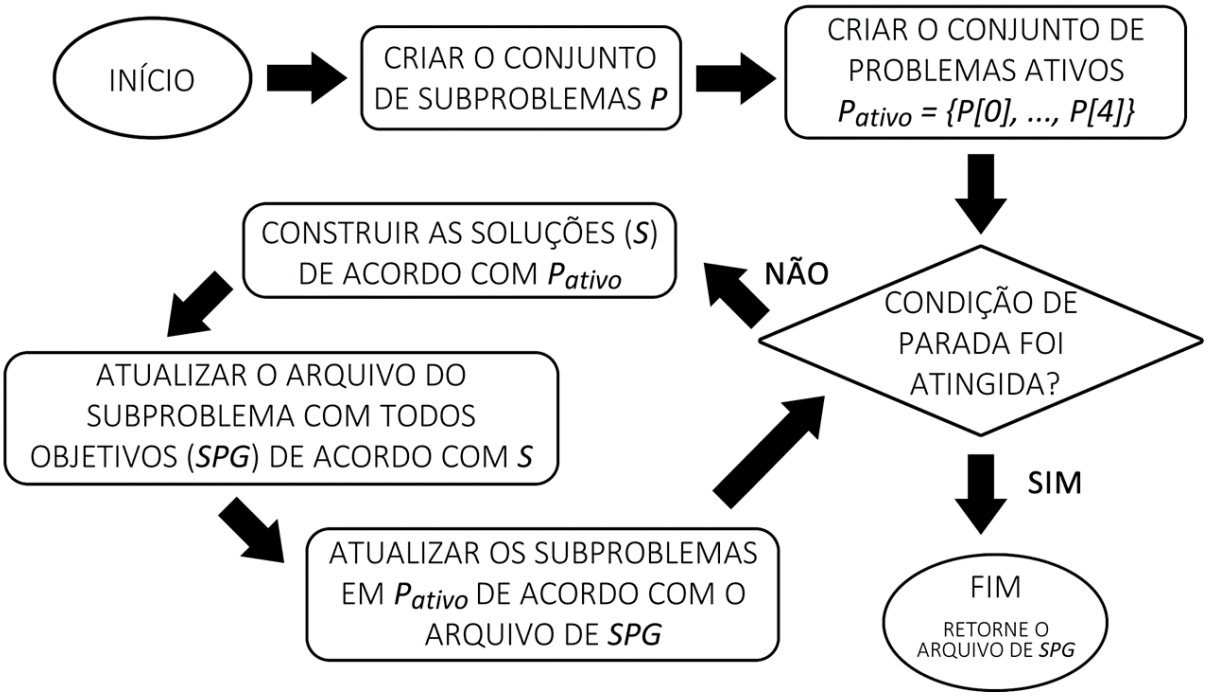

Figura 20 – Fluxo geral de execução do MACO/NDS

Para um problema de seis objetivos ($m = 6$), por exemplo, são criados 57 subproblemas que decompõem o problema principal. São 57 subproblemas, pois, se $m = 6$, $|P| = \sum_{i=2}^{m} C_m^i = 15 + 20 + 15 + 6 + 1 = 57$.

Os subproblemas são utilizados no processo de construção da solução e devem ser atualizados em toda iteração do algoritmo. Manipular todos os elementos de P (57 subproblemas, no caso de seis objetivos) é computacionalmente caro e inviável. Por essa razão, define-se um número máximo de subproblemas para se trabalhar em um dado momento. Em ambos os problemas utilizados neste trabalho (PMM e PRM), foi utilizado um limite de cinco subproblemas ativos simultaneamente. A ordem de ativação dos elementos de P é determinada pela sua ordem. Por isso, é importante a sequência em que as estruturas são criadas: os primeiros subproblemas instanciados são aqueles que representam um menor número de objetivos, ou seja, combinações 2 a 2. Em seguida, instanciam-se os subproblemas de 3 objetivos e, assim por diante, até que o último subproblema, com o número total de objetivos, seja criado.

Na linha 2 do algoritmo, os cinco subproblemas mais simples são ativados (ou seja, as 5 primeiras combinações de objetivos). No processo de atualização (linha 8), assim que algum dos subproblemas passa a não contribuir para o conjunto de soluções, esse é desativado em favor do próximo subproblema. Dessa forma, o conjunto de soluções do MACO/NDS cresce gradualmente, partindo das decomposições mais simples, que utilizam um número menor de objetivos, em direção às mais complexas, que utilizam um número maior, até que seja ativado o subproblema que lida com todos os m objetivos.

Na linha 3, a variável *sgi* é usada para armazenar o índice do subproblema que considera todos os objetivos (último subproblema de P). O laço principal do ACO, descrito entre as linhas 4 e 9 do algoritmo, consiste em gerar soluções e atualizar as estruturas de subproblema. Inicialmente (linha 5), é construído o conjunto de soluções S de acordo com os feromônios dos subproblemas em P_{ativo}, esse processo é detalhado na seção 5.3.1. Na linha 6, o arquivo do último subproblema (com todos os objetivos) é atualizado com as soluções em S. Na linha 7, esse arquivo é armazenado em S_{nd}, como referência ao arquivo principal com todas as soluções não dominadas envolvendo o problema original (todos os objetivos). Por último, na linha 8, as soluções de S_{nd} são utilizadas para atualizar cada uma das estruturas de subproblema em P_{ativo}. No final, o algoritmo MACO/NDS retorna todas as soluções não dominadas encontradas (S_{nd}).

5.3.1 Construção das soluções

As soluções em colônias de formigas são construídas a partir dos feromônios do subproblema ativo (P_{ativo}), das heurísticas (H) e dos valores de α e β. Os feromônios são criados e atualizados no decorrer do algoritmo, enquanto os demais são parâmetros de entrada. A heurística é uma função que estima a qualidade de uma parcela da solução, ex: arestas em problemas que envolvam grafos, como o PRM, e itens em problemas representados por vetores, como o PMM. Enquanto o parâmetro α determina a importância do feromônio ao tomar uma decisão a respeito da composição da solução, β representa a importância da heurística.

No MACO/NDS existem múltiplos feromônios (encapsulados pelas estruturas de subproblema) e heurísticas. Portanto, para que se possa construir uma solução, é necessário antes escolher quais valores de feromônio e qual função de heurística serão utilizados.

Os subproblemas utilizados no processo de construção das soluções pelas formigas é a lista de subproblemas ativos do MACO/NDS (P_{ativo}). Ou seja, 5 subproblemas são utilizados por vez para construir o conjunto de soluções S com n elementos. Logo, cada subproblema servirá de base para construir $n/5$ soluções de S.

Com relação às heurísticas, o MACO/NDS utiliza o mesmo processo proposto em (RIVEROS et al., 2016). Admite-se um grau de importância para cada função: 0 (não importante), 1 (importante) ou 2 (muito importante). O valor de importância é sorteado para cada heurística e funciona como um peso. A função única utilizada para construir a solução é a soma ponderada das heurísticas considerando os pesos sorteados.

O processo geral para a construção das soluções pelas formigas é exposto no Algoritmo 6.

O Algoritmo 6 apresenta o processo geral para se construir um conjunto de soluções. Na linha 1, S é inicializado como um conjunto vazio. A partir da linha 2, um laço é executado para gerar todas as soluções necessárias. Nas linhas 3 a 6, $|H|$ valores entre 0 e 2 (inclusive) são sorteados para um vetor de pesos W. Na linha 7, é criada a função

Algoritmo 6 Construção das soluções

1: $S \leftarrow \emptyset$
2: **para** $i \leftarrow 0$ até n **faça**
3: $W \leftarrow \emptyset$
4: **para** $j \leftarrow 0$ até $|H| - 1$ **faça**
5: $W[j] \leftarrow random(0, 2)$
6: **fim para**
7: $h(x) \leftarrow \sum_{i \leftarrow 0}^{|H|-1} \frac{H[i](x) * W[i]}{\sum_{w \in W} w}$
8: $p \leftarrow P_{ativo}[i \pmod{|P_{ativo}|}]$
9: $s \leftarrow gerarSolucao(p.feromonios, h, \alpha, p.\beta)$
10: $S \leftarrow S \cup \{s\}$
11: **fim para**
12: **retorne** S

heurística correspondente à combinação de todas as heurísticas de acordo com os pesos sorteados. Na linha 8, p recebe o subproblema que será utilizado na construção da solução i. A escolha do subproblema é feita de forma sequencial e circular, de modo a garantir uma distribuição uniforme entre os indivíduos do conjunto de soluções S. A nova solução é criada por uma formiga na linha 9 de acordo com os feromônios de p, a função heurística h, o parâmetro do α do algoritmo e o valor β de p. Na linha 10, a nova solução é incluída no conjunto de soluções S. Por fim, na linha 12, retorna-se o conjunto S de soluções.

Na linha 9 do algoritmo 6 constrói-se a solução em si. Esse processo depende exclusivamente do problema em questão e representa a principal parte na elaboração do modelo para um algoritmo baseado em colônia de formigas. No caso do problema da mochila multiobjetivo, a estratégia utilizada é aquela descrita em 5.1. Para o problema do roteamento multicast, a estratégia de construção da solução foi apresentada em 5.2.

5.3.2 Atualização das estruturas de subproblemas

O processo de atualizar as estruturas de subproblema a partir das novas soluções não dominadas encontradas (linha 8 do Algoritmo 5) é mostrado com mais detalhes no Algoritmo 7, que recebe como entrada o conjunto P_{ativo} e o conjunto de soluções S_{nd}

O Algoritmo 7 mostra a atualização das estruturas de subproblema no MACO/NDS. O processo iterativo (linha 1) passa por todas as estruturas em P_{ativo}. Nas linhas 2 a 5, a estrutura atual é chamada de a e seu arquivo é atualizado de acordo com as soluções no conjunto S_{nd}. Além disso, todas as soluções adicionadas ao arquivo são guardadas na lista A, enquanto todas as removidas são colocadas em R. As linhas 7, 8 e 9 são executadas apenas se o arquivo de a foi atualizado. Na linha 7, o valor *beta* de a é reinicializado para o valor original (parâmetro β do algoritmo); na linha 8, os valores de feromônio de a são incrementados de acordo com as novas soluções (A); e na linha 9, os valores de feromônios de a são decrementados de acordo com as soluções removidas

Algoritmo 7 Atualização dos subproblemas

```
1: para i ← 0 até |P_ativo| faça
2:     a ← P_ativo[i]
3:     dif = atualizarArquivo(a, S_nd)
4:     A ← dif.added
5:     R ← dif.removed
6:     se |A| > 0 então
7:         a.β ← β
8:         depositarFeromonios(a, A)
9:         evaporarFeromonios(a, R)
10:    senão
11:        a.convergencia ← a.convergencia + 1
12:        a.β ← a.β × β_inc
13:        se a.convergencia > MAX_convergencia então
14:            P_ativo[i].convergencia ← 0
15:            P_ativo[i] ← proxima_estrura_feromonios,
16:        fim se
17:    fim se
18: fim para
```

(R). As operações das linhas 8 e 9 são detalhadas na seção 5.3.3, sobre a atualização dos feromônios. Os valores de feromônios são alterados apenas se o arquivo relacionado ao subproblema corrente ($a = P_{ativo}[i]$) tiver sido modificado. Por outro lado, caso o arquivo não tenha sofrido atualização, as linhas 11 a 16 são executadas. Na linha 11, a convergência de a é incrementada em 1. Na linha 12, o valor β de a é modificado de acordo com a constante pré-definida β_{inc}. Como temos $\beta_{inc} > 1$, o valor de β atual correspondente ao subproblema corrente é aumentado. A ideia é que se diminua a importância das heurísticas sempre que a busca parecer estagnada. Como na implementação utilizada neste trabalho os valores das heurísticas são normalizadas entre 0 e 1, para diminuir a importância da heurística, deve-se aumentar o valor de β, por isso $\beta_{inc} > 1$. O valor de β_{inc} é um parâmetro que depende do problema. Nos nossos experimentos (descritos no capítulo 6), o valor de β aumenta em 10% tanto no PMM quanto no PRM, ou seja, $\beta_{inc} = 1,1$. Caso novas soluções sejam encontradas, o valor β associado ao subproblema corrente ($a.\beta$) volta ao valor inicial de β (linha 7). Por último, nas linhas 14 e 15, se a convergência de a atingiu o limite máximo $MAX_{convergencia}$, substitui-se a estrutura a em P_{ativo} pelo próximo $p_i \in P$, e reinicia-se o valor da convergência de a, de modo que possa ser reutilizado, caso necessário, no decorrer do algoritmo (isso acontece quando todos os subproblemas foram utilizados, mas a condição de parada não foi atingida). O valor da constante $MAX_{convergencia}$ utilizado neste trabalho foi 10.

5.3.3 Atualização dos feromônios

No MACO/NDS, a atualização dos valores de feromônio em um subproblema pode ser de dois tipos:

1. **Depósito:** a partir de uma nova solução construída, adiciona-se uma quantidade δ ao feromônio correspondente a cada parcela da solução. No caso de um problema em grafos, por exemplo, para cada aresta da solução, incrementa-se em δ o feromônio daquela aresta. O depósito de feromônio ocorre quando novas soluções não-dominadas são encontradas.

2. **Evaporação:** similar ao depósito, mas ao invés de incrementar o feromônio em δ, decrementa-se. Ocorre quando uma solução é removida do arquivo em decorrência de uma nova solução melhor.

A evaporação no MACO/NDS, diferentemente da maioria dos algoritmos baseados em ACO, não ocorre para todas as arestas (parcelas da solução) em todas as iterações. Ao invés disso, o feromônio só é decrementado quando uma solução passa a ser dominada (linha 9 do Algoritmo 7). Essa evaporação seletiva poderia causar convergência prematura se usada na maioria dos ACOs, pois grande parte dos feromônios atingiriam valores muito altos antes da condição de parada ser atingida. Isso não acontece no MACO/NDS devido à troca das matrizes de feromônios utilizadas para construir as soluções, quando suas respectivas convergências ultrapassam o limiar estabelecido (linha 13 do Algoritmo 7). Dessa forma, as matrizes de feromônios correspondentes a cada subproblema não ficam ativas tempo suficiente para se tornaram um problema no que se refere à convergência do algoritmo.

Dada uma solução s do problema do roteamento multicast (PRM), se s deve ser reforçada (depósito), cada aresta e da árvore s incrementa o valor correspondente na matriz de feromônios em um fator δ. Matematicamente, o valor de δ é definido por:

$$\delta(e) = (1 - pesos(e)) \times \rho \tag{22}$$

Sendo, $pesos(e)$ a média dos pesos normalizados na aresta e e ρ a taxa de evaporação (parâmetro do MACO/NDS). Por outro lado, se s se tornou uma solução dominada (saiu do arquivo) e, por isso, deve ser desencorajada (evaporação), os valores na matriz de feromônios correspondentes às arestas de s devem ser decrementados em δ.

Dada uma solução s do problema da mochila multiobjetivo (PMM), se s deve ser reforçada (depósito), cada item i de s incrementa o valor correspondente no vetor de feromônios em um fator δ. O cálculo de δ no PMM é dado por:

$$\delta(i) = lucros(i) \times (1 - peso(i)/peso\_max) \times \rho \tag{23}$$

Onde, $lucros(i)$ é a média dos valores normalizados de lucro do item i, e $peso\_max$ é o maior peso dentre todos os itens. Se s deve ser desencorajada (evaporação), os valores no vetor de feromônios correspondentes aos itens de s devem ser decrementados em δ.

Capítulo 6

Experimentos

Neste capítulo são apresentados os experimentos realizados durante a pesquisa. Esses experimentos visam avaliar o desempenho de diversas abordagens bio-inspiradas (algoritmos evolutivos e baseados em colônias de formigas), incluindo o algoritmo proposto nesta dissertação (MACO/NDS), quando empregadas na resolução de dois problemas discretos com muitos objetivos.

Para facilitar a análise dos resultados, esses experimentos foram divididos em 4 etapas distintas de acordo com os objetivos desejados. Na primeira etapa, cinco algoritmos evolutivos multiobjetivos (AEMOs), encontrados da literatura, foram avaliados em problemas discretos de 2 a 6 objetivos. O objetivo dessa etapa é analisar o comportamento desses algoritmos de acordo com o número de objetivos tratados. Na segunda etapa, a fim de se desenvolver a base para a elaboração do algoritmo MACO/NDS pra o PRM multiobjetivo, avaliaram-se diversas estratégias para a construção de uma solução em um ACO mono-objetivo. A terceira etapa de experimentos contou com o algoritmo proposto (MACO/NDS) em sua versão final, ou seja, com a melhores estratégias observadas na segunda etapa de experimentos. O objetivo da terceira etapa foi avaliar o MACO/NDS contra os algoritmos *many-objective* vistos até então (NSGA-III, MOEA/D, AEMMT e AEMMD) nos dois problemas (PMM e PRM) em instâncias com formulações a partir de 4 objetivos. A última etapa de experimentos conta com a adição de mais três métodos em relação a etapa anterior: o SPEA2-SDE, o MOEA/D-ACO e o MOACS. O objetivo dessa última seção é avaliar o comportamento dos algoritmos em instâncias mais complexas dos problemas da mochila e do roteamento. Além disso, na quarta etapa, é utilizada a métrica hiper-volume na avaliação dos resultados.

6.1 Ambiente de teste

Esta seção descreve a metodologia empregada nos experimentos, isto é, os algoritmos utilizados, as formas de avaliação e a apresentação dos resultados. Os seguintes algoritmos multiobjetivos foram utilizados em nossos experimentos:

- *Non-Dominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002a);
- *Non-Dominated Sorting Genetic Algorithm III* (NSGA-III) (DEB; JAIN, 2014);
- *Strength Pareto Evolutionary Algorithm 2* (SPEA2) (ZITZLER; LAUMANNS; THIELE, 2001);
- *SPEA2 with Shift-Based Density Estimation* (SPEA2-SDE) (LI; YANG; LIU, 2014);
- *Multiobjective Evolutionary Algorithm Based on Decomposition* (MOEA/D) (ZHANG; LI, 2007);
- *Multi-Objective Evolutionary Algorithm with Many Tables* (MEAMT). Nesta dissertação, referenciamos esse trabalho pelo nome proposto originalmente em português, Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas (AEMMT) (BRASIL; DELBEM; SILVA, 2013);
- *Multi-Objective Evolutionary Algorithm based on Non-dominated Decomposed Sets* (MEANDS). Nesta dissertação, referenciamos esse trabalho pelo nome proposto originalmente em português, Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias (AEMMD) (LAFETÁ et al., 2018);
- *Multiobjective Evolutionary Algorithm Based on Decomposition ACO* (MOEA/D-ACO) (KE; ZHANG; BATTITI, 2013);
- *Multi-Objective Ant Colony Optimization Algorithm* (MOACS) (RIVEROS et al., 2016)
- e o ACO proposto: *Many-objective Ant Colony Optimization based on Non-dominated Decomposed Sets* (MACO/NDS).

Todos esses algoritmos foram avaliados considerando dois problemas discretos multiobjetivos: o Problema da Mochila Multiobjetivo (PMM) e o Problema Problema do Roteamento Multicast (PRM). A fim de verificar o comportamento dos algoritmos em relação ao número de objetivos, diversas formulações foram consideradas, avaliando-se problemas desde dois até seis objetivos. Assim como o número de funções objetivo, a topologia da rede (PRM) e a quantidade de itens (PMM) também afetam a complexidade. Portanto, elaboraram-se instâncias diferentes para cada problema. No PRM, são consideradas seis redes de diferentes complexidades, as quais também foram utilizadas em (LAFETÁ, 2016). No PMM variou-se a quantidade de itens em 30, 40, 50, 100 e 200.

Parte das métricas utilizadas para avaliar os algoritmos são paramétricas, ou seja, dependem do conhecimento prévio sobre a fronteira de Pareto do problema. Como foi inviável a obtenção do conjunto Pareto ótimo exato para cada instância, antes de executar os experimentos, foi calculado um conjunto aproximado para cada um dos cenários de

teste. Tal conjunto aproximado, denominado "Pareto aproximado" ou "fronteira de Pareto aproximada", foi obtido através de várias execuções de todos os algoritmos utilizados nos experimentos sobre o cenário. Além disso, no caso das instâncias de redes do PRM, esses resultados foram comparados com os Paretos aproximados obtidos em (LAFETÁ et al., 2018).

Considerando as quatro etapas de experimentos, foram usadas 4 métricas multiobjetivo para se avaliar o desempenho dos algoritmos, além do tempo de execução. São elas:

- **Taxa de Erro (ER)**: sendo S o conjunto de soluções encontrado pelo algoritmo e P a fronteira de Pareto aproximada. A taxa de erro é a porcentagem das soluções de S que não aparecem em P, como mostra a fórmula a seguir:

$$ER = \frac{\sum_{s \in S} e(s)}{|S|}, \quad e(s) = \begin{cases} 0 & s \in P \\ 1 & s \notin P \end{cases} \tag{24}$$

  Por exemplo, tome $S = \{A, B, C, D, F\}$ e $P = \{A, D\}$. Como 3 das soluções de S não estão em P, a taxa de erro é $3/5 = 0,6$, ou seja, 60%.

- ***Generational Distance* (GD)**: mede a distância das soluções no conjunto encontrado pelo algoritmo (S) para as soluções mais próximas na fronteira de Pareto aproximada (P). A distância euclidiana ($dist$) de cada $s \in S$ é medida em relação a cada $p \in P$. A menor distância para p de cada s é então elevada ao quadrado e adicionada à soma. Ao final, retorna-se o valor da raiz quadrada do somatório dividido pela quantidade de elementos em S. O cálculo da GD é dada por:

$$GD = \frac{\sqrt{\sum_{s \in S} \min_{p \in P} dist(s,p)^2}}{|S|} \tag{25}$$

  Por exemplo, considere um problema bi-dimensional, onde o S encontrado é $\{(1,2),(5,2),(3,7)\}$ e o Pareto aproximado P vale $\{(2,3),(5,2)\}$. O primeiro passo é encontrar as menores distâncias de cada elemento de S para P. Tomando o primeiro elemento de S, $s_1 = (1,2)$, calcula-se as distâncias para p_1 e p_2. $dist(s_1,p_1) = \sqrt{(1-2)^2 + (2-3)^2} = \sqrt{2}$ e $dist(s_1,p_2) = \sqrt{(1-5)^2 + (2-2)^2} = 4$. Portanto, a menor distância de s_1 para P é $\sqrt{2}$. O mesmo cálculo é feito para s_2 e s_3, onde obtém-se que a menor distância de s_2 para P é 0 e a menor distância de s_3 para P é $\sqrt{17}$. Logo, a distância GD é dada por $\frac{\sqrt{\sqrt{2}+0+\sqrt{17}}}{3} = 0,78$.

- ***Generational Distance (Excluding Pareto)* (GDp)**: similar à definição anterior, mas ao invés de considerar todas as soluções no cálculo de distância, considera apenas aquelas que não fazem parte do Pareto aproximado, ou seja, elimina do cálculo as distâncias que valem 0. No exemplo dado para o GD, o valor de GDp seria dado por $\frac{\sqrt{\sqrt{2}+\sqrt{17}}}{2} = 1,18$.

- ***Pareto Subset* (PS)**: número absoluto de soluções encontradas (S) que fazem parte do Pareto aproximado (P), o qual é obtido por:

$$PS = \sum_{s \in S} a(s), \quad a(s) = \begin{cases} 1 & s \in P \\ 0 & s \notin P \end{cases} \tag{26}$$

  Por exemplo, dado $S = \{A, B, C, D, F\}$ e $P = \{A, D\}$. Como 2 das soluções de S estão em P, o valor do PS é 2.

- **Hiper-Volume (HV)**: métrica multiobjetivo que independe do conhecimento prévio da fronteira de Pareto. Mede o volume da figura geométrica m-dimensional (sendo m o número de objetivos) formada pelas soluções encontradas considerando a origem do sistema de coordenadas como um ponto arbitrário p_{ref}, chamado de ponto de referência. As coordenadas de p_{ref} são diferentes para cada cenário (problema, formulação de objetivos e instância) e são determinadas pelo pior valor encontrado em cada um dos objetivos considerando a união das soluções dos Paretos obtidos em cada execução. Se o PMM de 5 objetivos e 100 itens é executado 10 vezes, por exemplo, extraem-se os 10 resultados e colocam-se as soluções em um único conjunto S_{todos}. Varre-se S_{todos} procurando pelo pior valor em cada uma das 5 coordenadas e cria-se uma solução fictícia para os valores encontrados. Essa solução fictícia é então utilizada como ponto de referência (p_{ref}) para o PMM de 5 objetivos e 100 itens. Em nossos experimentos, foi utilizada a implementação do cálculo de hiper-volume descrito em While, Bradstreet e Barone (2012)[1]. Por exemplo, considere um problema de minimização bi-dimensional e tome $S = \{(6, 3), (9, 2), (4, 5), (5, 4), (3, 6)\}$ como o conjunto de soluções encontradas. O ponto de referência será dado pelo ponto formado pelo pior valor obtido na primeira coordenada e pelo pior valor obtido na segunda, ou seja, $p_{ref} = (9, 6)$. A Figura 21 mostra a representação gráfica de S e p_{ref}. Neste exemplo, o valor do hiper-volume é dado pelo volume da forma geométrica destacada em cinza (F), ou seja, $hv = 13$. Dessa forma, é fácil visualizar que quanto mais representativo e distribuído o conjunto de soluções, maior o volume da forma geométrica, ou seja, o hiper-volume deve ser maximizado.

- Tempo: tempo médio, em segundos, necessário para a execução do algoritmo. Foi aferido a partir das diferenças de tempo entre o momento em que o algoritmo inicia a execução até o momento quando retorna o conjunto de soluções encontradas. O valor final do tempo é obtido através da média entre três execuções.

Para obter os valores dessas métricas, nas seções 6.2 e 6.4, foram feitas 100 execuções de cada algoritmo em cada cenário. Na Seção 6.5, que utiliza o hiper-volume, o número

[1] O código da implementação para o hiper-volume utilizada pode ser encontrado em <http://www.wfg.csse.uwa.edu.au/hypervolume/code/WFG_1.15.tar.gz>

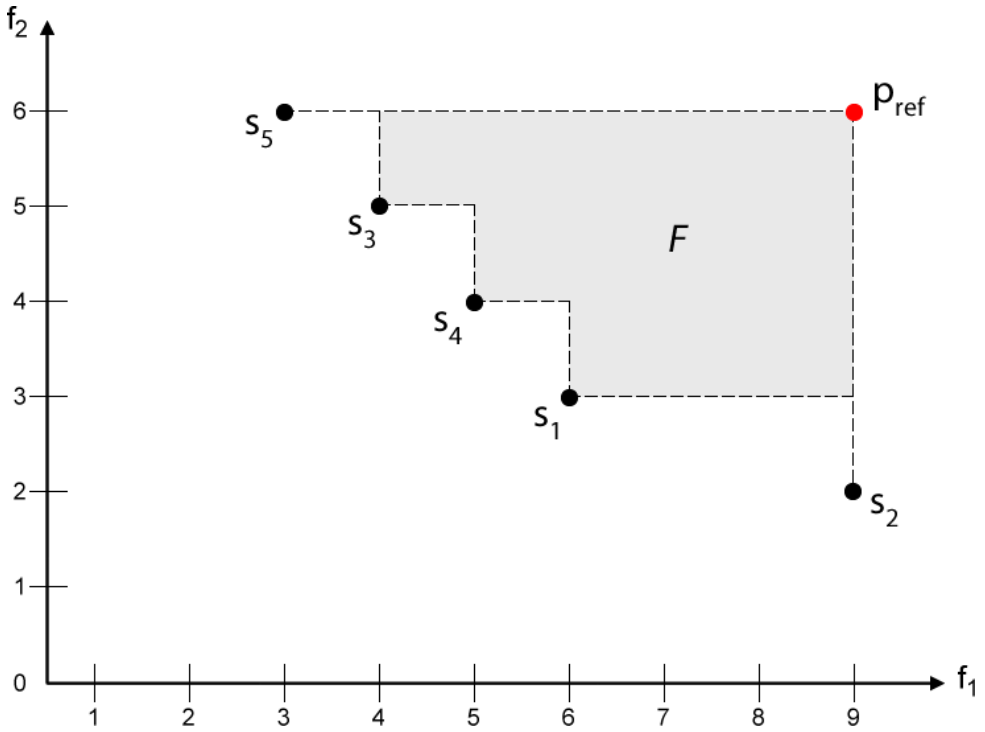

Figura 21 – Exemplo de hiper-volume

de execuções foi 30. O número reduzido de execuções na última etapa se deve à maior dificuldade em se calcular o hiper-volume em relação às demais métricas. Uma ressalva também é feita quanto ao tempo de execução, que foi calculado a partir da média aritmética de três execuções isoladas de cada algoritmo em cada cenário. O número reduzido de experimentos para se obter os valores de tempo se deve a dois fatores. Primeiramente, executar cada algoritmo isoladamente em uma máquina leva um tempo considerável, o que torna inviável a execução de vários experimentos. Além disso, o tempo não varia muito de uma execução para outra, fazendo com que a média entre três simulações seja suficiente.

A apresentação dos resultados é feita de três formas: gráficos, tabelas e testes de hipótese. Os gráficos utilizam barras para mostrar os valores da média dos resultados de cada métrica considerando todas as execuções para o algoritmo e cenário indicados. Em alguns gráficos, os desvios padrões são representados por linhas verticais sobre as barras. Os mesmos valores mostrados de maneira gráfica também são disponibilizados na forma de tabelas no apêndice A. Além disso, em alguns dos experimentos, é utilizado o testes de hipótese teste Z (*Z test*) para verificar se a superioridade de algum algoritmo em relação aos demais possui significância estatística. O teste de hipótese é apresentado através de uma tabela, onde cada linha representa um cenário e cada coluna uma métrica de desempenho, os sinais de “>” e “<” indicam se o método analisado obteve resultado significativamente maior ou menor em relação a seu adversário, enquanto o sinal “=” indica que não é possível determinar de forma significativa a superioridade entre eles.

Além dos sinais, no teste de hipótese, as células são coloridas em verde, caso o algoritmo analisado tenha obtido melhor desempenho, em vermelho, caso o desempenho tenha sido pior e em branco, nos casos onde o desempenho foi similar.

6.2 Análise Comparativa dos Algoritmos Evolutivos

Os experimentos apresentados nesta seção serviram como base para a elaboração de um artigo publicado no *Brazilian Conference on Intelligent Systems* (BRACIS) (FRANÇA et al., 2017). Na elaboração do artigo, o autor da presente dissertação implementou e efetuou os experimentos com o PMM, enquanto os experimentos com o PRM foram realizados por outro autor e também estudante de pós-graduação Thiago Fialho Lafetá, quem investigou o PRM em sua própria dissertação (LAFETÁ, 2016). Posteriormente, o PRM também foi implementado e o novo ambiente serviu de base para os experimentos descritos nas próximas seções. Os resultados no PRM gerados pelo Thiago Fialho são apresentados nessa seção, pois suas conclusões foram importantes na definição dos próximos passos da dissertação.

Nessa primeira etapa, foram realizados experimentos visando a análise comparativa dos AGs multiobjetivos NSGA-II, NSGA-III, SPEA2, MOEA/D e AEMMT nos problemas da mochila e do roteamento multicast. O número de objetivos varia entre dois e seis e, no PRM, três redes são testadas, enquanto o PMM lida com instâncias de 30, 50 e 100 itens. Além dos cinco algoritmos mencionados, avalia-se também uma pequena modificação no AEMMT chamada de AEMMT-F que remove o limite no tamanho do arquivo de soluções não-dominadas (*free size*). As avaliações dos algoritmos são feitas com base em métricas relacionadas ao Pareto aproximado calculado e permitem um melhor entendimento sobre o comportamento dos algoritmos. Isso facilitou a identificação de pontos fortes e fracos de cada estratégia, ajudando na elaboração do MACO/NDS, que é o novo algoritmo proposto neste trabalho.

Durante os experimentos, os algoritmos NSGA-II, NSGA-III, SPEA2, MOEA/D e AEMMT foram avaliados em diferentes cenários dos dois problemas investigados (PRM e PMM). Ao todo, 30 cenários de teste foram considerados (15 para cada problema):

- PMM: 5 formulações de objetivos (2 a 6) e 3 instâncias (30, 50 e 100 itens). Os valores de cada objetivo e dos pesos dos itens são gerados aleatoriamente para cada instância do problema, conforme descrito na Seção 4.1.
- PRM: 5 formulações de objetivos (P_2 a P_6) e 3 redes (R_1, R_2 e R_3). Tanto as formulações quanto as redes estão descritas na Seção 4.2.

Para cada um dos cenários foi obtida um conjunto de Pareto aproximado, através de múltiplas execuções dos 5 algoritmos testados. A Tabela 2 mostra a cardinalidade de cada Pareto encontrado.

Tabela 2 – Conjunto de Pareto considerado para os cenários investigados na primeira etapa de experimentos

	PRM			**PMM**		
Objetivos	R1	R2	R3	30 itens	50 itens	100 itens
2	14	9	6	15	67	170
3	30	18	17	106	501	6288
4	122	72	60	425	986	88374*
5	424	326	551	1765	5213	176868*
6	1196	657	1078	5800	35760*	248198*

Na Tabela 2, a quantidade de elementos nos paretos do PMM é demasiadamente grande para as formulações *many-objective* com 100 itens. Isso acontece porque o espaço de busca do problema da mochila cresce exponencialmente com o número de itens. Esse crescimento é potencializado com o número de objetivos considerado, ou seja, quanto maior a quantidade de funções objetivos na formulação, maior o número de soluções que serão não-dominadas. O asterisco ao lado de alguns valores no PMM significa que não foi possível estabilizar o Pareto, ou seja, a cada rodada de execuções dos algoritmos, novas soluções eram encontradas. Tal comportamento motivou a realização dos experimentos da etapa 4 (Seção 6.5), nos quais não se usa um Pareto pré-definido para se avaliar os algoritmos. Apesar disso, os resultados para o problema de 100 itens ainda são relevantes, pois mesmo que os Paretos não sejam estáveis, comparam-se as execuções dos algoritmos contra o conjunto com todas as soluções encontradas, possibilitando determinar o algoritmo com melhor potencial para encontrar boas soluções nos diferentes cenários dos problemas multiobjetivos.

Os resultados utilizados no cálculo das métricas de desempenho ER, GD e PS foram obtidos a partir da média entre 100 execuções de cada algoritmo. Nessas execuções foram adotados os parâmetros de configuração listados na Tabela 3. Os parâmetros foram retirados de trabalhos anteriores (LAFETÁ, 2016; BRASIL; DELBEM; SILVA, 2013; ISHIBUCHI; AKEDO; NOJIMA, 2015). Na tabela, o asterisco em "número de gerações" indica que nem todos os algoritmos seguem esse parâmetro. O AEMMT executa 9 mil gerações para o PRM e 7500 para o PMM. Isso acontece, pois esse algoritmo gera apenas 1 filho por ciclo no PRM e apenas 2 no PMM necessitando, portanto, de mais gerações para fazer o mesmo número de avaliações de novos indivíduos. No problema da mochila com 100 itens, devido à complexidade do problema, dobrou-se a quantidade de gerações. Ainda na tabela 3, o termo "variável" indica que a taxa de mutação adotada varia de acordo com a instância do problema. Nos experimentos realizados, foi empregada uma taxa de mutação de 6% para o PMM de 30 itens, 4% para o de 50 itens e 2% para o de 100 itens, similar aos valores utilizados em (ISHIBUCHI; AKEDO; NOJIMA, 2015). Esses valores foram empregados em todos os algoritmos.

As Figuras 22, 23 e 24 mostram, respectivamente, os resultados para o PMM de 30, 50

Tabela 3 – Parâmetros utilizados pelos AEMOs na 1ª etapa, de acordo com o problema tratado.

Parâmetro	**(A) PRM**	**(B) PMM**
Tamanho da população	90	150
Número de gerações*	100 (9000*)	100 (7500*)
Taxa de crossover	100%	100%
Taxa de mutação	20%	variável
Tamanho da vizinhança (MOEA/D)	10	10
Tamanho das tabelas (AEMMT)	30	50
Tamanho da tabela de dominância (AEMMT)	90	150
Número de subdivisões (NSGA-III)	8	8

e 100 itens. Nas Figuras 26, 27 e 28 são apresentados os resultados dos algoritmos para o PRM considerando as redes R_1, R_2 e R_3, respectivamente. Uma análise consolidada, considerando a média entre as três instâncias de cada problema, é apresenta nas Figuras 25 (PMM) e 29 (PRM).

Como pode ser observado na Figura 22, o PMM com 30 itens é um problema relativamente simples e, no geral, os métodos de otimização multiobjetivo alcançam bons resultados. Até 4 objetivos, apenas o MOEA/D passou a marca de 20% de erro. Para 2 e 3 objetivos, os algoritmos NSGA-II, SPEA2 e AEMMT apresentaram desempenhos muito similares, resultando nos melhores percentuais de erro entre todos os algoritmos avaliados. A partir de 4 objetivos é possível notar uma queda considerável no desempenho dos algoritmos multiobjetivos clássicos (NSGA-II e SPEA-2). Considerando-se todas as formulações de objetivo no PMM, os melhores resultados foram obtidos pelo AEMMT que manteve o erro abaixo de 6% em todos os casos. Analisando a métrica GD, obtêm-se as mesmas conclusões tiradas a partir do ER. Isto é, considerando todas as formulações, o AEMMT gera os melhores resultados. Por outro lado, apesar de terem um desempenho muito bom nos problemas de 2 e 3 objetivos, os valores da métrica GD para o NSGA-II e o SPEA2 apresentam resultados ruins a partir de 4 objetivos. O tamanho da fronteira de Pareto encontrada, medida pelo PS, é maior nos algoritmos MOEA/D e AEMMT. Esse comportamento é esperado no MOEA/D, pois diferentemente dos demais AEMOs, ele não aplica uma limitação sobre o tamanho do arquivo que armazena as soluções não-dominadas. O alto valor de PS também é esperado no AEMMT, pois, apesar da tabela de não-dominância ser limitada, o resultado final é dado pelas soluções não dominadas considerando todas as tabelas, ou seja, o limite sobre o tamanho do Pareto é mais fraco que os demais algoritmos. Considerando-se os valores das três métricas em todas as formulações de objetivos para o PMM de 30 itens, está claro que, no geral, o AEMMT é o melhor dentre os métodos testados.

O PMM com 50 itens (Figura 23) é um problema mais complexo que sua versão com

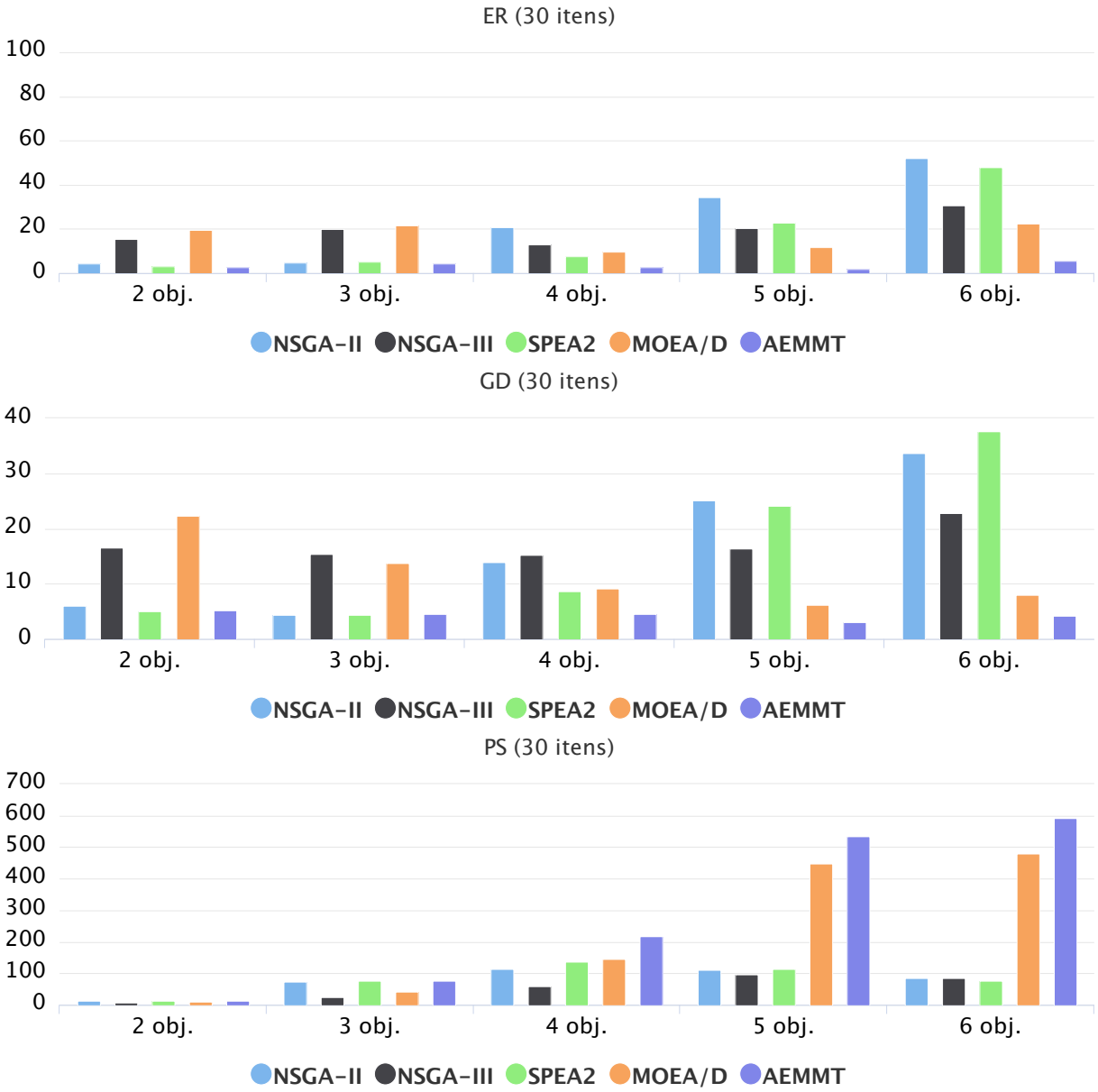

Figura 22 – Desempenho dos algoritmos na 1ª etapa para o PMM com 30 itens

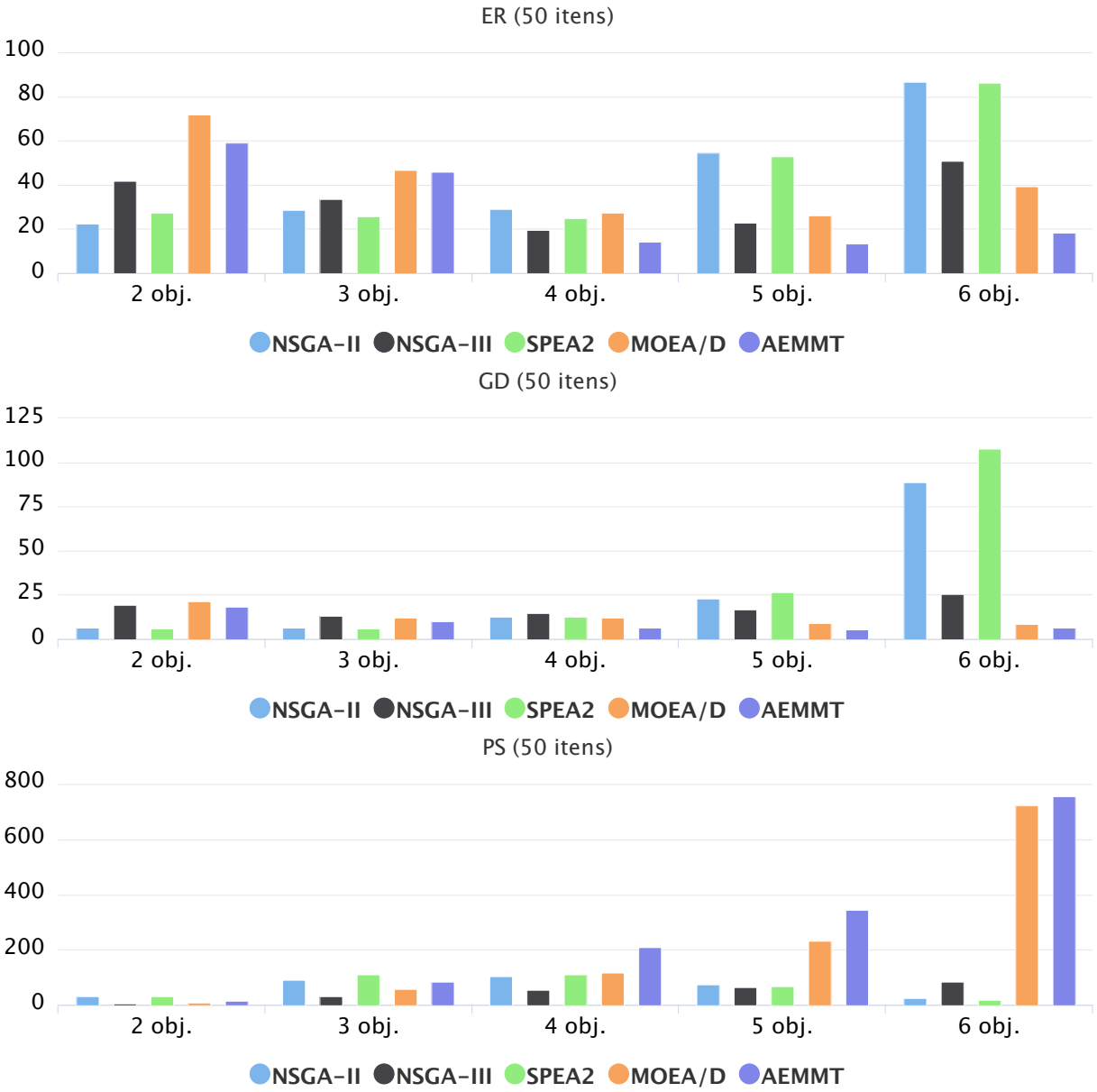

Figura 23 – Desempenho dos algoritmos na 1ª etapa para o PMM com 50 itens

30 itens e os algoritmos têm mais dificuldade para convergir para o Pareto aproximado, o que pode ser comprovado pelas taxas de erro maiores em relação ao experimento anterior. Nessa instância, é mais evidente o comportamento de cada algoritmo em função das diferentes formulações de objetivos. Nas formulações com 2 e 3 objetivos, os algoritmos clássicos (NSGA-II e SPEA2) são imbatíveis, independentemente da métrica analisada. A partir de 4 objetivos, o AEMMT assume a primeira posição e é o melhor algoritmo tanto para ER quanto para GD e PS. Depois do AEMMT, para 4 objetivos, a menor taxa de erro é atingida pelo NSGA-III, seguido pelos SPEA2, MOEA/D e NSGA-II. Para 5 objetivos, o NSGA-III continua apresentando a segunda menor taxa de erro, enquanto o MOEA/D obtém resultado melhor que o SPEA2. Para seis objetivos, o MOEA/D tem ER pior apenas que o AEMMT e o NSGA-III apresenta o terceiro melhor resultado. Com relação ao GD, o MOEA/D obtém o segundo melhor desempenho para as formulações *many-objective* (a partir de 4 objetivos) e o NSGA-III obtém o terceiro melhor, excluindo a formulação com 4 objetivos, onde o SPEA2 é melhor. O PS, a partir de 4 objetivos, é dominado pelos algoritmos AEMMT e MOEA/D, o NSGA-III possui um comportamento mais parecido com os algoritmos clássicos que com os algoritmos *many-objective*. É possível notar que, com o aumento do número de objetivos, o NSGA-II e o SPEA2 pioram enquanto o AEMMT mantém o ER e o GD estáveis e melhora o PS. O MOEA/D é o segundo melhor algoritmo para problemas *many-objectives* e o NSGA-III é o terceiro. Para o PMM de 50 itens, o AEMMT é a melhor opção para muitos objetivos, assim como observado no PMM com 30 itens.

O tamanho do espaço de busca no problema da mochila é 2^n, sendo n o número de itens. Portanto, a complexidade do PMM com 100 itens, cujos resultados são apresentados na Figura 24, é muito maior que nas instâncias anteriores. Nesse caso, não foi possível encontrar um Pareto estável para as formulações de 4, 5 e 6 objetivos. Dessa forma, apesar das métricas ER, GD e PS serem um bom indicativo de desempenho entre os algoritmos, a melhor forma de avaliação seria o hiper-volume. Um experimento similar, mas utilizando o hiper-volume, é apresentado na Seção 6.5. É difícil avaliar o problema considerando 100 itens, pois, a partir dos resultados apresentados para a métrica de erro (ER) na Figura 24, é possível perceber que poucos algoritmos conseguiram encontrar boas soluções. Considerando a métrica ER, o algoritmo com melhor desempenho retornou uma taxa de erro acima de 50%. O NSGA-III apresentou o menor ER nos cenários de 2, 3, 4 e 5 objetivos, enquanto que na formulação de 6, os melhores desempenhos nessa métrica foram obtidos pelos AEMMT e MOEA/D. Com relação ao GD, o NSGA-II e o SPEA2 apresentaram os melhores resultados para 2 objetivos. Na formulação de 3 objetivos em diante, o MOEA/D foi o algoritmo com menor GD, sendo que , a partir de 4 objetivos, o AEMMT obteve desempenho similar. A situação se repete ao analisar o PS. A partir de 3 objetivos, o MOEA/D traz um conjunto maior de soluções na fronteira de Pareto, seguido pelo AEMMT a partir de 4 objetivos. Em problemas com muitos objetivos, o AEMMT

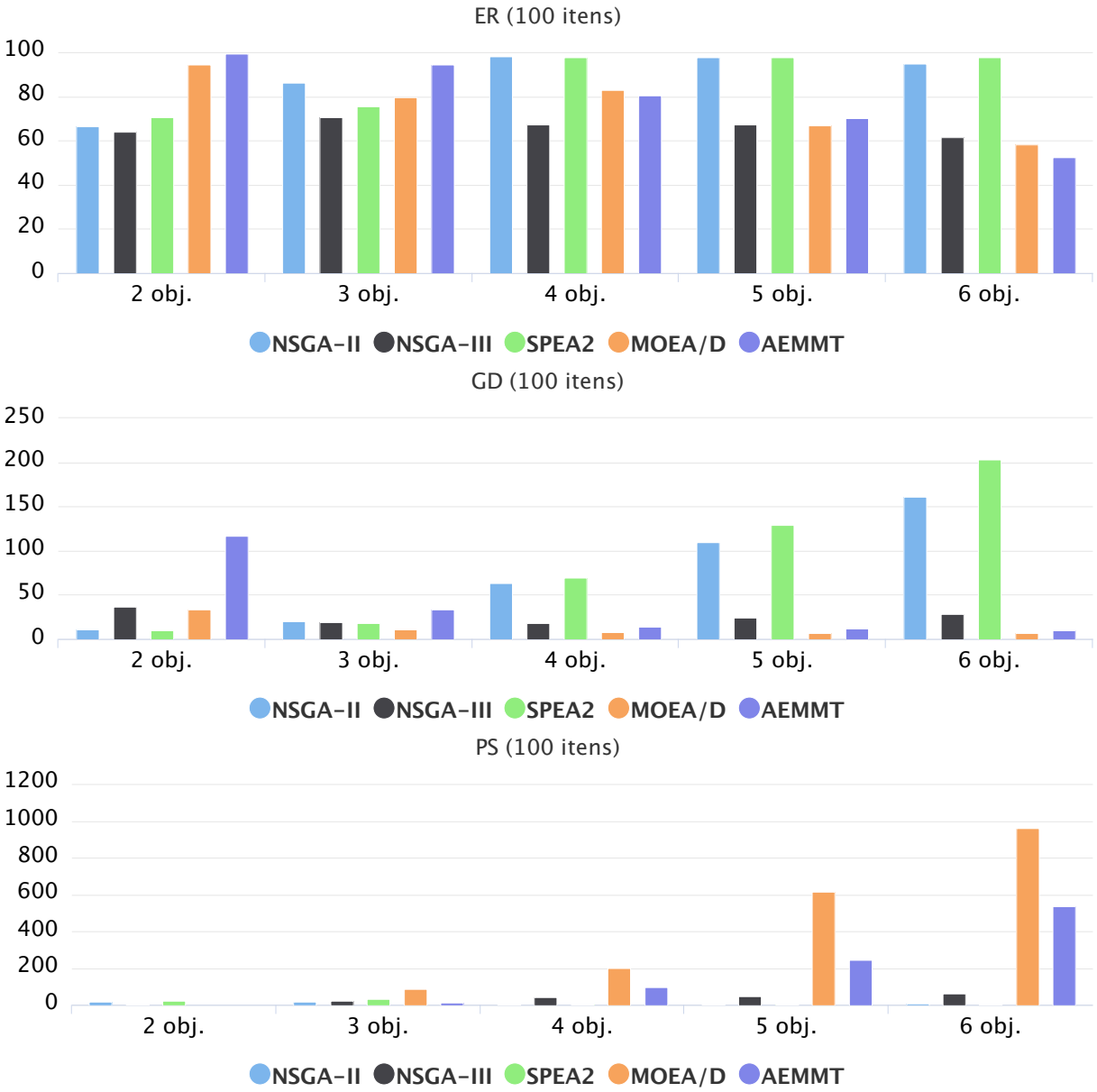

Figura 24 – Desempenho dos algoritmos na 1ª etapa para o PMM com 100 itens

e o MOEA/D são as melhores opções, sendo que o AEMMT apresenta uma taxa de erro levemente menor e o MOEA/D obtém um valor de PS consideravelmente melhor. Para problemas com 4 objetivos ou menos, o NSGA-III é o método que consegue as soluções mais próximas do Pareto. Entretanto, o conjunto de soluções gerado pelo NSGA-III não é tão grande quanto aquele gerado pelo MOEA/D. Conforme ressaltado anteriormente, o NSGA-III possui um limite no número de soluções não-dominadas possíveis de serem encontradas, diferentemente do AEMMT e do MOEA/D, que é o próprio tamanho da população. Entretanto, é possível observar que o tamanho de PS obtido pelo NSGA-III é abaixo desse limite (150).

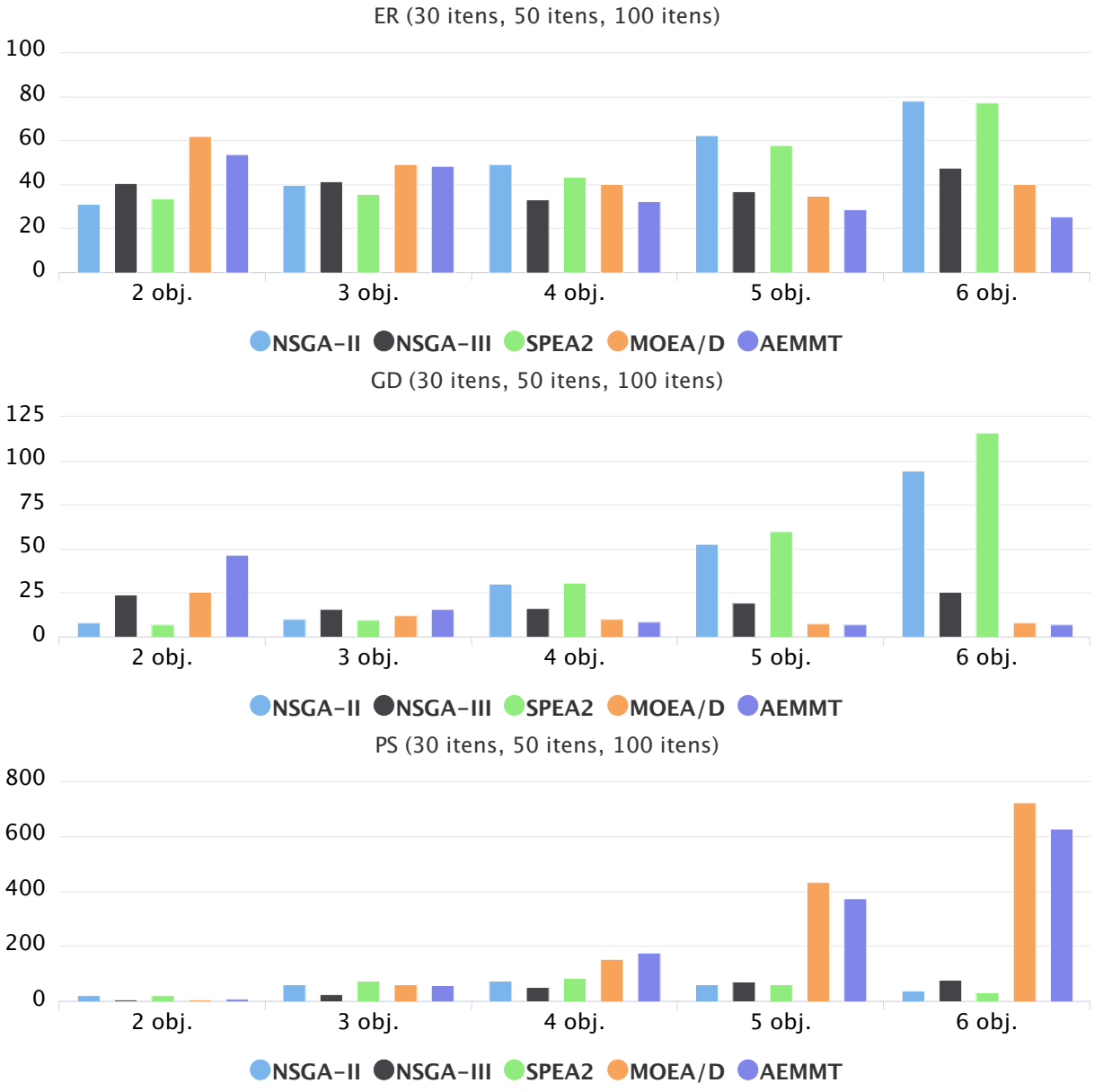

Figura 25 – Resultado consolidado da 1ª etapa considerando o PMM com 30, 50 e 100 itens

A Figura 25 apresenta os resultados do PMM de 30, 50 e 100 itens de forma consolidada para que se possa analisar, de forma geral, o comportamento dos algoritmos nas diferentes formulações de objetivo. Os gráficos representam as médias (simples) entre os três cenários de dificuldade (30, 50 e 100 itens). Como esperado, o NSGA-II e o SPEA2 são os melhores algoritmos para as formulações de 2 e 3 objetivos, tendo resultado nos melhores valores para as métricas ER, GD e PS. Por outro lado, a partir de 4 objetivos, o desempenho de ambos os algoritmos cai consideravelmente, enquanto o AEMMT passa a apresentar os melhores resultados. O NSGA-III, no problema de 4 objetivos, apresenta um erro quase tão baixo quanto o AEMMT, mas seu GD e PS são piores. O MOEA/D, para 5 e 6 objetivos, apresenta o segundo melhor desempenho em qualquer uma das métricas. Em resumo, o NSGA-III não parece uma boa opção em nenhum dos casos, pois sempre há outro algoritmo que o supera. O NSGA-II e o SPEA2 são igualmente bons e os melhores em problemas com poucos objetivos. O AEMMT e o MOEA/D são ótimas opções para problemas a partir de 4 objetivos, sendo que o MOEA/D confere um melhor PS enquanto o AEMMT proporciona uma menor taxa de erro.

Na Figura 26 são apresentados os desempenhos dos AEMOs na resolução do PRM para a rede 1. Essa é a instância mais simples testada e retornou baixas taxas de erro para todos os algoritmos. Considerando todas as formulações de objetivo, o maior valor de ER foi obtido pelo NSGA-II aplicado ao problema de 6 objetivos (36,3%). Diferentemente do esperado, o NSGA-III mostrou o melhor resultado (ER, GD e PS) para os problemas com 2, 3 e 4 objetivos. A partir de 5 objetivos, o AEMMT e o MOEA/D são os dois melhores métodos, sendo que o primeiro apresenta uma menor taxa de erro, enquanto o segundo obtém uma fronteira de soluções não-dominadas de maior cardinalidade. Para poucos objetivos, o NSGA-III é claramente o melhor método, para 5 ou mais critérios de otimização, ambos AEMMT e MOEA/D são boas opções.

Considerando o desempenho dos AEMOs para o PRM com a rede 2 (Figura 27), o NSGA-III é o melhor algoritmo nos problemas com 2, 3 e 4 objetivos, em qualquer uma das métricas, seguido pelo NSGA-II e SPEA2. Para 5 objetivos, o AEMMT apresenta a menor taxa de erro (16,6%), seguido de perto pelo NSGA-III (16,9%). Com relação ao GD no problema de 5 objetivos, o AEMMT obtém o melhor desempenho, mas o NSGA-III e o SPEA2 atingem valores similares. Em termos de PS, o MOEA/D, o AEMMT e o NSGA-III conseguem valores bem similares, sendo o MOEA/D o melhor dentre eles. Para 6 objetivos, os menores ER e GD foram obtidos pelo AEMMT (3,6%; 1,7), em primeiro lugar, pelo MOEA/D (16,3%; 3,4), em segundo e pelo NSGA-III (17,4%; 4,5), em terceiro. Os melhores valores de PS foram apresentados pelo MOEA/D (155,4), que ficou bem a frente dos segundo e terceiro melhores algoritmos (AEMMT com 84,8 e NSGA-III com 79,9). Nos cenários com poucos objetivos, o NSGA-III mostrou ser a melhor opção para a rede 2, enquanto nos com 5 ou mais objetivos, é preferível utilizar o AEMMT, caso uma baixa taxa de erro seja desejável, ou o MOEA/D, caso deseja-se um maior conjunto de

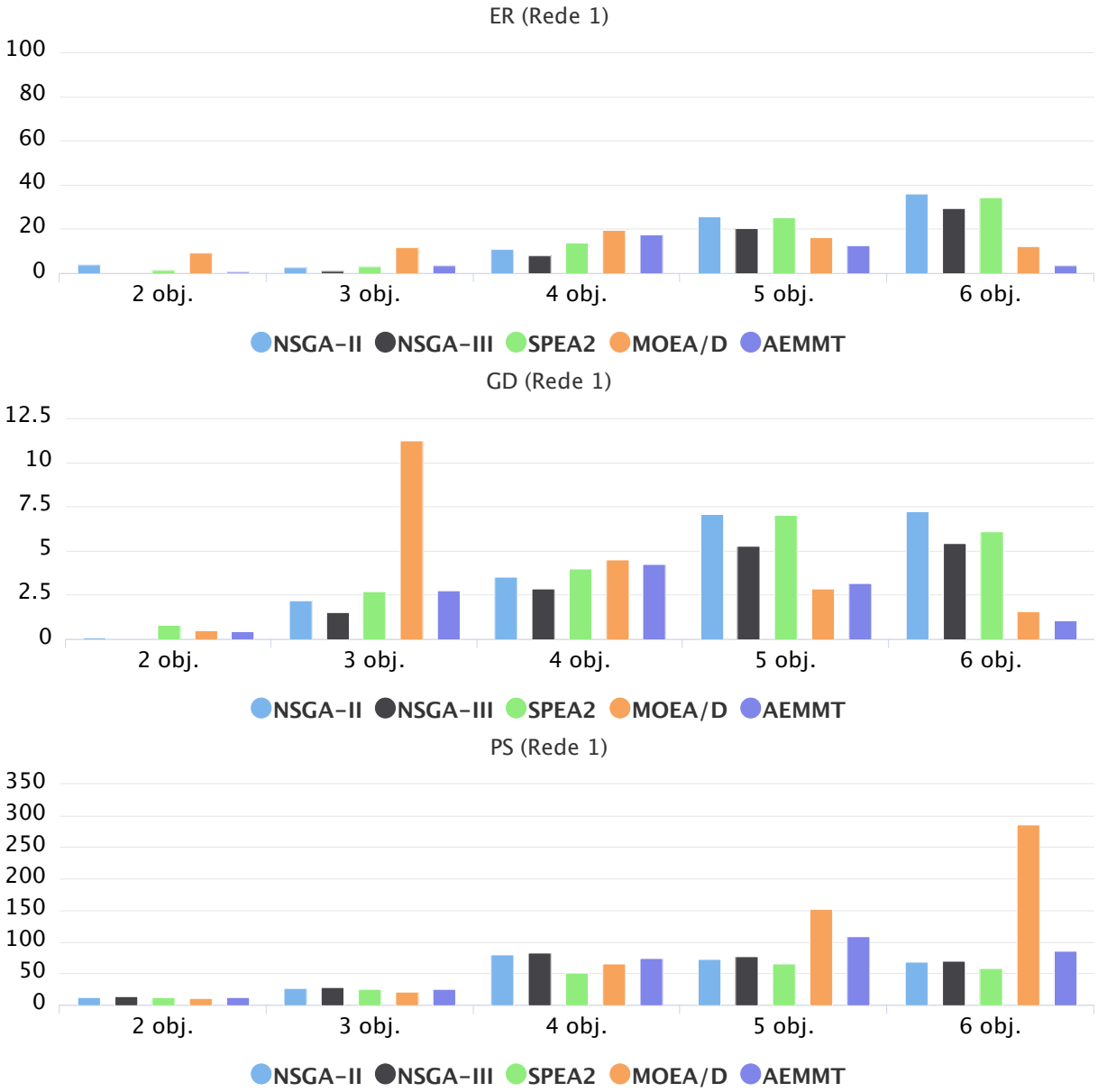

Figura 26 – Desempenho dos algoritmos na 1ª etapa para o PRM na rede 1

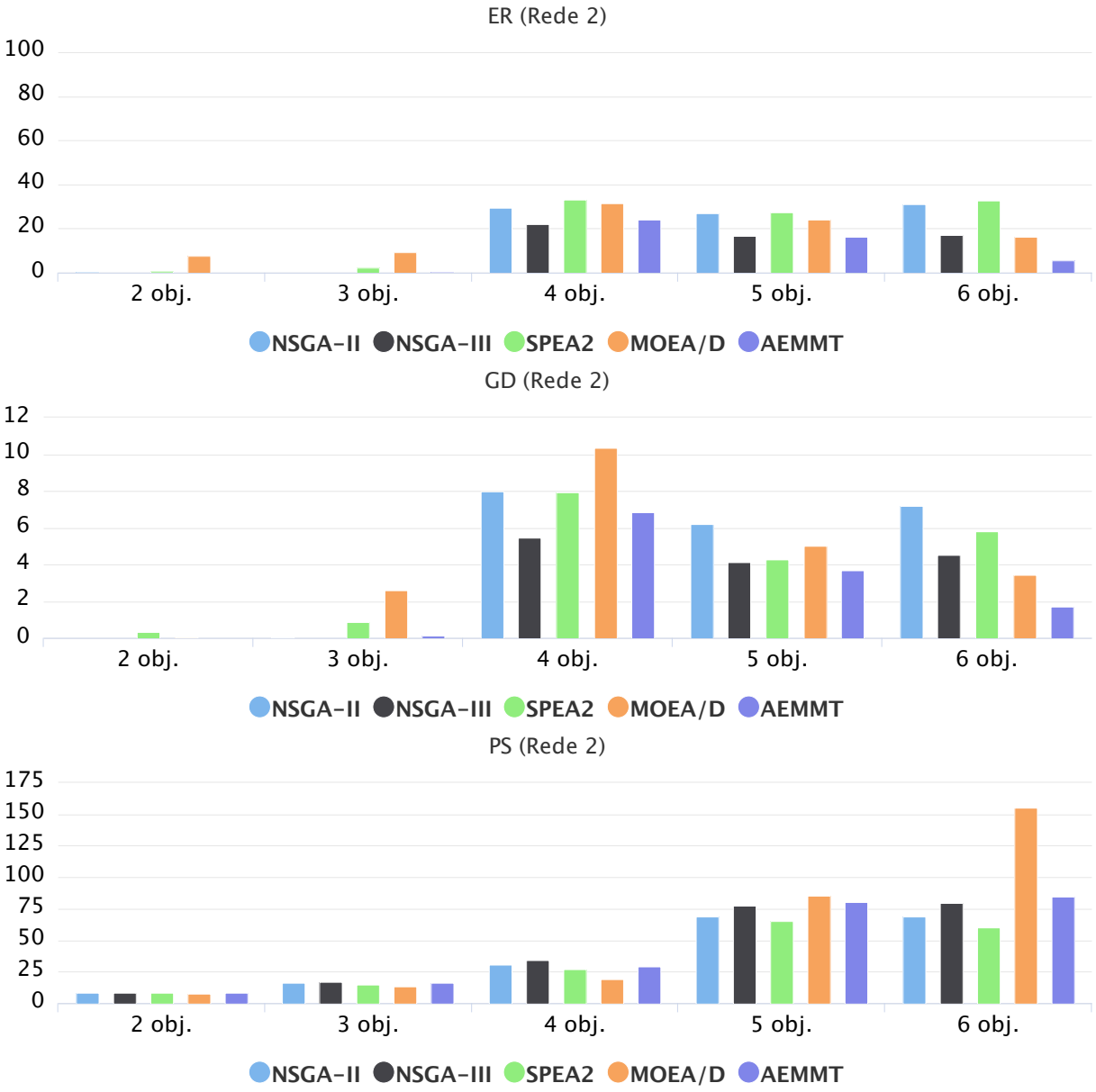

Figura 27 – Desempenho dos algoritmos na 1ª etapa para o PRM na rede 2

soluções não-dominadas.

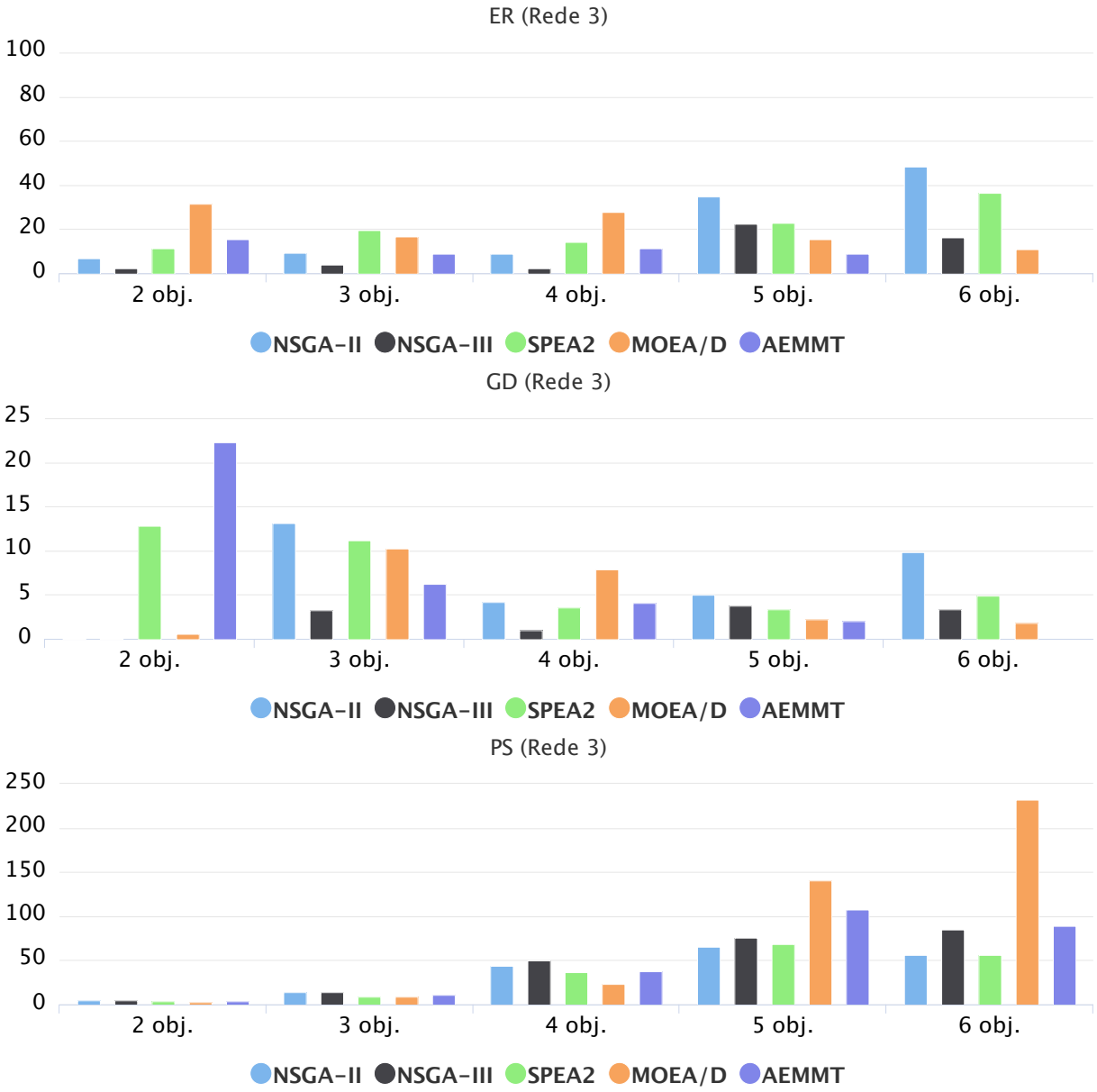

Figura 28 – Desempenho dos algoritmos na 1ª etapa para o PRM na rede 3

A rede 3 é a mais complexa analisada nesta etapa dos experimentos. Analisando-se os resultados dos gráficos apresentados na Figura 28, a tendência já observada nas demais instâncias é mantida, na qual o NSGA-III continua sendo o melhor método para as formulações com poucos objetivos. Até 4 objetivos, em todas as métricas, o NSGA-III apresenta os melhores resultados. Para 5 e 6 objetivos, o AEMMT apresenta menor ER e GD, enquanto o MOEA/D consegue maior PS.

A fim de fazer uma análise geral do PRM, os desempenhos dos algoritmos em todas as instâncias do problema (redes 1, 2 e 3) são consolidados nos gráficos da Figura 29. Assim como no PMM, essa consolidação é obtida através da média aritmética dos valores

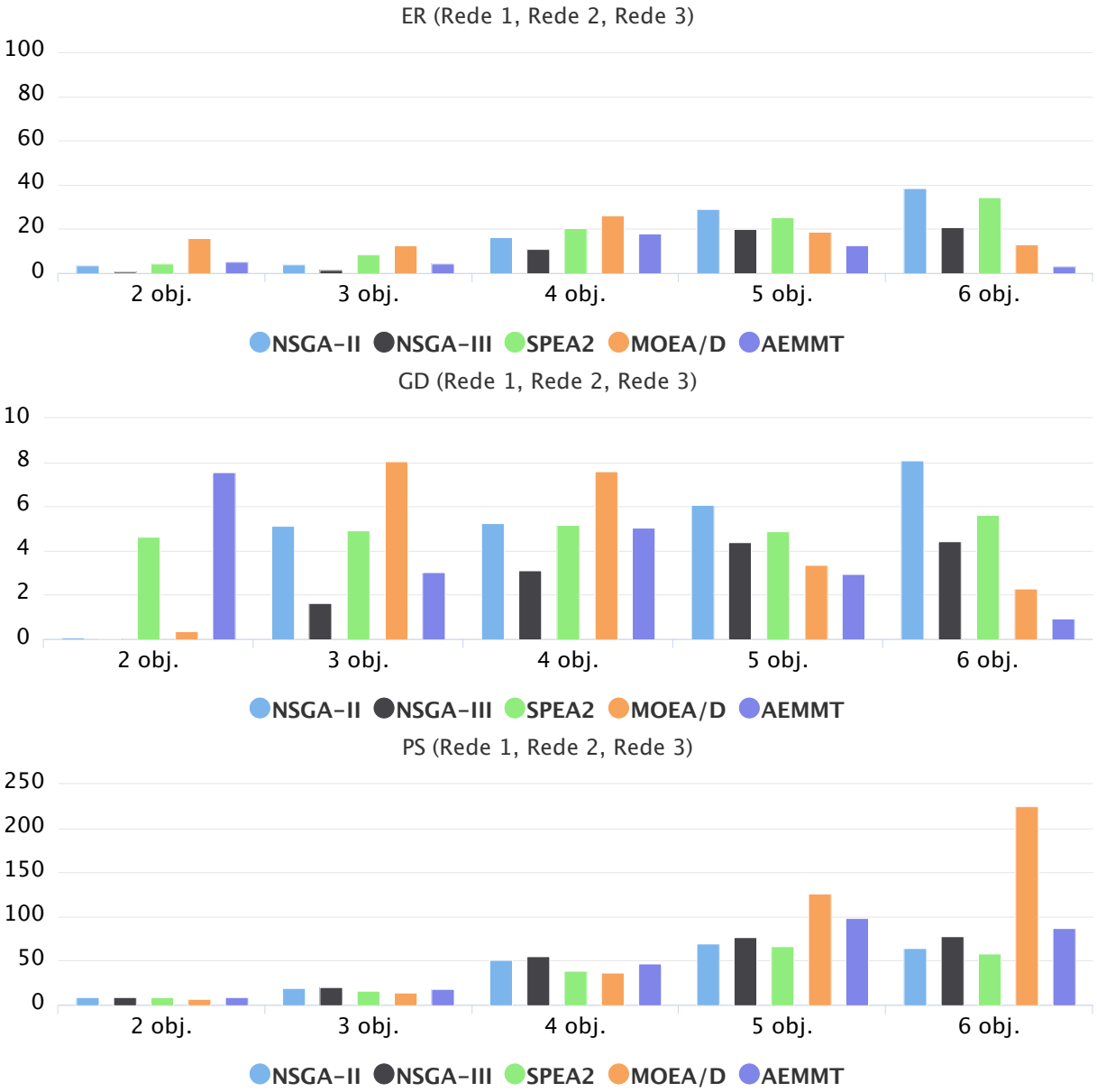

Figura 29 – Resultado consolidado da 1ª etapa considerando o PRM nas redes 1, 2, e 3

das métricas em cada instância. Apesar de ser esperado que o NSGA-II e o SPEA2 fossem os melhores métodos para poucos objetivos, a partir dos gráficos da Figura 29, observa-se que o NSGA-III obteve o melhor resultado para as formulações com 2, 3 e 4 objetivos. O NSGA-II também atinge bons resultados para 2 e 3 objetivos, mas o SPEA2 apresenta um GD relativamente ruim. Considerando problemas de 5 e 6 objetivos, a escolha do algoritmo depende do intuito da busca. Se a quantidade encontrada de soluções do Pareto for mais relevante, o MOEA/D é mais indicado. Por outro lado, se um menor erro é preferível, então o AEMMT é a melhor opção.

Considerando-se ambos os problemas, PMM e PRM, os algoritmos NSGA-II, SPEA2 e NSGA-III são os que geram melhores resultados para problemas com poucos objetivos. O desempenho dos algoritmos clássicos (NSGA-II e SPEA2) diminui consideravelmente à medida que se aumenta o número de objetivos. Em contrapartida, o desempenho dos métodos AEMMT e MOEA/D melhora a partir de quatro objetivos, tornando-os mais indicados para problemas *many-objectives*.

Outra observação que pôde ser feita a partir dos experimentos desta etapa é que o AEMMT perde apenas em PS para o MOEA/D. Uma das características do AEMMT é a limitação no tamanho da tabela de não dominância (arquivo com os indivíduos não dominados), o que levanta a seguinte questão: se não houvesse esse limite, o AEMMT alcançaria melhor resultado em todas as métricas? Visando responder tal questão, foram executados novos experimentos para avaliar o desempenho de uma variação do AEMMT (AEMMT-F), na qual não existe um limite para o tamanho da tabela de não dominância. Na maioria dos experimentos, o AEMMT-F apresentou uma taxa de erro maior que sua versão original. Em contrapartida, essa variação alcançou valores médios para as métricas ER e PS melhores que aqueles obtidos pelo MOEA/D, principalmente para as formulações com 5 e 6 objetivos (*many-objective*). Essa superioridade foi confirmada através de um teste de hipótese paramétrico, chamado de teste Z (*Z-test*) com 0,1% de significância ($\alpha = 0,1$) (FRANçA; MARTINS; OLIVEIRA, 2018). Os novos resultados obtidos pelo AEMMT-F permitem dizer que ele é preferível em relação ao MOEA/D, mas não é necessariamente melhor que o algoritmo AEMMT original, que continua sendo a melhor opção em termos de taxa de erro (ER).

Considerando que essa dissertação visa estratégias para lidar com problemas com muitos objetivos, para as próximas etapas serão utilizados apenas cenários a partir de 4 objetivos. Da mesma forma, os AEMOs tradicionais (NSGA-II e SPEA2) não serão mais avaliados e os experimentos serão realizados apenas com os algoritmos *many-objective*.

6.3 Análise das estratégias e configurações para o MACO/NDS

Na elaboração do novo algoritmo ACO para a manipulação de problemas *many-objective*, fez-se necessário um estudo sobre os métodos para a construção das soluções pelas formigas no PRM. Além disso, foi também preciso testar modelos de atualização de feromônio e ideias sobre os parâmetros de entrada α e β do algoritmo. A fim de propor o novo método de otimização *many-objective* e um modelo de aplicação para o PRM, realizou-se a segunda etapa de experimentos, onde várias configurações possíveis do algoritmo e do modelo foram testadas.

Inicialmente, um modelo de construção das soluções do PRM para colônia de formigas foi concebido de acordo com os experimentos realizados nesta etapa. O mesmo processo não foi realizado para o PMM, pois o modelo descrito em (ALAYA; SOLNON; GHÉDIRA, 2004) para algoritmos ACO se mostrou adequado desde os primeiros experimentos com o novo modelo *many-objective*. Além disso, nesses experimentos também avaliaram-se diferentes aspectos sobre a atualização de feromônios e parâmetros de entrada no ACO, resultando em particularidades do algoritmo proposto.

Os experimentos dessa etapa foram realizados em duas fases. A primeira investiga o impacto de diferentes modelos de construção das soluções no desempenho do algoritmo proposto. Nesses experimentos, optou-se por simplificar o problema investigado a partir da adoção de um cenário com um único objetivo (mono-objetivo), possibilitando a avaliação, de maneira isolada, do processo de construção das soluções. Os experimentos da segunda fase visam analisar a influência de mudanças nas estratégias e configurações adotadas pelo MACO/NDS para resolver problemas multiobjetivos. Essas mudanças envolvem a construção de soluções por amostragem, formas de depósito de feromônio e dinamização dos parâmetros de entrada. Os experimentos de cada fase são descritos a seguir.

Nos experimentos com o problema mono-objetivo, foram avaliadas quatro estratégias (descritas na Seção 5.2):

1. Formiga única: o espaço de busca é explorado de forma aleatória, mas sempre considerando apenas a vizinhança da posição atual da formiga.

2. Múltiplas formigas: uma formiga para cada destino, unindo os caminhos gerados por cada agente em uma árvore.

3. Formiga com sobreposição quântica: uma formiga explora o espaço de busca de forma aleatória, podendo estar em vários nós ao mesmo tempo. Essa estratégia elimina o problema de localidade na busca.

4. Formigas invertidas: uma formiga para cada destino, mas ao invés construírem uma trajetória da raiz ao destino, fazem o caminho contrário, ou seja, partem do destino e tentam encontrar a raiz.

O problema mono-objetivo utilizado consiste em minimizar o valor do produto entre custo e atraso ($f(x) = custo(x) \times delay(x)$) de uma árvore. Portanto, quanto menor esse valor, melhor a árvore obtida. Além das quatro estratégias avaliadas, também foi implementada uma modificação do algoritmo de Prim (PRIM, 1957) para servir como referência de uma solução potencial para o PRM. Cada estratégia foi executada cinco vezes e a tabela 4 mostra o resultado da melhor execução de cada estratégia. Nessa tabela, a coluna "resultado" representa a soma dos valores de $custo \times delay$ das arestas da árvore obtida como solução. Ou seja, quanto menor esse valor, melhor a solução obtida.

Tabela 4 – Desempenho em função da estratégia de construção de uma solução para o PRM

Estratégia	Rede	Resultado	Tempo (s)
Prim	R_1	3,28	0,02
1	R_1	3,04	2,49
2	R_1	3,03	5,94
3	R_1	3,00	3,88
4	R_1	3,00	3,16
Prim	R_2	3,13	0,01
1	R_2	3,34	4,94
2	R_2	3,26	13,22
3	R_2	3,13	10,69
4	R_2	3,30	5,549
Prim	R_3	7,96	0,024
1	R_3	8,13	4,212
2	R_3	8,23	25,60
3	R_3	7,48	9,82
4	R_3	8,11	6,93
Prim	R_4	1,80	0,02
1	R_4	2,34	4,71
2	R_4	2,32	12,47
3	R_4	1,85	11,01
4	R_4	1,97	4,93
Prim	R_5	6,34	0,01
1	R_5	6,12	6,87
2	R_5	6,33	17,38
3	R_5	5,85	14,76
4	R_5	5,85	8,42

Como pode ser observado na tabela, a estratégia número 3, que usa a ideia de formiga com sobreposição, obteve o melhor valor para a função objetivo (resultado). Os valores obtidos por essa estratégia foram inclusive melhores que os obtidos pelo algoritmo utilizado

como referência (Prim). Em contrapartida, ela foi a segunda pior estratégia em relação ao tempo. Por essa razão, propõe-se a ideia de amostragem explicada na Seção 5.2. Ao invés de se utilizar todo o conjunto de exploração, a construção da solução é realizada sobre uma amostra dele. Em nossos experimentos para o PRM, a amostragem é de 10 elementos. A Tabela 5 mostra uma comparação dos melhores valores obtidos utilizando a estratégia 3, com e sem essa amostragem.

Tabela 5 – Resultados obtidos a partir da estratégia 3 de acordo com o espaço de exploração considerado (com e sem amostragem)

Amostragem	Rede	Resultado	Tempo (s)
s/ amostragem	R_1	3	3,88
c/ amostragem	R_1	3	3,41
s/ amostragem	R_2	3,13	10,69
c/ amostragem	R_2	3,13	7,60
s/ amostragem	R_3	7,48	9,82
c/ amostragem	R_3	7,48	7,26
s/ amostragem	R_4	1,85	11,01
c/ amostragem	R_4	1,78	7,28
s/ amostragem	R_5	5,85	14,76
c/ amostragem	R_5	5,76	10,03

Como pode ser observado na Tabela 5, a amostragem não só diminuiu o tempo do algoritmo como também melhorou o resultado para as redes mais complexas (R_5 e R_6). O aumento da qualidade da solução pode ser atribuído ao maior grau de aleatoriedade dado ao algoritmo, similar ao que acontece nos algoritmos genéticos quando se utiliza operações como a mutação ou seleções por torneio.

A partir desses resultados, implementou-se a primeira versão do novo algoritmo ACO *many-objective*, denominado MACO/NDS-alpha. O MACO/NDS-alpha adota a estratégia 3 (formiga com sobreposição quântica) e a amostragem do espaço de exploração. Considerando essa primeira versão do algoritmo no PRM *many-objetive*, percebeu-se que o AEMMD atingiu um desempenho muito superior ao do novo algoritmo, como apresentado na Tabela 6. Nessa etapa, o AEMMD foi adotado como referência, uma vez que, até onde se sabe, é o algoritmo com o melhor desempenho para o PRM *many-objective* (LAFETÁ, 2016). A diferença de desempenho entre os algoritmos motivou a implementação de novas estratégias no modelo proposto. Assim, depois de alguns experimentos preliminares, duas modificações foram incorporadas ao modelo baseado em ACO:

- Depósito de feromônio baseado na qualidade da aresta (ou do item, no caso do PMM): em um problema multiobjetivo, ao depositar feromônios sobre as arestas correspondentes às soluções não-dominadas, normalmente é utilizada a mesma quantidade independentemente da solução e da aresta. Isso ocorre porque, segundo a relação de não-dominância, ambas opções são igualmente boas. Entretanto, nossa

nova estratégia muda esse conceito, relacionando a quantidade de feromônios depositada à qualidade da aresta. Nesse contexto, considerando que é um problema de minimização, a quantidade passa a ser inversamente proporcional à soma dos pesos da aresta.

- Dinamização do parâmetro de entrada β: esse parâmetro controla a importância da heurística no cálculo das probabilidades de cada aresta (ou item, no PMM) fazer parte da solução. Nossa proposta é diminuir um pouco a importância da heurística, dando maior peso à informação de feromônio sempre que, após uma iteração, não for encontrada uma nova solução. Esse processo se repete a cada iteração até que uma nova solução seja encontrada e o β seja restaurado ao seu valor original (fornecido como entrada para o algoritmo).

O algoritmo resultante foi chamado de *Many-objective Ant Colony Optimization based on Non-dominated Decomposed Sets* (MACO/NDS). Considerando a formulação do PRM com seis objetivos, a Tabela 6 mostra as diferenças entre os desempenhos dos algoritmos AEMMD, MACO/NDS antes das alterações no depósito de feromônios e do parâmetro β (MACO/NDS-alpha) e do algoritmo após as modificações (MACO/NDS). A comparação é feita com base nos valores médios das métricas de desempenho multiobjetivo (ER, GDp e PS) em 100 execuções.

Tabela 6 – Análise comparativa entre as implementações do MACO/NDS e o AEMMD no PRM multiobjetivo (6 objetivos)

Algoritmo	Rede	ER	GDp	PS
AEMMD	R_1	6,59	0,38	502,8
MACO/NDS-alpha	R_1	18,42	0,30	337,6
MACO/NDS	R_1	11,57	0,36	424,2
AEMMD	R_2	7,58	0,49	296,6
MACO/NDS-alpha	R_2	11,75	0,47	284,4
MACO/NDS	R_2	11,18	0,37	300
AEMMD	R_3	11,73	0,19	388,6
MACO/NDS-alpha	R_3	36,47	0,16	206
MACO/NDS	R_3	30,20	0,25	232,4
AEMMD	R_4	35,88	0,17	234
MACO/NDS-alpha	R_4	54,48	0,11	150,2
MACO/NDS	R_4	50,57	0,15	186,6
AEMMD	R_5	32,67	0,20	181,2
MACO/NDS-alpha	R_5	32,95	0,22	168,6
MACO/NDS	R_5	32,67	0,30	160,8

Na Tabela 6, pode-se observar que, na maioria dos casos, as duas alterações propostas melhoraram o resultado do MACO/NDS nas métricas ER e PS (nas quais o AEMMD é superior). Portanto, o algoritmo final proposto inclui essas duas alterações. Entretanto,

mesmo com a melhoria na eficiência do MACO/NDS, o AEMMD ainda apresenta os melhores resultados para as métricas de desempenho.

Uma vez que foram definidas as estratégias e configurações que melhoram a qualidade dos resultados obtidos pelo método proposto, as próximas etapas visam analisar o desempenho do MACO/NDS em comparação com outros algoritmos *many-objective* bio-inspirados.

6.4 Análise comparativa entre o MACO/NDS e os AEMOs *many-objective*

Na terceira etapa, os experimentos visam avaliar a eficiência dos métodos de otimização *many-objective* NSGA-III, MOEA/D, AEMMT, AEMMD e do algoritmo proposto, MACO/NDS, nos problemas da mochila e do roteamento multicast. Nessa etapa, foram utilizadas formulações com 4, 5 e 6 objetivos, avaliando-se 3 redes no PRM e problemas com 30, 40 e 50 itens no PMM. Os resultados dos experimentos revelam as vantagens e fraquezas do algoritmo proposto neste trabalho, possibilitando a identificação dos cenários nos quais sua utilização é mais adequada e as formas de melhorá-lo em cenários onde os seus resultados foram desfavoráveis. Esses experimentos representam a primeira vez em que o algoritmo AEMMD foi implementado para o PMM e também a primeira vez que o algoritmo proposto (MACO/NDS) foi implementado para o PMM e o PRM.

Por abordar apenas problemas com muitos objetivos (*many-objective*), nesses experimentos foram descartados os AEMOs clássicos (NSGA-II e SPEA2) e incluídos os algoritmos AEMMD e o MACO/NDS (algoritmo proposto neste trabalho). Em todas as execuções, os algoritmos utilizaram os mesmos parâmetros de configuração, os quais estão descritos na Tabela 7. No caso do AEMMT e do AEMMD, o número de gerações (marcado com asterisco) deve ser multiplicado pelo tamanho da população. Esse ajuste visa equiparar a quantidade de soluções avaliadas, dado que esse algoritmos realizam apenas um *crossover* por geração. Durante a análise dos resultados, foram empregadas todas as métricas de desempenho utilizadas nas etapas anteriores: erro (ER), distância (GDp) e número de soluções do Pareto encontradas (PS). Além dessas três métricas multiobjetivo, mediu-se o tempo de execução do algoritmo (Tempo). A medida de tempo, como não pode ser obtida através da execução paralela dos algoritmos e necessita de exclusividade sobre a máquina, considerou a média de 3 execuções de cada algoritmo em cada cenário testado. As demais métricas foram obtidas através das médias entre 100 execuções dos 18 cenários na lista a seguir:

- PRM: 3 formulações de objetivos (P_4 a P_6) e 3 redes (R_1, R_2 e R_3). Tanto as formulações quanto às redes foram descritas na Seção 4.2.
- PMM: 3 formulações de objetivos (4 a 6) e 3 instâncias (30, 40 e 50 itens).

Assim como na etapa 1, o Pareto aproximado foi pré-definido a partir de múltiplas execuções dos algoritmos. A quantidade de soluções pertencentes ao Pareto obtido para cada instância dos problemas investigados é apresentada na Tabela 8. É importante destacar que os conjuntos de soluções não dominadas (Pareto aproximado) adotados nesta etapa para o PRM são diferentes dos obtidos na primeira etapa para as mesmas redes (Tabela 2). Isso ocorreu porque, os conjuntos de Pareto originalmente utilizados (FRANÇA et al., 2017) foram complementados com novas soluções não dominadas encontradas em execuções preliminares dos algoritmos implementados (MACO/NDS e AEMMD). Observa-se que a cardinalidade dos conjuntos de Pareto encontrados para esta etapa de experimentos (Tabela 8) aumentou em relação aos conjuntos obtidos para a primeira etapa. Os Paretos aproximados também foram recalculados para o PMM com 30 e 50 itens, gerando resultados ligeiramente maiores que os apresentados na Tabela 2. Na Tabela 8, o valor correspondente ao cenário do PMM com 6 objetivos e 50 itens destaca-se pela quantidade de elementos em seu Pareto, isso se deve ao tamanho do espaço de busca que é multiplicado por 2^{10} em relação à instância anterior (40 itens) e à formulação de 6 objetivos, que faz com que uma parte muito grande desse conjunto seja considerada não dominada. Esse comportamento dificulta a obtenção de Paretos aproximados estáveis para cenários mais complexos do PMM.

Tabela 7 – Parâmetros utilizados pelos AEMOs na 3ª etapa, de acordo com o problema tratado

Parâmetro	**PRM**	**PMM**
Tamanho da população	90	150
Número de gerações*	100	100
Taxa de crossover	100%	100%
Taxa de mutação	20%	5%
Tamanho da vizinhança (MOEA/D)	10	10
Tamanho das tabelas (MEAMT)	30	50
Tamanho da tabela de dominância (MEAMT)	90	150
Número de divisões (NSGA-III)	8	8
α, β, ρ (MACO/NDS)	1, 2, 0,3	1, 4,3; 0,3
Intervalo de valores para os feromônios (MACO/NDS)	[0,1; 0,9]	[0,1; 0,9]
Tamanho das amostras (MACO/NDS)	10	25% do nº de itens
Tamanho do grupo de estruturas ativas (MACO/NDS)	5	5

As Figuras 30, 31 e 32 mostram, respectivamente, o desempenho dos algoritmos para o PMM de 30, 40 e 50 itens. Nas Figuras 34, 35 e 36 são apresentados os resultados para o PRM considerando as redes R_1, R_2 e R_3, respectivamente. Uma análise consolidada, com a média entre as três instâncias de cada problema, é apresenta na Figura 33 para o PMM e na Figura 37 para o PRM.

Tabela 8 – Fronteira de Pareto estabelecida para os cenários investigados na 3ª etapa de experimentos

	PRM			**PMM**		
Objetivos	R1	R2	R3	30 itens	40 itens	50 itens
4	122	553	1349	425	1199	1012
5	75	372	712	1769	3862	5467
6	60	660	1283	5828	6491	55471

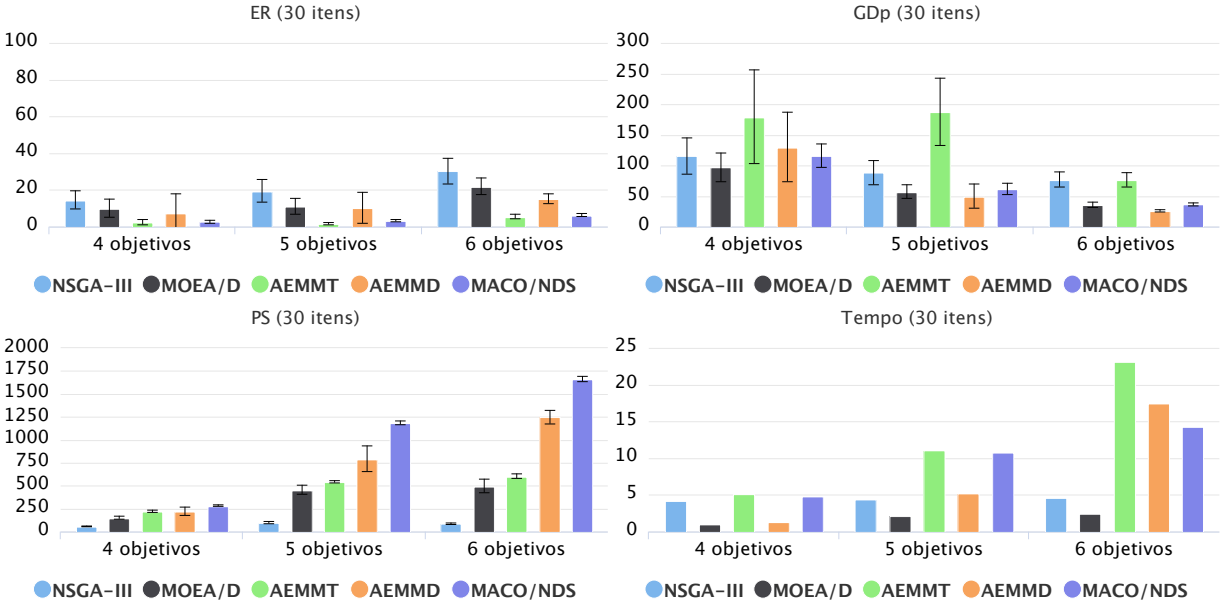

Figura 30 – Desempenho dos algoritmos na 3ª etapa para o PMM com 30 itens

A Figura 30 retrata a versão mais simples do problema da mochila (30 itens). Nesse cenário, o AEMMT apresenta a menor taxa de erro dentre os algoritmos *many-objective*, seguido pelo MACO/NDS e pelo AEMMD. Considerando a métrica GDp e a formulação com 4 objetivos, os melhores resultados são encontrados pelo MOEA/D, seguido pelo MACO/NDS. Com 5 e 6 objetivos, o AEMMD produz o menor *GD*, sendo acompanhado de perto pelo MACO/NDS. Na métrica *PS*, o MACO/NDS é o algoritmo com melhor desempenho para todas as formulações de objetivo. Na sequência, os algoritmos com os melhores valores de *PS* são: AEMMD, AEMMT, MOEA/D e NSGA-III. Destre esses algoritmos, o único com um limite fixo no tamanho do Pareto é o NSGA-III. Portanto, é esperado que possua um valor de *PS* menor. Quanto ao tempo, o MOEA/D é o algoritmo mais rápido, enquanto que o AEMMT é o mais lento. No geral, o MACO/NDS apresentou o melhor equilíbrio entre as métricas de desempenho, revelando-se a melhor opção, nessa instância (30 itens), para formulações com muitos objetivos. Além disso, o MACO/NDS apresenta os menores desvios padrões, ou seja, é o método com maior estabilidade quanto à qualidade de suas soluções, independentemente da execução.

Os resultados obtidos para o PMM de 40 itens é apresentado na Figura 31. O AEMMT

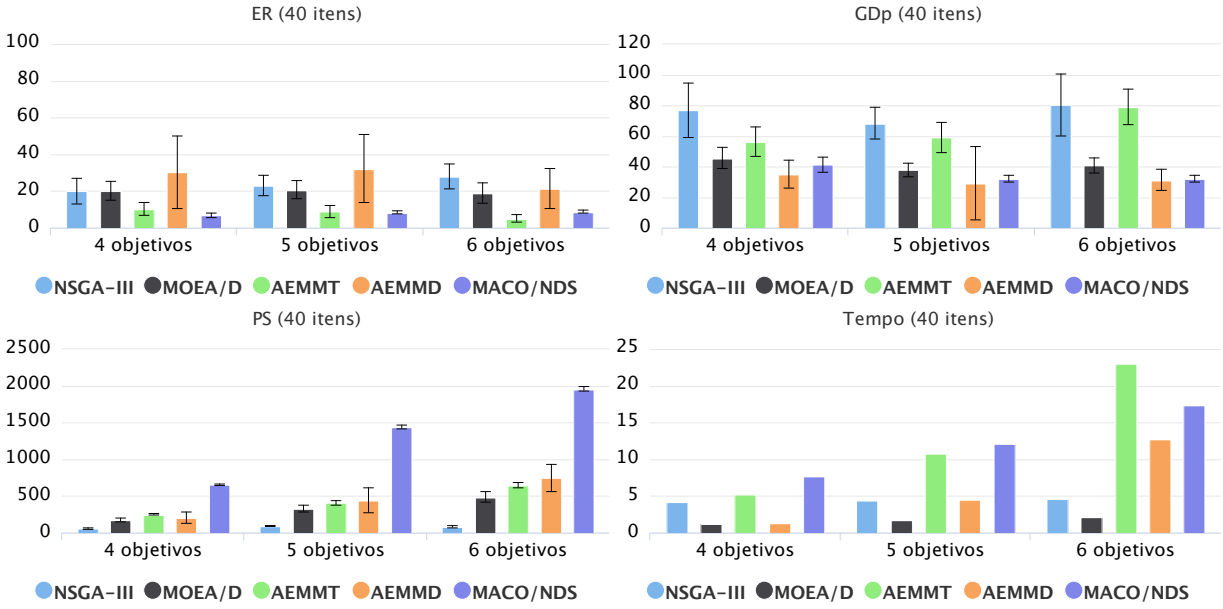

Figura 31 – Desempenho dos algoritmos na 3ª etapa para o PMM com 40 itens

e o MACO/NDS apresentam as menores taxas de erro, sendo que o MACO/NDS é melhor para 4 e 5 objetivos (6,9% e 8,4%, respectivamente), enquanto o AEMMT obtém o melhor resultado para 6 objetivos (5,1% contra 8,7% do MACO/NDS). Os melhores valores de GDp foram encontrados pelo AEMMD e MACO/NDS, sendo o primeiro um pouco melhor que o segundo. Em contrapartida, o AEMMD apresenta uma alta variação nos resultados, indicando que algumas execuções produzem soluções muito próximas do Pareto enquanto outras nem tanto. Considerando o tamanho dos Paretos encontrados (PS), novamente o MACO/NDS retorna os melhores valores independentemente da formulação de objetivos. Em termos do tempo médio de execução, o MOEA/D é o algoritmo mais rápido (5,2 segundos ao somar o tempo das 3 formulações de objetivos), enquanto o AEMMT e o MACO/NDS são os mais lentos (39 e 37,2 segundos, ao somar o tempo das 3 formulações de objetivos, respectivamente).

A Figura 32 mostra o desempenho dos algoritmos no PMM com 50 itens. Como pode ser observado, com exceção do GDp, as demais métricas apresentaram um comportamento similar ao das instâncias anteriores. O MACO/NDS e o AEMMT continuam sendo os algoritmos com menores taxas de erro. Enquanto o MACO/NDS consegue menor ER em 4 e 5 objetivos, o AEMMT apresenta melhor resultado em 6 objetivos. O MACO/NDS obtém os melhores valores de GDp e PS, e o MOEA/D é o algoritmo mais rápido. Dentre os algoritmos analisados, o MACO/NDS é o que apresenta o melhor compromisso entre as métricas e a melhor estabilidade quanto à qualidade de soluções (menor desvio padrão).

A fim de se fazer uma análise geral dos resultados, a média entre os 3 cenários é apresentada nos gráficos da Figura 33. As taxas de erro são muito baixas para o AEMMT e o MACO/NDS, quando comparados aos demais algoritmos. Os resultados em GDp são

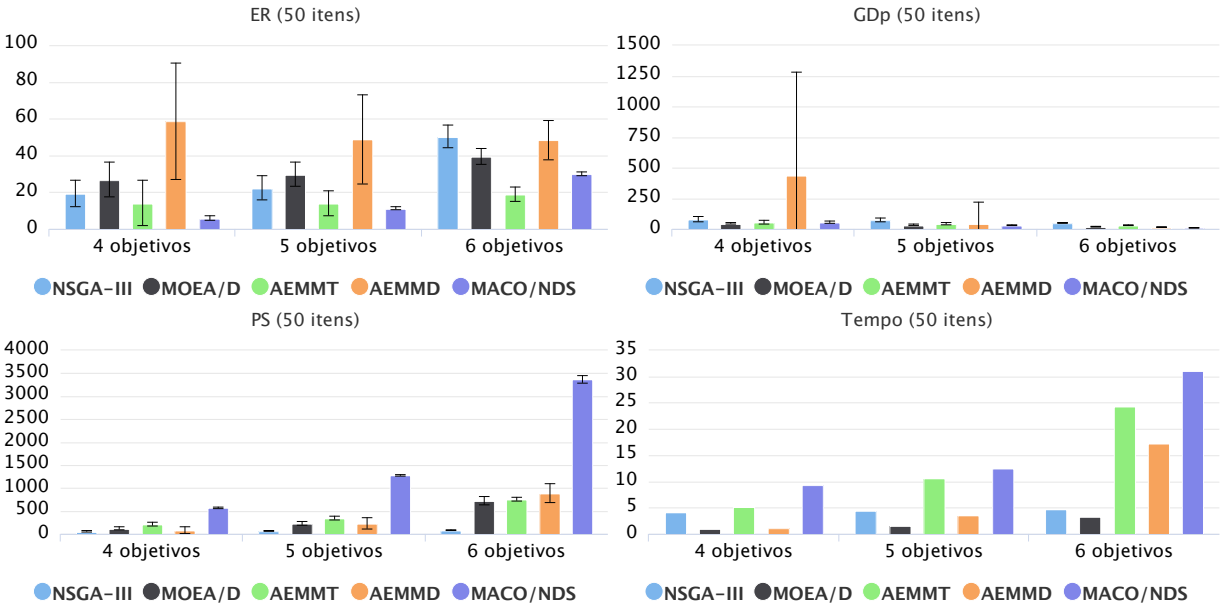

Figura 32 – Desempenho dos algoritmos na 3ª etapa para o PMM com 50 itens

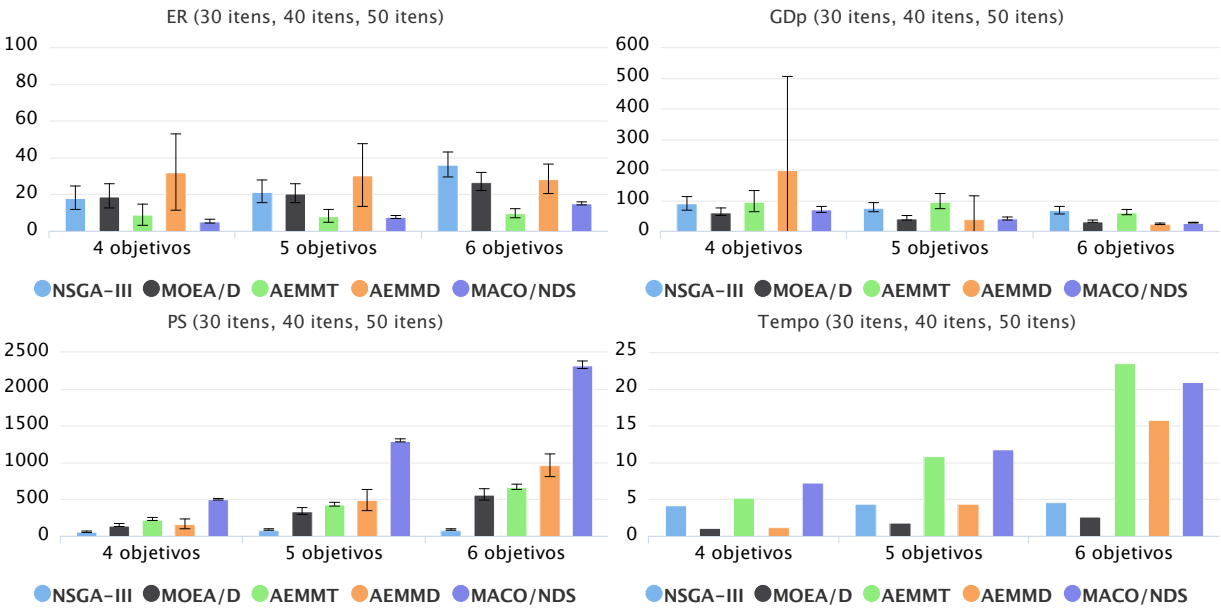

Figura 33 – Resultado consolidado da 3ª etapa considerando o PMM com 30, 40 e 50 itens

bons para a maioria dos métodos. Apenas o AEMMD apresenta um valor alto de *GDp* e com elevado desvio padrão para o problema de 4 objetivos (200 com desvio padrão de 504). O MOEA/D produz o melhor *GDp* no problema de 4 objetivos (62,9), enquanto que, para 5 e 6 objetivos, o AEMMD (40,7 para 5 objetivos e 23,6 para 6) e o MACO/NDS (41,1 para 5 objetivos e 26,8 para 6) conseguem valores melhores que o MOEA/D (43 para 5 objetivos e 31,9 para 6), mas ainda similares. É importante notar que os desvios padrões no AEMMD são altos, representando uma certa instabilidade do algoritmo em gerar boas soluções. Em termos da cobertura da fronteira de Pareto (*PS*), o MACO/NDS alcançou os melhores valores em todas as formulações de objetivos. Analisando o tempo de execução médio, o AEMMT, o AEMMD e o MACO/S variam bastante com o número de objetivos, enquanto os demais são estáveis. O MOEA/D é o algoritmo mais rápido entre os avaliados nesta etapa dos experimentos.

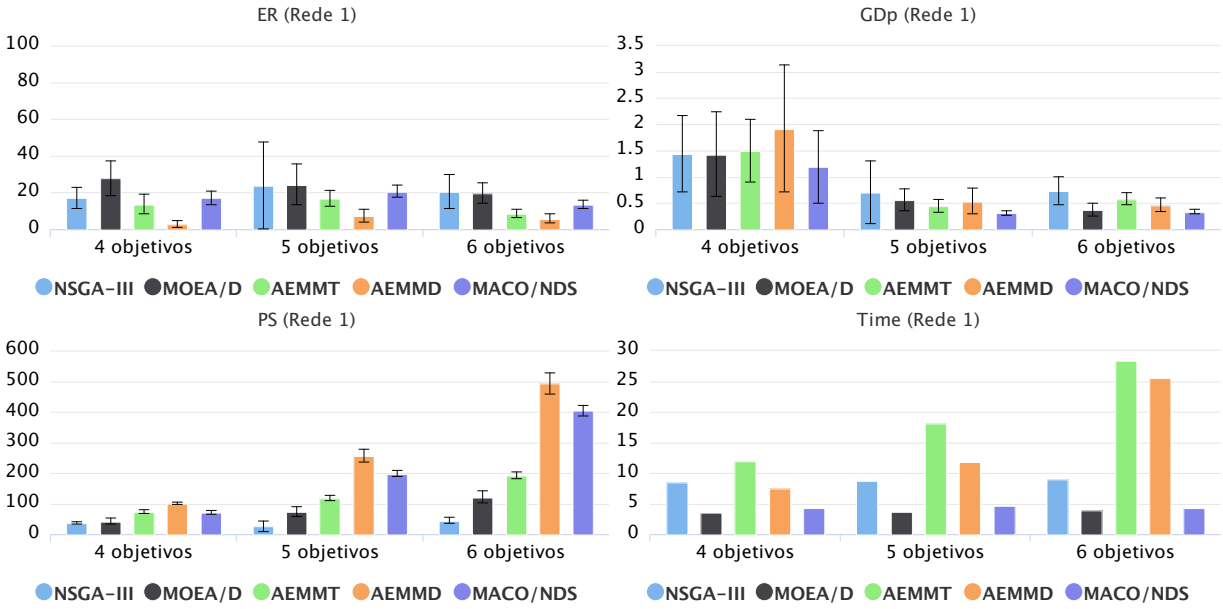

Figura 34 – Desempenho dos algoritmos na 3ª etapa para o PRM na rede 1

Os gráficos correspondentes ao PRM na rede 1 são apresentados na Figura 34. Em relação à taxa de erro, o AEMMD obtém os menores valores em todas as formulações de objetivos. Em seguida, aparecem o AEMMT e o MACO/NDS, respectivamente, mas com valores próximos. Por fim, os piores valores de ER foram obtidos pelo NSGA-III e pelo MOEA/D. Com relação ao *GDp*, é possível observar que o MACO/NDS e o MOEA/D atingem desempenhos bem próximos, sendo o MACO/NDS o melhor entre os dois. Considerando o tamanho das fronteiras de Pareto encontrada (*PS*), o AEMMD é o melhor algoritmo, seguido pelo MACO/NDS, AEMMT, MOEA/D e NSGA-III, nessa ordem. O MACO/NDS e o MOEA/D são os algoritmos mais rápidos e permanecem estáveis com o aumento do número de objetivos, enquanto o AEMMT e o AEMMD são os mais lentos e aumentam significativamente com o número de objetivos.

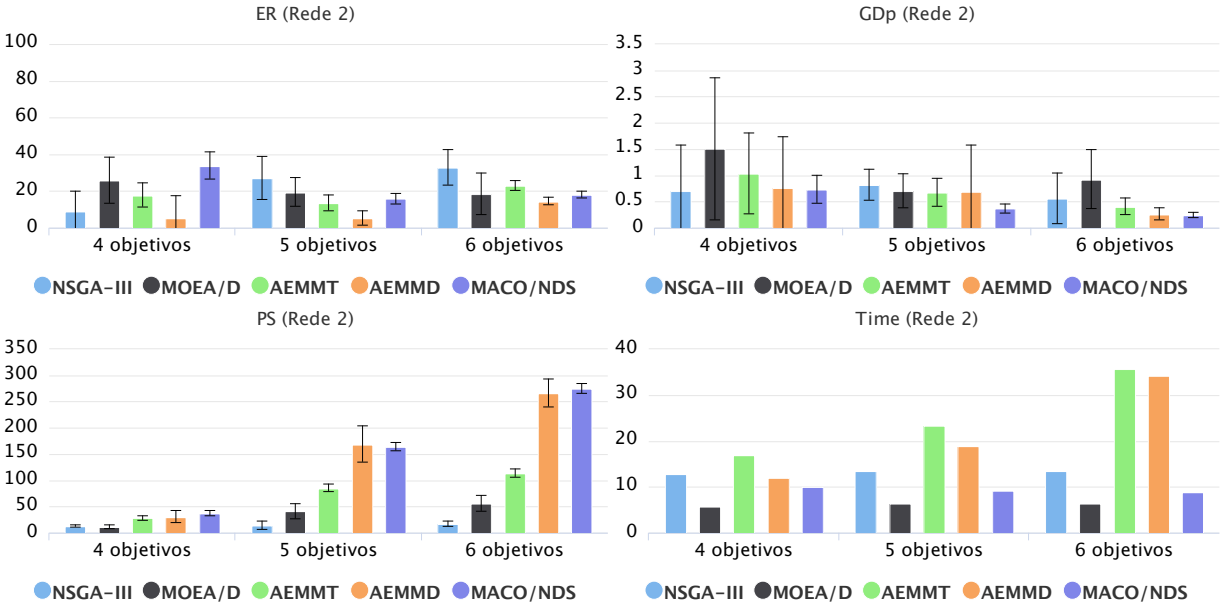

Figura 35 – Desempenho dos algoritmos na 3ª etapa para o PRM na rede 2

A Figura 35 apresenta os resultados obtidos pelos algoritmos para a rede 2. Apesar de mais complexa, os gráficos apresentam um comportamento similar ao da instância anterior (rede 1). O AEMMD obtém as menores taxas de erro, seguido pelo AEMMT. O MACO/NDS é o algoritmo que alcança os menores valores para a métrica *GDp*, sendo seguido de perto pelo AEMMT e pelo AEMMD. Considerando a métrica *PS*, os melhores valores foram encontrados pelo AEMMD, mas com uma diferença muito pequena em relação ao MACO/NDS. Os piores valores de *PS* são retornados pelo NSGA-III. Tal comportamento é decorrência da limitação do algoritmo quanto ao crescimento do Pareto. Analisando o tempo médio de execução, o método mais rápido é o MOEA/D e o mais lento é o AEMMT.

A Figura 36 apresenta o desempenho dos algoritmos no PRM considerando a rede 3, o AEMMT encontra as melhores taxas de erro para 4 (15,2%) e 5 (15,7%) objetivos, enquanto que o MOEA/D consegue o melhor resultado para 6 objetivos (11,5%). O NSGA-III produz o segundo menor *ER* no problema de 4 objetivos (18,9%), mas é o segundo pior para 5 objetivos (22,5%) e o pior com 6 objetivos (52,7%). O menor *GDp* no problema com 4 objetivos é dado pelo MACO/NDS (0,7), enquanto o AEMMT obtém os melhores valores nos problemas com 5 e 6 objetivos (0,3 e 0,1, respectivamente). As maiores fronteiras de Pareto são encontradas pelo AEMMD e MACO/NDS, sendo o AEMMD o melhor entre os dois. O algoritmo mais rápido é o MACO/NDS, executando em tempo 5 vezes menor que o método mais lento (AEMMT).

Para analisar de forma conjunta os resultados das três redes, todos os experimentos são consolidados através da média aritmética das métricas de desempenho avaliadas. Esses resultados consolidados são apresentados na Figura 37. De maneira geral, o NSGA-III

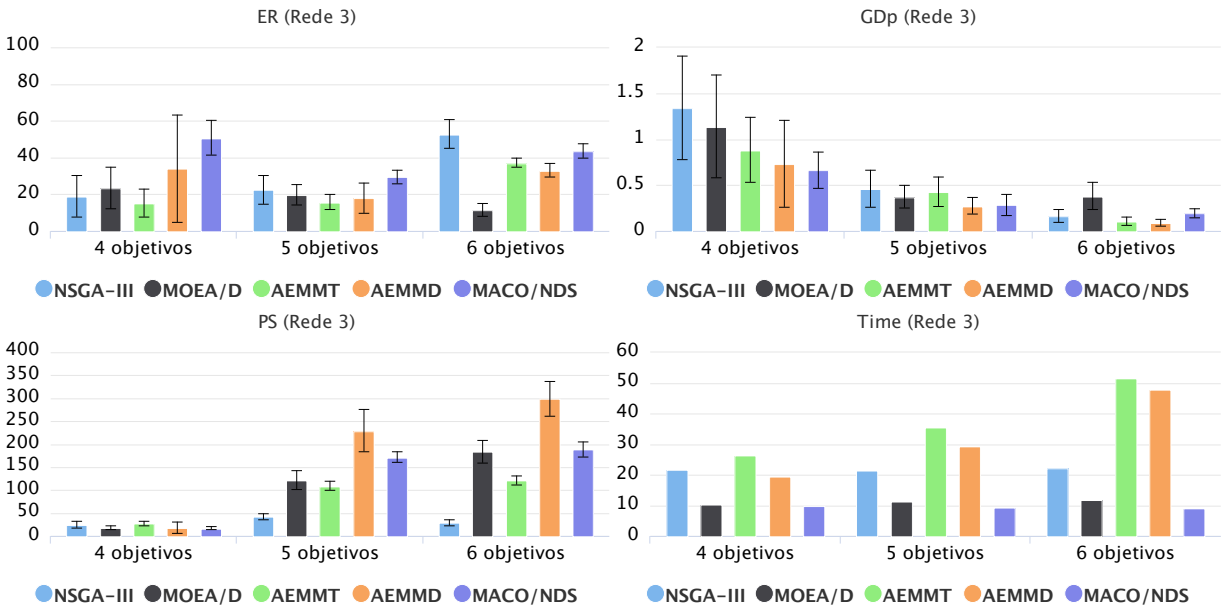

Figura 36 – Desempenho dos algoritmos na 3ª etapa para o PRM na rede 3

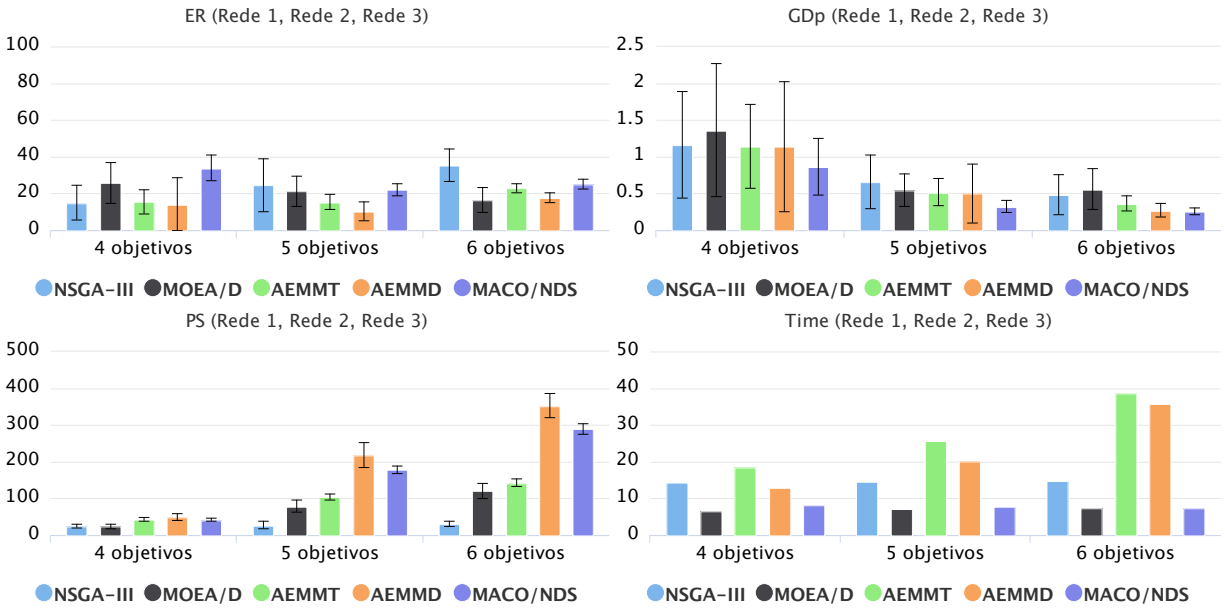

Figura 37 – Resultado consolidado da 3ª etapa considerando o PRM nas redes 1, 2 e 3

consegue soluções de qualidade razoável, mas apresenta baixo PS e um tempo de execução mediano. O MOEA/D obtém as melhores taxas de erro no problema de seis objetivos, mas tem um desempenho modesto nos problemas de 4 e 5 objetivos. O algoritmo também está entre os piores para as métricas GDp e PS. A velocidade de execução do MOEA/D é tão boa quanto a do MACO/NDS, sendo os dois algoritmos mais rápidos. Assim, o uso desse algoritmo pode ser uma boa opção quando o objetivo da busca é conseguir uma baixa taxa de erro em uma execução mais rápida, sem se preocupar com o PS. Analisando-se todas as métricas, os melhores resultados foram obtidos pelo AEMMD e o algoritmo proposto (MACO/NDS). Com relação ao erro, o AEMMD consegue as melhores taxas, enquanto o MACO/NDS, juntamente com o NSGA-III, apresentam os piores valores. Analisando-se o GDp, existe uma pequena diferença entre os algoritmos, mas o MACO/NDS gera melhores soluções que o AEMMD. Além disso, o desvio padrão do MACO/NDS é significativamente menor, tornando-o um método mais estável quanto à qualidade das soluções. Um desempenho similar também é observado para a métrica PS. Entretanto, nesse caso, o AEMMD produz as maiores fronteiras de Pareto que o MACO/NDS. Por outro lado, ao analisar o tempo médio de execução, observa-se que o MACO/NDS é consideravelmente mais rápido, chegando a executar em quase 5 vezes menos tempo. Considerando que o PRM é um problema sensível ao tempo de execução, apesar do AEMMD obter um desempenho razoavelmente superior nas métricas ER e PS, o MACO/NDS é a melhor opção entre os algoritmos testados, pelo menos nos cenários analisados.

A principal diferença no desempenho do algoritmo proposto (MACO/NDS) em relação aos dois problemas investigados (PMM e PRM) é o tempo de execução. Considerando-se os resultados do PMM apresentados na Figura 33, o MACO/NDS é o segundo algoritmo mais lento e seu tempo de execução está altamente relacionado ao número de objetivos. Na Figura 37, que representa o comportamento geral dos algoritmos no PRM, observa-se o oposto, ou seja, o MACO/NDS é o algoritmo mais rápido e seu tempo de execução se mantém estável, independentemente da formulação de objetivos. Concluímos que essa diferença de comportamento é decorrência do tamanho da fronteira de Pareto (soluções não dominadas). O processo de maior custo computacional no MACO/NDS é a atualização dos feromônios, onde é necessário recalcular o conjunto de soluções não-dominadas nd. A cada iteração, para toda solução criada, é necessário passar por todos elementos em nd verificando a relação de não-dominância. Naturalmente, quanto maior o conjunto nd, mais caro se torna o processo. No PRM, foram encontradas fronteiras de Pareto de tamanho razoavelmente pequenos, todas com menos de 350 elementos. Nesse caso, para cada solução criada, é preciso fazer no máximo 350 comparações. No PMM, as fronteiras de Pareto são muito grandes. Por exemplo, no problema com 6 objetivos e 50 itens, a cardinalidade do conjunto nd chega a ser maior que 4500 soluções. Quanto maior a quantidade de objetivos do problema, maior o número de soluções no Pareto e mais lento

será a classificação de soluções não dominadas. Se o conjunto *nd* é pequeno até mesmo para a maior quantidade de objetivos (caso do PRM), o processo será rápido e o tempo de execução não será muito diferente entre as formulações de objetivos. Por outro lado, se a cardinalidade de *nd* é grande e cresce consideravelmente com o aumento na quantidade de objetivos, o tempo necessário para se atualizar os feromônios será muito alto e apresentará grande variação conforme a formulação de objetivos.

A fim de melhor comparar o MACO/NDS aos algoritmos AEMMT e AEMMD, realizou-se o teste de hipótese Z (*Z-test*) com 5% de significância ($\alpha = 0,05$), sendo os resultados mostrados nas Tabelas 9 e 10, respectivamente. Em ambas as tabelas, uma célula de fundo verde representa um cenário onde o MACO/NDS obteve melhor resultado que seu adversário, vermelho significa que o adversário foi melhor e a célula branca indica empate.

Tabela 9 – Testes de hipótese entre o MACO/NDS e o AEMMT para os problemas investigados

	4 objectives			**5 objectives**			**6 objectives**		
Instance	**ER**	**GDp**	**PS**	**ER**	**GDp**	**PS**	**ER**	**GDp**	**PS**
30 items	=	<	>	>	<	>	>	<	>
40 items	<	<	>	<	<	>	>	<	>
50 items	<	=	>	<	<	>	>	<	>
Rede 1	>	<	<	>	<	>	>	<	>
Rede 2	>	<	>	>	<	>	<	<	>
Rede 3	>	<	<	>	<	>	>	>	>

Tabela 10 – Testes de hipótese entre o MACO/NDS e o AEMMD para os problemas investigados

	4 objectives			**5 objectives**			**6 objectives**		
Instance	**ER**	**GDp**	**PS**	**ER**	**GDp**	**PS**	**ER**	**GDp**	**PS**
30 items	<	<	>	<	>	>	<	>	>
40 items	<	>	>	<	>	>	<	>	>
50 items	<	<	>	<	<	>	<	<	>
Rede 1	>	<	<	>	<	<	>	<	<
Rede 2	>	=	>	>	<	<	>	<	>
Rede 3	>	<	<	>	=	<	>	>	<

Analisando-se o comportamento dos algoritmos, a partir da Tabela 9, é possível confirmar a superioridade do MACO/NDS em relação ao AEMMT na maioria dos cenários, em ambos os problemas. Na comparação entre o MACO/NDS e o AEMMD, apresentados na Tabela 10, o algoritmo baseado em formigas foi melhor no PMM, mas pior no PRM. Entretanto, o teste de hipótese não considera o tempo de execução, no qual o MACO/NDS foi muito superior ao AEMMD, o que pode ser observado pela Figura 37.

No PMM, se o tempo de execução é uma preocupação, a melhor alternativa é o algoritmo MOEA/D, que é mais rápido e produz bons resultados. Se a rapidez do algoritmo

não é tão importante e deseja-se encontrar um conjunto de soluções mais próximo do Pareto real, o MACO/NDS é o algoritmo mais indicado. No PRM, o método que representa melhor relação entre qualidade e tempo é o MACO/NDS. Por outro lado, se tempo não é importante, o AEMMD é preferível.

A partir desses experimentos, foi publicado um artigo científico no *2018 IEEE Congress on Evolutionary Computation* (FRANçA; MARTINS; OLIVEIRA, 2018).

6.5 Análise baseada no hiper-volume

Na quarta etapa, os experimentos visam avaliar os algoritmos de otimização *many-objective* NSGA-III, SPEA2-SDE, MOEA/D, AEMMT, AEMMD, MOEA/D-ACO, MO-ACS e MACO/NDS em instâncias complexas do PMM e PRM utilizando a métrica hiper-volume, a qual independe da estimação e/ou do conhecimento prévio da fronteira de Pareto. Esses experimentos são necessários, pois as etapas anteriores consideraram apenas problemas com complexidade razoável, nos quais é possível estimar previamente a Fronteira de Pareto. Em grande parte dos problemas reais, esse conhecimento prévio é impraticável. Por exemplo, com a inclusão de mais itens no PMM ou o uso de redes mais complexas no PRM torna inviável a obtenção da fronteira de Pareto e se faz necessária a utilização de uma métrica não-paramétrica, neste caso o hiper-volume, para avaliar o desempenho dos algoritmos.

A fim de analisar o comportamento dos algoritmos em espaços de busca mais complexos, duas novas redes (redes 4 e 5), , também investigadas em (LAFETÁ, 2016), e duas novas instâncias do problema da mochila (100 e 200 itens) foram utilizadas. Entretanto, o aumento da complexidade influencia diretamente na cardinalidade do espaço de soluções, tornando inviável a obtenção de um conjunto estável de soluções não-dominadas (fronteira de Pareto aproximada). Tal situação é observada no PRM com as redes R_3, R_4 e R_5, e no PMM com 50, 100 e 200 itens. Nesses cenários, não é adequado basear as conclusões das análises em métricas de desempenho paramétricas, as quais são dependentes do Pareto aproximado. Por isso, nos experimentos dessa etapa, as avaliações consideram apenas duas métricas não-paramétricas, hiper-volume e tempo, as quais são independentes do Pareto.

O hiper-volume, junto ao *Inverted Generational Distance* (IGD), são as métricas mais utilizadas na literatura para avaliar algoritmos *many-objective*. Uma definição do IGD pode ser encontrada em (ISHIBUCHI et al., 2014). Como explicado no início deste capítulo, o hiper-volume calcula o volume da figura geométrica formada pelas distâncias das soluções encontradas a um ponto de referência pré-definido. Os pontos de referência utilizados nessa etapa para cada cenário avaliado são apresentados na Tabela 11. Nessa tabela, os valores para o PMM são negativos, pois todos os algoritmos foram implementados para lidar com problemas de minimização. Dessa forma, como no problema da

mochila deseja-se maximizar o valor de lucro carregado na mochila, todos os valores são multiplicados por -1 antes de se iniciar a busca.

Tabela 11 – Pontos de referência e limitações no tamanho do arquivo usados para cada cenário de teste

Instância	**Obj.**	**Ponto de referência**	**Limite**
PMM 50 itens	4	[-15665; -17464; -19122; -16978]	600
	5	[-15948; -15980; -14696; -14800; -14610]	1400
	6	[-16094; -14354; -12511; -14240; -17865; -11801]	4500
PMM 100 itens	4	[-30442; -25071; -30870; -29800]	1300
	5	[-30673; -30266; -30171; -30922; -28821]	3200
	6	[-29389; -27406; -30824; -32040; -30531; -30171]	3400
PMM 200 itens	4	[-64608; -58090; -61540; -59399]	1500
	5	[-62513; -63014; -58939; -64477; -65814]	3000
	6	[-59835; -60434; -65232; -60525; -60843; -60753]	3200
PRM Rede 3	P_4	[0,758449304; 283; 115; 52]	90
	P_5	[0,47215566; 0,776046738; 304; 159; 56]	250
	P_6	[0,471416081; 0,776046738; 310; 159; 56; 83,459]	400
PRM Rede 4	P_4	[0,717830882; 185; 107; 33]	90
	P_5	[0,454221232; 0,776046738; 256; 157; 42]	200
	P_6	[0,457194502; 0,776046738; 239; 157; 40; 80,667]	350
PRM Rede 5	P_4	[0,776046738; 259; 146; 43]	90
	P_5	[0,458729196; 0,776046738; 296; 166; 49]	100
	P_6	[0,458729196; 0,776046738; 287; 177; 48; 101,938]	250

Nesta etapa foram avaliados 8 algoritmos bio-inspirados *many-objective*: NSGA-III, SPEA2-SDE, MOEA/D, AEMMT, AEMMD, MOEA/D-ACO, MOACS, e MACO/NDS. Esses algoritmos foram usados em 3 formulações de objetivo para cada problema (4, 5 e 6 objetivos) e 3 instâncias, totalizando 18 cenários de teste (problema, instância e objetivo). Foram realizadas 30 execuções, para cada algoritmo e em cada cenário, e os resultados foram calculados a partir das médias dos hiper-volumes das execuções. O tempo foi avaliado através da média de três execuções em uma máquina i7-3770k@4.36GHz. Todos os algoritmos foram implementados na linguagem Javascript.

Os parâmetros utilizados na configuração de cada algoritmo podem ser encontrados na Tabela 12. Os parâmetros quantidade de formigas e tamanho dos grupos (marcados com "*" na tabela) dependem, respectivamente, das constantes H e K descritas originalmente em (KE; ZHANG; BATTITI, 2013). Ambos os parâmetros H e K servem como guia para gerar os pesos aleatórios das formigas e dos grupos. Quanto maior o valor de H, maior será o número de formigas. A mesma relação também vale para K e a quantidade de grupos. Em nossos experimentos foi adotado um mesmo valor para K em todos os cenários testados (PMM e PRM). O valor de H também foi constante para os cenários do PRM, mas variou de acordo com a quantidade de objetivos no PMM. No PMM, os valores de H foram: $H = 8$ para 4 objetivos, $H = 6$ para 5 objetivos e $H = 5$ para 6 objetivos. Independentemente dos valores adotados, o número de iterações no laço

principal é ajustado para que sempre se tenha o mesmo número de avaliações (9000 no PRM e 15000 no PMM). Todos os parâmetros foram ajustados de acordo com execuções prévias dos algoritmos.

Tabela 12 – Parâmetros utilizados pelos algoritmos da 4ª etapa, de acordo com o problema tratado.

Parâmetro	**PRM**	**PMM**
Tamanho da população	90	150
Número de comparações	9000	15000
Taxa de crossover	100%	100%
Taxa de mutação	20%	5%
Tamanho da vizinhança (MOEA/D e MOEA/D-ACO)	10	10
Tamanho das tabelas (MEAMT)	30	50
Tamanho da tabela de dominância (MEAMT)	90	150
Número de divisões (NSGA-III)	8	8
α, β, ρ (ACOs)	1; 2; 0,3	1; 4,3; 0,3
Intervalo de valores para os feromônios (ACOs)	[0,1; 0,9]	[0,1; 0,9]
δ (MOEA/D-ACO)	0,2	0,2
H, relacionado à quantidade de formigas (MOEA/D-ACO)*	6	variável
K, relacionado à quantidade de grupos (MOEA/D-ACO)*	3	3
Taxa de elitismo (MOEA/D-ACO)	0,9	0
Tamanho das amostras (MACO/NDS)	10	10
Tamanho do grupo de estruturas ativas (MACO/NDS)	5	5

Nesta etapa, a implementação dos algoritmos NSGA-III e SPEA2-SDE emprega uma estratégia diferente para limitar o tamanho do arquivo, ou seja, a fronteira de Pareto aproximada que é devolvida como solução da execução. Ao invés de impedir o crescimento do conjunto de soluções não-dominadas, além do tamanho máximo da população, esse limite foi ajustado para coincidir com o tamanho médio dos Paretos encontrados pelos algoritmos que retornaram os maiores conjuntos de soluções. Dessa forma, ambos algoritmos (NSGA-III e SPEA2-SDE) podem encontrar Paretos tão grandes quanto os demais métodos. A Tabela 11 mostra os limites utilizados para cada cenário de teste.

Para os algoritmos baseados em colônias de formigas (MOEA/D-ACO, MOACS e MACO/NDS), a fim de se testar o *framework* da forma mais isolada possível, foi utilizada a mesma estratégia de construção da solução para os três métodos. O MOEA/D-ACO não foi proposto para os problemas da mochila nem do roteamento, o que fez necessária a adaptação da construção da solução em ambos os casos. O MOACS, por sua vez, já foi proposto para um problema parecido com o PRM, o problema do roteamento de veículos com janelas de tempo (BARÁN; SCHAERER, 2003). Mesmo assim, preferiu-se utilizar o método proposto para o MACO/NDS, pois, através de experimentos prévios, observou-se que nossa estratégia retornou os melhores resultados no PRM. Essas estratégias de construção da solução para ambos os problemas estão descritas no capítulo 5.

No MOEA/D-ACO, cada formiga possui uma solução corrente que interfere nas probabilidades ao construir uma nova solução. Para suportar essa característica, o processo de construção foi adaptado de modo que, ao calcular o feromônio no MOEA/D-ACO, soma-se um novo termo correspondente à presença ou ausência da aresta (ou item) na solução atual da formiga.

Para o PRM, todos os 8 algoritmos foram executados 30 vezes. Entretanto, devido à maior cardinalidade do espaço de busca do PMM, o custo computacional para realizar tais repetições se mostrou proibitivo em algumas situações. Por exemplo, a execução do SPEA2-SDE no cenário mais simples do problema, ou seja, 50 itens e 4 objetivos, levou 5,4 minutos para executar. Com 50 itens e 5 objetivos, o algoritmo precisou de 1 hora e 11 minutos. Já no próximo cenário (50 itens e 6 objetivos), o computador apresentou problemas de memória. Por essa razão, o SPEA2-SDE não foi avaliado para o problema da mochila. Com relação ao NSGA-III, o tempo de execução foi muito superior aos demais algoritmos analisados para esse problema, chegando a mais de uma hora em uma das situações. Por essa razão, nos cenários com 6 objetivos, tomou-se a média de apenas 5 execuções ao invés das 30, originalmente previstas. Além disso, para facilitar a visualização do tempo de execução nos gráficos, as informações referentes ao NSGA-III. Em vez disso, o tempo do NSGA-III, para cada um dos cenários do PMM, é informado na Tabela 13. Assim, no PMM foram avaliados apenas 7 algoritmos, sendo que os cenários com 6 objetivos do NSGA-III foram feitos a partir da média de 5 execuções ao invés de 30.

Tabela 13 – Tempos médios de execução para o NSGA-III nos cenários do PMM

Instância	**Obj.**	**Tempo de execução**
PMM 50 itens	4	1m 1s
	5	5m 34s
	6	1h 9m 5s
PMM 100 itens	4	4m 39s
	5	30m 34s
	6	35m 10s
PMM 200 itens	4	6m 21s
	5	26m 52s
	6	30m 49s

As Figuras 38 a 40 mostram o desempenho dos algoritmos nos 9 cenários do PMM, enquanto as Figuras 41 a 43 apresentam os gráficos com os resultados para os 9 cenários do PRM. O hiper-volume é representado pelas barras e o eixo do lado esquerdo, enquanto o tempo de execução é mostrado pela linha e o eixo do lado direito. No eixo do hiper-volume, as unidades K, M, P e E significam, respectivamente, 10^3, 10^6, 10^{15} e 10^{18}. Em ambos os problemas, os gráficos foram agrupados por formulação de objetivo.

Para 4 objetivos, o melhor valor de hiper-volume é obtido pelo MOEA/D-ACO, seguido do MACO/NDS e do MOACS. Além disso, os dois algoritmos baseados em ACO com

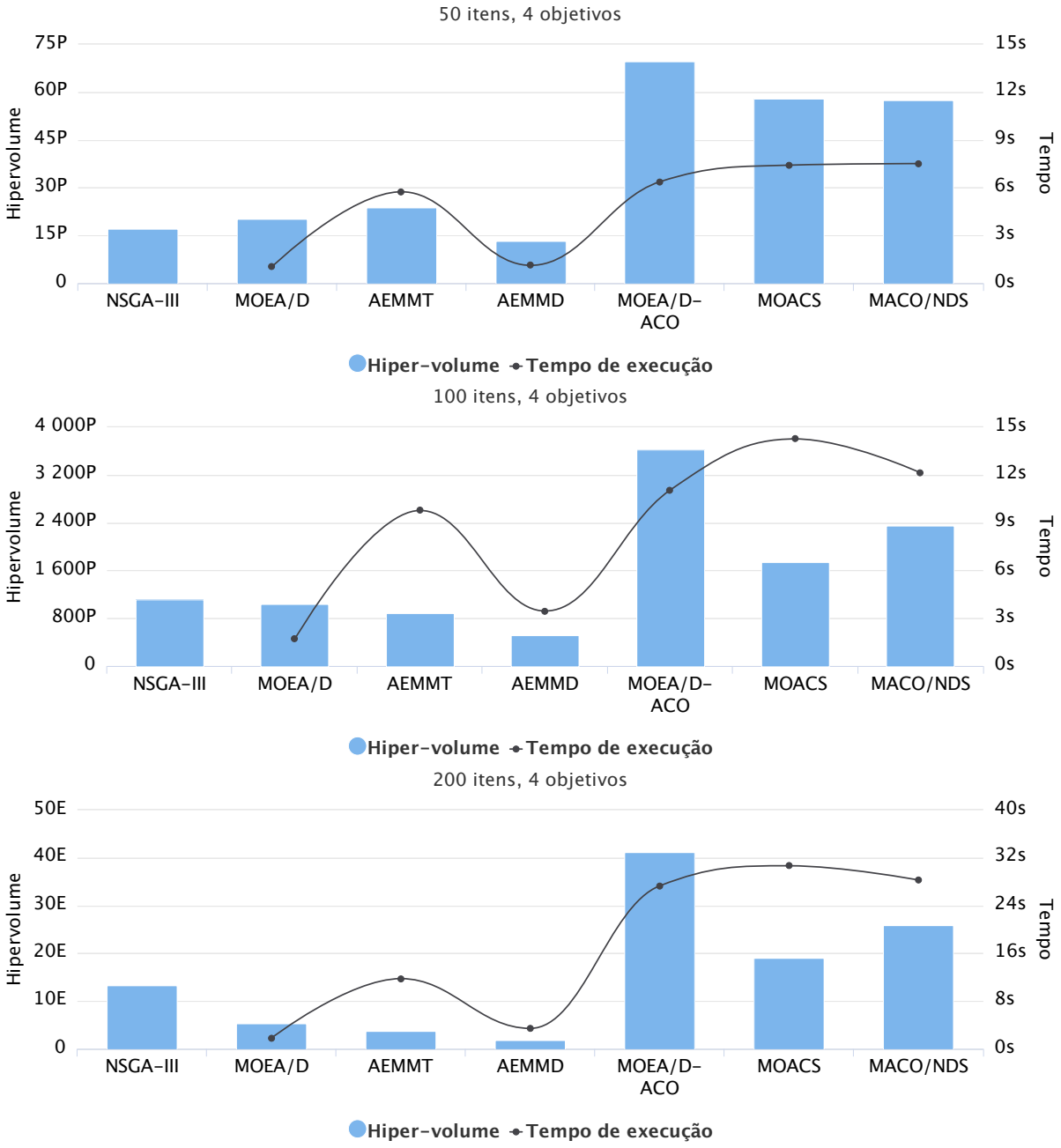

Figura 38 – Resultados dos algoritmos *many-objective* no PMM com 4 objetivos

hiper-volumes menores (MOACS e MACO/NDS) levam tempo similar ou maior que o MOEA/D-ACO. Portanto, o MOEA/D-ACO é o melhor ACO nesse cenário. Todos os algoritmos evolutivos obtêm valores de hiper-volume muito abaixo àqueles obtidos pelos ACOs, mas o MOEA/D e o AEMMD têm a vantagem de serem muito mais rápidos, sendo que o MOEA/D apresenta a melhor relação custo/benefício entre os algoritmos evolutivos.

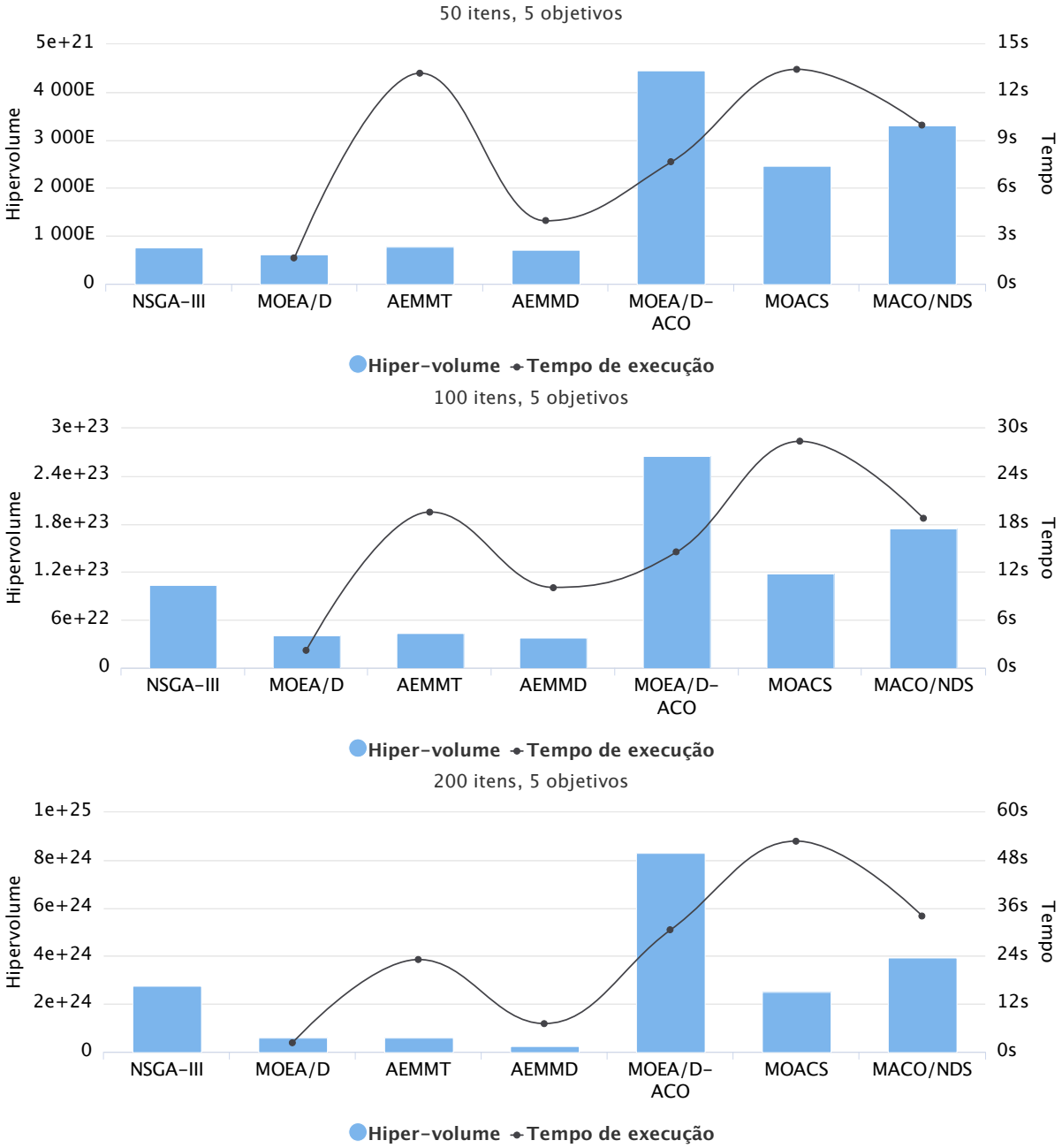

Figura 39 – Resultados dos algoritmos *many-objective* no PMM com 5 objetivos

Os resultados para 5 objetivos não mudam muito em relação à formulação anterior. O MOEA/D-ACO continua sendo o algoritmo que obtém o melhor valor de hiper-volume.

Dentre os três algoritmos baseados em colônia de formiga, o MOACS apresentou o pior resultado, tanto em tempo quanto em hiper-volume. O MACO/NDS executou em tempo similar ao melhor algoritmo (MOEA/D-ACO), mas obteve pior desempenho em relação ao hiper-volume. Novamente, os algoritmos evolutivos atingiram hiper-volumes inferiores aos ACOs, com exceção do NSGA-III que, apesar do alto custo em tempo de execução, conseguiu um valor de hiper-volume similar ao MOACS. Em termos de tempo de execução, o melhor algoritmo foi o MOEA/D, mas seu hiper-volume foi muito baixo em comparação aos métodos com os melhores resultados.

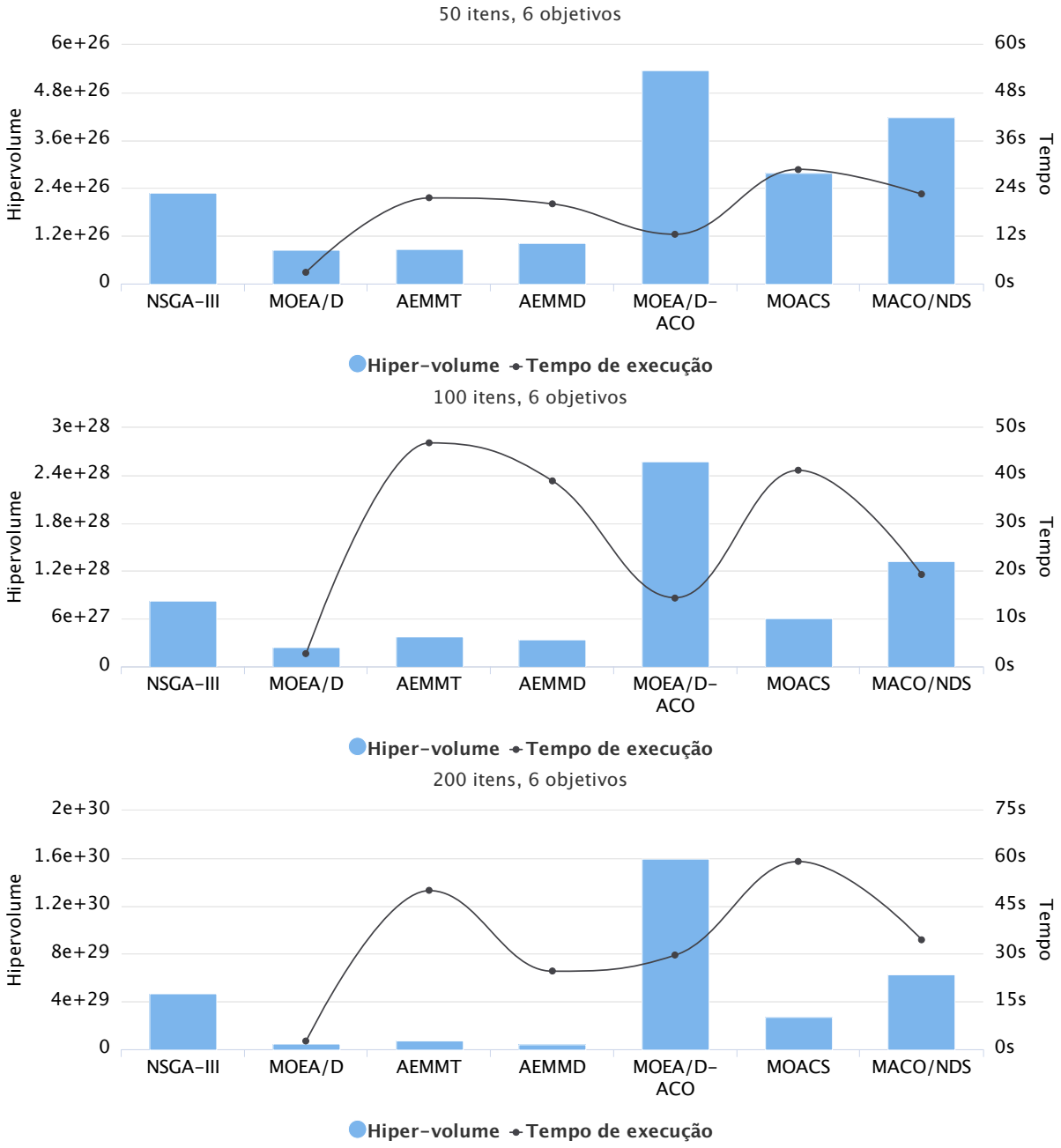

Figura 40 – Resultados dos algoritmos *many-objective* no PMM com 6 objetivos

Com 6 objetivos, o tempo médio necessário para executar os algoritmos AEMMT e AEMMD crescem consideravelmente, tornando-os ainda menos eficientes para o PMM. Nesses cenários, o MOEA/D-ACO possui, em geral, o segundo menor tempo de execução, perdendo apenas para o MOEA/D. O melhor hiper-volume, assim como nas formulações de objetivo anteriores, é dado pelo MOEA/D-ACO, o que faz dele o melhor algoritmo, pois consegue o melhor conjunto de soluções em um tempo satisfatório.

Em geral, no problema da mochila, todos os métodos evolutivos obtiveram hiper-volumes baixos quando comparados aos ACOs. Em termos de tempo médio de execução, o MOEA/D é, de longe, o algoritmo mais rápido dentre todos os métodos investigados. O NSGA-III, apesar de conseguir o melhor hiper-volume dentre os algoritmos evolutivos, é muito mais lento que qualquer outro método. Além disso, apresenta valores de hiper-volume menores que as abordagens baseadas em ACO. Portanto, não é uma boa opção em nenhum dos cenários analisados. Apesar de demandarem um maior tempo de execução, mas ainda se mantendo abaixo de 1 minuto, o MOEA/D-ACO e o MACO/NDS geram conjuntos de soluções de qualidade muito superior aos obtidos pelos algoritmos evolutivos. Dentre os três ACOs, o MOACS demorou mais a executar e obteve os piores resultados, enquanto o MOEA/D-ACO foi superior tanto em hiper-volume quanto em tempo, fazendo dele o melhor algoritmo para o problema da mochila multiobjetivo considerando-se os cenários analisados.

No PRM, o primeiro gráfico da Figura 41 mostra o desempenho dos algoritmos quando submetidos ao cenário mais simples (rede 3 com 4 objetivos). Nesse cenário, o AEMMD apresenta o pior hiper-volume (922), enquanto o MOACS e o MACO/NDS se destacam, sendo o MOACS o melhor, tanto em hiper-volume quanto em tempo de execução (2548 em 8,2 segundos). Os algoritmos evolutivos SPEA2-SDE e o AEMMT atingem um hiper-volume próximo ao do MACO/NDS, mas levam consideravelmente mais tempo para executar (2207 em 15,1 segundos e 2141 em 15,5 segundos, respectivamente). Neste cenário, o MOACS é claramente a melhor estratégia para o problema.

Na rede 4 com 4 objetivos (segundo gráfico da Figura 41), os algoritmos evolutivos apresentaram os hiper-volumes mais baixos, enquanto que as abordagens baseadas em colônias de formigas retornaram os maiores valores. O MOEA/D-ACO é o algoritmo mais rápido e obtém um bom hiper-volume (7502 em 4,3 segundos), enquanto o MOACS resultou no melhor hiper-volume, mas gastando um pouco mais de tempo (8330 em 7,1 segundos). O MACO/NDS foi o pior algoritmo baseado em formiga para esse cenário, apesar do desempenho muito superior aos AEMOs. Ele precisa de 9,5 segundos de execução para atingir um hiper-volume de 5953.

A rede 5 é a mais complexa utilizada em nossos experimentos. Considerando os resultados dos experimentos com 4 objetivos, apresentados no terceiro gráfico da Figura 41, o NSGA-III apresentou o pior resultado de hiper-volume (1036). O MOEA/D-ACO levou praticamente o mesmo tempo que o algoritmo MOEA/D (6,3 contra 6,5 segundos), mas

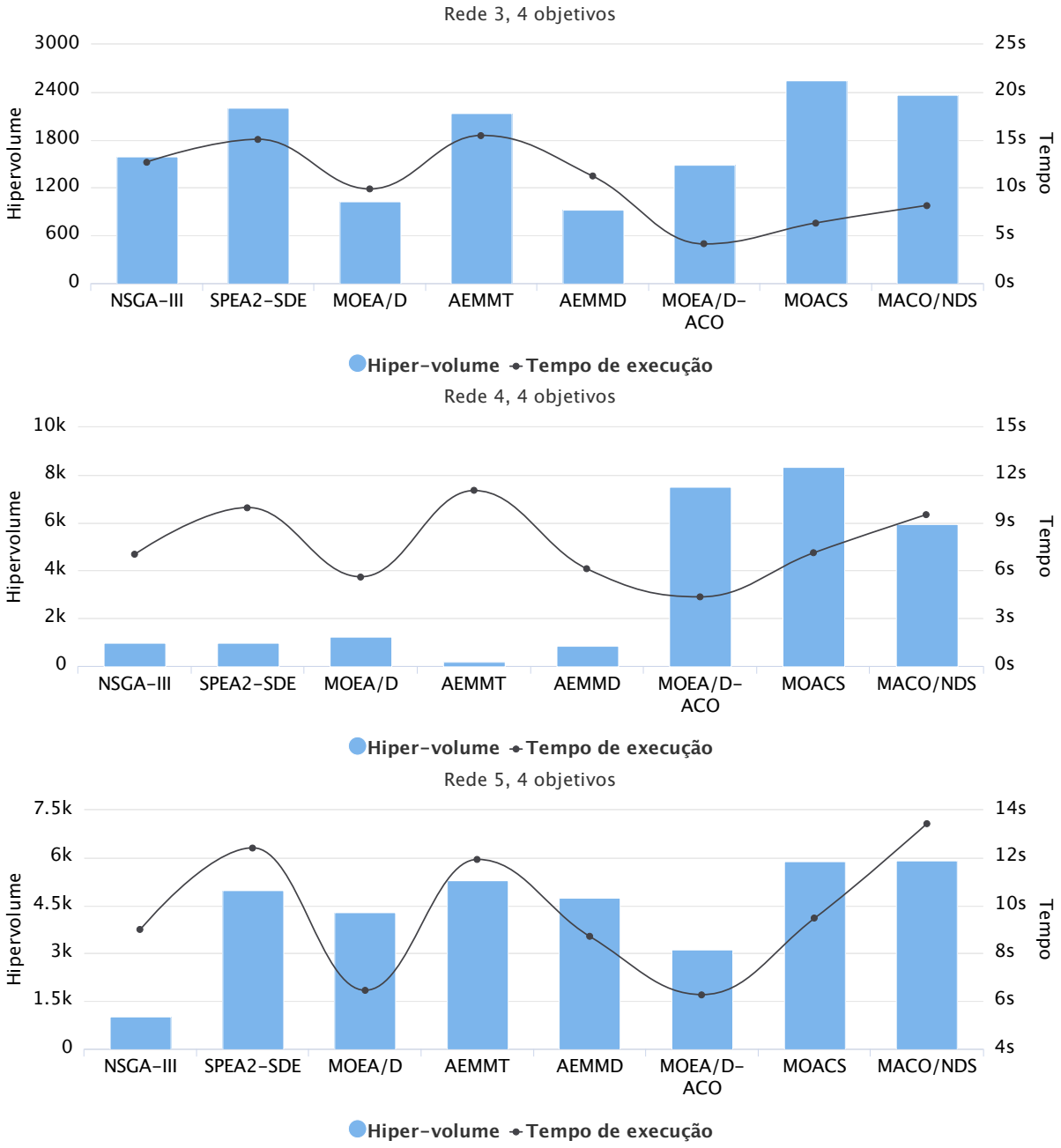

Figura 41 – Resultados dos algoritmos *many-objective* no PRM com 4 objetivos

obteve um hiper-volume consideravelmente maior (5904 contra 4292). Já os demais algoritmos evolutivos (SPEA2-SDE, AEMMD e AEMMT), atingiram um desempenho intermediário (4985, 4756 e 5305, respectivamente). Os melhores valores de hiper-volume foram obtidos pelo MACO/NDS e pelo MOACS. Entretanto, a melhor relação custo/benefício foi obtida pelo MOACS, que apresentou o segundo maior hiper-volume (5904) e um tempo de execução razoável (9,5 segundos). O MACO/NDS, apesar de alcançar o melhor valor de hiper-volume (5927), teve o pior desempenho em tempo de execução (13,4 segundos).

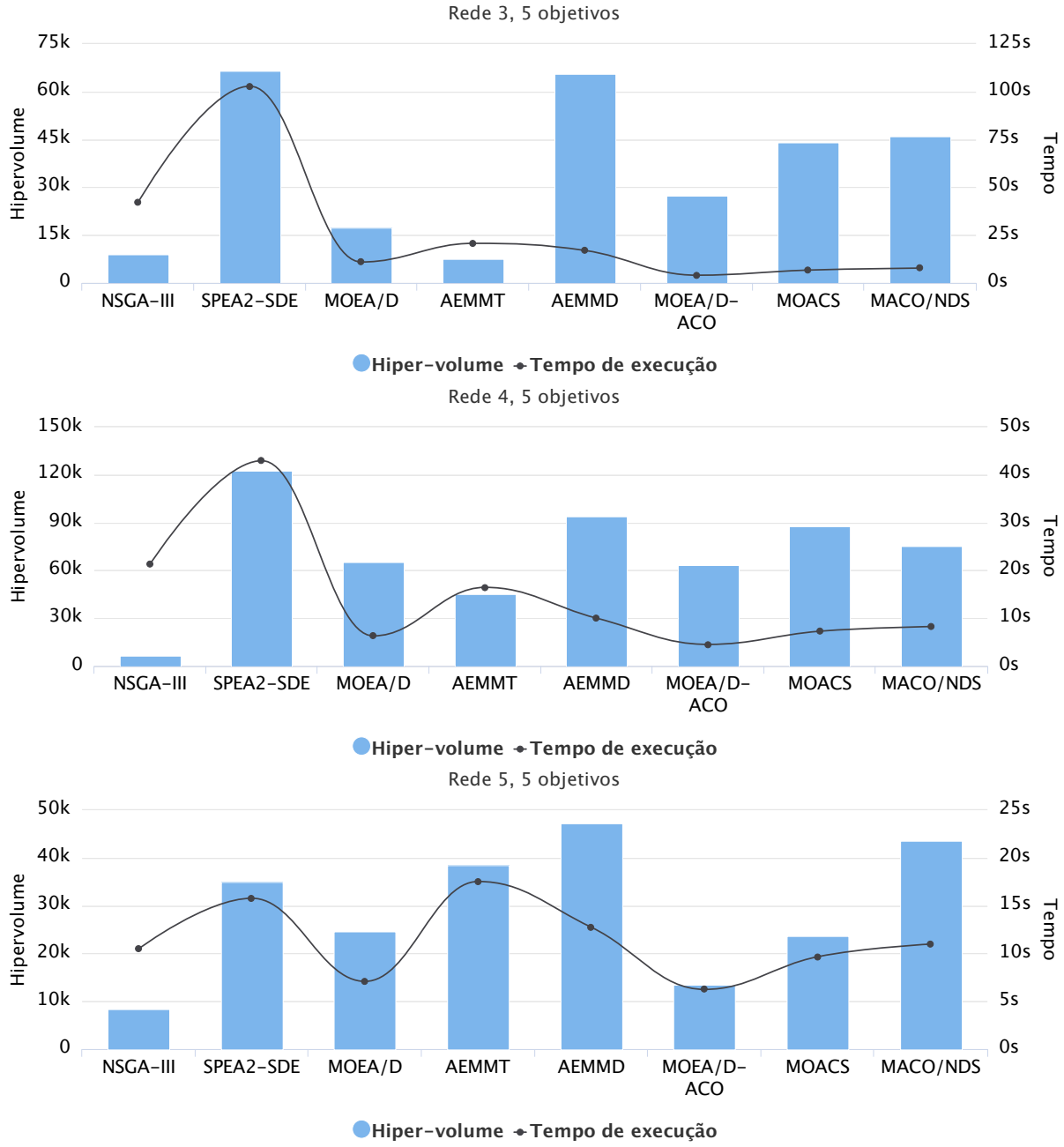

Figura 42 – Resultados dos algoritmos *many-objective* no PRM com 5 objetivos

Para a rede 3 com 5 objetivos (primeiro gráfico da Figura 42), o SPEA2 com a mo-

dificação SDE resultou no melhor hiper-volume entre todos os algoritmos (66.647). Em contrapartida, o custo do algoritmo, uma das principais desvantagens do SPEA2, também é um problema para o SPEA2-SDE. Nos experimentos, a execução desse algoritmo exigiu um tempo muito maior que as demais estratégias. Enquanto a execução do SPEA2-SDE demora, em média, 1,71 minutos, o NSGA-III (segundo pior algoritmo em tempo) demanda 42,1 segundos e as abordagens baseadas em ACO gastam, em média, 5,2 segundos. Em termos de hiper-volume, os piores resultados foram obtidos pelos algoritmos AEMMT, NSGA-III, MOEA/D e MOEA/D-ACO, nesta ordem. O AEMMD também apresentou bom valor de hiper-volume (65.747), bem próximo do SPEA2-SDE, só que com um tempo de execução bem mais baixo (17 segundos). Os algoritmos MOACS e MACO/NDS são mais rápidos (6,7 e 7,8 segundos, respectivamente) e atingiram bons valores de hiper-volume (44.303 e 46.033), embora inferiores ao AEMMD. Neste cenário, não está claro qual é o melhor algoritmo. Se o hiper-volume for priorizado em relação ao tempo de execução, o AEMMD é a melhor opção. Caso seja aceitável uma pequena redução no desempenho em favor de um algoritmo mais rápido, tanto o MOACS quanto o MACO/NDS são adequados. Ainda, se for analisada a relação hiper-volume/tempo, tem-se que o melhor algoritmo é o MOACS (6612 HV/s), seguido do MACO/NDS (5902 HV/s) e do AEMMD (3867 HV/s).

Considerando o segundo gráfico da Figura 42 (rede 4 com 5 objetivos), o SPEA2-SDE volta a mostrar os melhores valores para o hiper-volume (122.749). Entretanto, seu alto custo (42,9 segundos) em relação aos demais algoritmos, o torna menos atrativo em aplicações práticas. Os algoritmos AEMMD e MOACS são os que apresentam melhor desempenho geral. Enquanto o AEMMD consegue o segundo maior hiper-volume (94.005) e leva 10 segundos para executar, o MOACS atinge um hipe-rvolume um pouco mais baixo (87.681) em 7,3 segundos. Os algoritmos MACO/NDS e MOEA/D apresentam um desempenho similar e um pouco abaixo dos algoritmos anteriores. O MACO/NDS obteve um hiper-volume de 75.276 em 8,3 segundos de execução, enquanto que o MOEA/D atingiu 65.579 de hiper-volume em 6,3 segundos. Os piores valores de hiper-volume foram retornados pelo NSGA-III e AEMMT (6.554 e 45.310, respectivamente).

Na rede 5 com 5 objetivos (terceiro gráfico da Figura 42), os melhores hiper-volumes são dados pelo AEMMD e MACO/NDS. O AEMMD apresenta um conjunto de soluções de melhor qualidade (47.297 de hiper-volume), mas leva um pouco mais de tempo para calculá-las (12,7 segundos). As soluções encontradas pelo MACO/NDS são ligeiramente inferiores (43.666 de hiper-volume), mas o tempo necessário para obtê-las é um pouco menor (11 segundos).

No primeiro gráfico da Figura 43, que mostra os resultados para a rede 3 com 6 objetivos, o SPEA2-SDE repete o comportamento dos cenários anteriores, ou seja, consegue um resultado de hiper-volume muito bom, mas a um custo muito alto de tempo (4.768.243 em 6,53 minutos). O MOACS retorna um ótimo valor de hiper-volume, demandando um

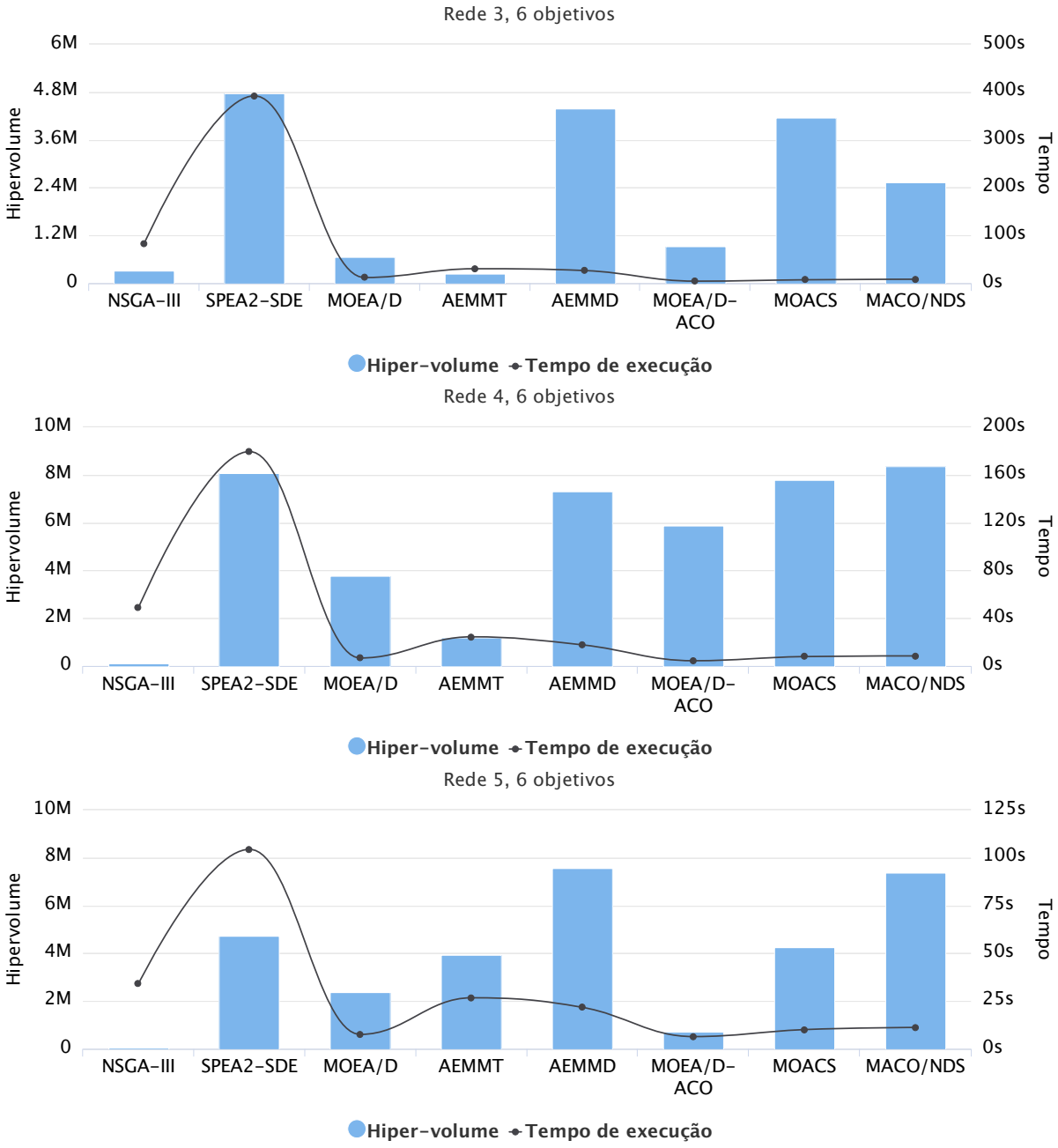

Figura 43 – Resultados dos algoritmos *many-objective* no PRM com 6 objetivos

tempo significativamente menor de execução (4.162.784 em 7,3 segundos). O AEMMD consegue hiper-volume pouco melhor que o MOACS (4.394.789), mas exigindo quase o quádruplo ($3,6\times$) do tempo de execução (26,6 segundos). O MACO/NDS obteve um valor intermediário para o hiper-volume (2.538.591), com um tempo de execução similar ao MOACS (8,4 segundos). Os demais algoritmos, apesar de terem um bom tempo de execução (exceto o NSGA-III), retornaram valores muito baixos para o hiper-volume.

Na rede 4 com 6 objetivos (segundo gráfico da Figura 43), o melhor hiper-volume é dado pelo MACO/NDS (8.684.287), seguido de perto pelo SPEA2-SDE (8.073.724), pelo MOACS (7.816.354) e pelo AEMMD (7.329.717). Devido ao seu tempo de execução (3 minutos), o SPEA-SDE possui uma péssima relação custo/benefício quando comparado às demais abordagens. O AEMMD possui um tempo de execução satisfatório, mas pior que as abordagens baseadas em ACO. Entre o MOACS e o MACO/NDS, o segundo consegue um conjunto de soluções de qualidade superior ao primeiro e leva apenas um segundo a mais para executar, sendo o algoritmo indicado para esse cenário.

O terceiro gráfico da Figura 43 apresenta os resultados obtidos para o cenário mais complexo do PRM nesta etapa, o qual é dado pela rede 5 com 6 objetivos. Nele, os piores resultados foram encontrados pelo NSGA-III (53.130) e MOEA/D-ACO (743.365) e MOEA/D (2.385.997). O SPEA2-SDE e o AEMMT obtiveram valores intermediários de hiper-volumes (4.731.148 e 3.954.286, respectivamente), mas , junto com o NSGA-III, apresentam os piores tempos de execução: 6,26 minutos para o SPEA2-SDE, 34,1 segundos para o NSGA-III e 26,8 segundos para o AEMMT. Novamente, repetindo o ocorrido no cenário anterior, o AEMMD e MACO/NDS foram os algoritmos que atingiram os melhores resultados de hiper-volume. O AEMMD com hiper-volume melhor e tempo pior (7.577.440 em 21,7 segundos), e o MACO/NDS com hiper-volume pior e tempo melhor (7.402.986 em 11,2 segundos).

De forma geral, percebe-se que o SPEA2-SDE encontra ótimos resultados, mas a um custo muito alto de tempo de execução, tornando-o inviável para aplicações como o PRM, onde é importante que se tenha uma resposta rápida sobre qual rota tomar. Dentre os métodos evolutivos, o melhor método foi o AEMMD que, na maioria dos casos, conseguiu um bom hiper-volume com um tempo de execução satisfatório. A grande vantagem das abordagens baseadas em ACO em relação aos algoritmos evolutivos investigados está no tempo de execução. Além disso, em 4 dos 9 cenários testados, os algoritmos baseados em formigas ultrapassam os AGs também em hiper-volume. Para o PRM, parece ser mais adequado utilizar uma abordagem baseada em ACO, pois assim é possível encontrar boas soluções a um custo computacional menor. Dentre os três ACOs investigados, na maioria da vezes, o MOACS apresentou melhores resultados que o MACO/NDS, tanto em tempo quanto em hiper-volume. Entretanto, é importante notar que, na rede 5, os valores de hiper-volume alcançados pelo MACO/NDS superam os do MOACS em todas as formulações de objetivos. Isso é um indício que o MACO/NDS pode se tornar a melhor

alternativa para redes mais complexas. Também é importante destacar que o MOACS implementado neste trabalho utiliza a abordagem do MACO/NDS para construir as soluções. Através de experimentos preliminares, foi possível observar que essa modificação melhorou o desempenho do algoritmo em relação à sua versão original (RIVEROS et al., 2016).

Considerando ambos os problemas, tanto o NSGA-III quanto o SPEA2-SDE não obtiveram desempenho satisfatório. O NSGA-III é muito caro computacionalmente para o PMM e não produz resultados muito bons de hiper-volume no PRM. O SPEA2-SDE leva muito tempo para executar tanto no PMM como no PRM, sendo inviável em ambos. O MOEA/D gera bons resultados em alguns cenários específicos do PRM, como a rede 5 com 4 e a rede 5 com 5 objetivos, mas, apesar de apresentar bons tempos de execução, no geral, não atinge um hiper-volume satisfatório. Os algoritmos AEMMT e AEMMD têm desempenhos muito ruins no PMM, mas, em contrapartida, atingem bons valores de hiper-volume no PRM, sendo que o AEMMD consegue o melhor resultado em vários cenários (rede 5 com 5 objetivos e rede 5 com 6 objetivos, por exemplo). A deficiência do AEMMT e do AEMMD está no tempo de execução, que é, em geral, superior aos algoritmos baseados em colônia de formigas. Os resultados mais interessantes foram obtidos pelos ACOs, que conseguiram bons valores de hiper-volumes a custos razoáveis de tempo. No PMM, o melhor algoritmo foi o MOEA/D-ACO, com valores de hiper-volume muito superiores aos AGs e tempo de execução, apesar de maior que os algoritmos evolutivos (excluindo o NSGA-III), inferior a 1 minuto. No PRM, os melhores algoritmos foram o AEMMD, o MOACS e o MACO/NDS. Em geral, o AEMMD obteve melhores valores de hiper-volume, mas exigindo um maior tempo de execução, enquanto os ACOs conseguiram valores muito bons de hiper-volume a um custo menor de tempo. Entre o MOACS e MACO/NDS, o primeiro atingiu melhor hiper-volume e tempo médio de execução na maioria dos cenários, mas o MACO/NDS conseguiu os melhores resultados nos experimentos mais complexos (redes 4 e 5 com 6 objetivos), o que é um indício de que possa funcionar melhor em instâncias mais complexas do problema. Também é interessante notar que o MACO/NDS foi o único algoritmo a conseguir bom desempenho tanto no problema da mochila quanto no problema do roteamento. De forma geral, considerando-se os 8 algoritmos investigados, o MACO/NDS foi o algoritmo de comportamento mais estável ao ser avaliado sobre os dois problemas discretos.

Capítulo 7

Conclusão

Este trabalho estendeu a pesquisa de (BUENO; OLIVEIRA, 2010) e (LAFETÁ et al., 2016), adicionando um novo problema de teste, o problema da mochila multiobjetivo (PMM), e 6 novos algoritmos *many-objective*, sendo três evolutivos (MOEA/D, SPEA2-SDE e NSGA-III) e três baseados em colônias de formigas (MOACS, MOEA/D-ACO e MACO/NDS). Dentre os ACO's incluídos no trabalho, um deles, o *Many-objective Ant Colony Optimization based on Non-Dominated Sets* (MACO/NDS), foi proposto pelo autor juntamente com uma nova estratégia de construção de solução para o problema do roteamento multicast (PRM).

O primeiro objetivo do trabalho, representado pela etapa 1 de experimentos (seção 6.2), foi incluir o novo problema PMM e comparar os algoritmos NSGA-II, SPEA2, MOEA/D, NSGA-III e AEMMT utilizando as métricas de desempenho paramétricas (baseadas no conhecimento prévio da fronteira de Pareto): taxa de erro (ER), *generational distance* (GD) e *pareto subset* (PS). Todos os métodos foram executados para ambos os problemas (PRM e PMM) em diversos cenários, variando a complexidade das instâncias e as quantidades de objetivo. O estudo permitiu verificar o comportamento de cada estratégia multiobjetivo em relação tanto à complexidade da entrada quanto ao número de funções de otimização. De maneira geral, confirmou-se a expectativa de se encontrar melhores resultados para os algoritmos clássicos NSGA-II e SPEA2 nos cenários com 2 e 3 objetivos e a decadência nos seus desempenhos a partir de 4 objetivos. Com 4 ou mais objetivos, os algoritmos MOEA/D e AEMMT obtiveram resultados bem melhores que os demais. O NSGA-III, por sua vez, se mostrou o método mais estável: não é o pior nem o melhor na maioria das situações. Para os problemas *many-objective*, foco deste trabalho, o AEMMT foi o algoritmo que se mostrou mais interessante.

Os AG's *multi* e *many-objective* estão massivamente presentes na literatura e já foram explorados de diversas maneiras possíveis, por outro lado, os algoritmos baseados em colônias de formigas, apesar do potencial, não são utilizados com a mesma frequência. Os ACO's, por utilizarem uma estratégia de busca diferente, podem encontrar soluções até então inexploradas. Por esse motivo, esta pesquisa estudou vários ACO's na literatura,

implementou dois deles e propôs um novo algoritmo, o MACO/NDS. Um dos principais aspectos de um ACO é a construção da solução, assim como num AG um dos principais fatores é o processo de *crossover*. Dessa forma, este trabalho também traz uma revisão dos métodos para se construir uma solução no PMM (seção 5.1) e uma nova estratégia para se criar as árvores do PRM (seção 5.2).

Considerando que os problemas com 2 e 3 objetivos são bem resolvidos pelos algoritmos multiobjetivos tradicionais (NSGA-II e SPEA2), mas que estes não são eficientes para lidar com problemas a partir de 4 objetivos, a partir da etapa 3 de experimentos, o foco desta pesquisa foi nos problemas e algoritmos *many-objective*. Assim, nessa etapa foi analisado pela primeira vez o comportamento do MACO/NDS contra os algoritmos genéticos NSGA-III, MOEA/D, AEMMT e AEMMD nos três cenários mais simples de cada problema. A análise foi feita através das métricas baseadas em Pareto (ER, GDp e PS) e do tempo de execução. Em geral, no PRM, o AEMMD e o método proposto, MACO/NDS, obtiveram os melhores resultados. O AEMMD consegue soluções um pouco melhores, mas ao mesmo tempo leva até quatro vezes mais tempo para executar. O MACO/NDS, apesar de não conseguir o melhor conjunto de soluções entre os dois métodos, chega em resultados bem próximos e é um algoritmo muito rápido, característica essencial para um problema de roteamento em redes. Com relação ao PMM, devido ao tamanho muito grande das fronteiras de Pareto, o MACO/NDS não exibe um comportamento tão bom em relação ao tempo, mas é consideravelmente o melhor algoritmo em relação ao PS, atingindo valores muito bons de ER e GDp.

Tendo comprovado a eficácia do MACO/NDS em relação aos algoritmos genéticos utilizando as instâncias mais simples de cada problema, na última etapa de experimentos (seção 6.5), foram avaliadas instâncias mais complexas do PMM e do PRM. Além disso, para que fosse possível aferir a qualidade do MACO/NDS em relação a outras estratégias baseadas em colônias de formigas, incluiu-se os algoritmos MOEA/D-ACO e MOACS. Um novo algoritmo genético também foi incluído nas comparações, o SPEA2-SDE, uma variação do SPEA2 com a adição do operador SDE, tornando esse AEMO mais eficiente ao lidar com problemas que envolvam 4 ou mais objetivos. Como forma de avaliação dos resultados, utilizou-se o hiper-volume devido à dificuldade de se extrair um Pareto aproximado para as redes 4 e 5 e os problemas da mochila com 100 e 200 itens. No PRM, foi possível verificar que os ACO's levam normalmente menos tempo para executar que os AG's. Dentre os algoritmos genéticos, o único método com bons resultados foi o AEMMD (exclusivamente no PRM). O SPEA2-SDE consegue ótimas soluções, mas o alto custo do algoritmo em espaços de alta dimensionalidade faz com que ele seja uma opção inviável para o PRM. Em geral, os ACO's apresentaram uma melhor relação de custo/benefício no PRM e dentre eles, o MOACS obteve melhor hiper-volume e tempo na maioria dos casos, enquanto o MACO/NDS mostrou uma tendência de obter o melhor conjunto de soluções à medida que cresce a complexidade da entrada. Quanto ao PMM, os algoritmos

se comportaram de maneira mais estável ao variar os cenários de testes. Para todos os casos do PMM, o MOEA/D-ACO apresentou o melhor custo benefício entre hiper-volume e tempo, sendo o MOEA/D uma alternativa a ser considerada quando é necessário um tempo de execução muito curto.

Desta forma, as principais contribuições deste trabalho para o campo de busca e otimização multiobjetivo foram:

- Comparação entre AG's: foram comparados 7 algoritmos genéticos multiobjetivos em dois problemas discretos diferentes, o que oferece uma grande gama de dados para que possam ser tomadas decisões a respeito de qual algoritmo utilizar em determinadas situações.

- Proposição de um novo ACO e um modelo de construção de soluções para o PRM: este trabalho propôs uma nova ideia para se implementar ACOs multiobjetivos, apresentando um algoritmo eficaz que lida bem com problemas de muitos objetivos em um campo relativamente pouco explorado (otimização multiobjetivo por colônias de formigas). O novo algoritmo de construção da solução apresentado para o PRM, não só é parte da proposição do MACO/NDS, como também pode ser utilizado em outros *frameworks* para ACO já estabelecidos, melhorando o desempenho desses algoritmos. Isso foi observado, por exemplo, ao se avaliar o método MOACS utilizando o novo modelo de construção e o modelo original proposto em (RIVEROS et al., 2016) para o PRM, onde o novo modelo foi superior.

- Comparação entre AGs e ACOs: outra contribuição desta pesquisa foram as comparações de desempenho entre os algoritmos genéticos e as algoritmos baseados em colônias de formigas, considerando diferentes cenários para cada problema (PRM e PMM) e utilizando métricas paramétricas (erro, GD e PS) e não paramétricas (hiper-volume e tempo de execução). Por meio das análises foi possível identificar alguns pontos fortes e fragilidades de cada metaheurística.

7.1 Trabalhos Futuros

Algumas novas linhas de investigação foram vislumbradas no decorrer deste trabalho e seria muito interessante abordá-las no futuro. Propomos a investigação dos seguintes tópicos em pesquisas futuras:

- Em problemas onde se conhece a fronteira de Pareto, a métrica de avaliação *inverse generation distance* (IGD) vem se tornando uma importante referência de avaliação nos trabalhos mais recentes sobre otimização multiobjetivo, até mesmo substituindo em alguns casos as métricas mais tradicionais ER, GD, GDp e PS. Assim, um

aspecto interessante na continuidade desse trabalho seria incluir essa métrica de avaliação nas análises futuras.

- ❏ Foi observado que o tempo de execução do MACO/NDS no PMM é muito alto, diferentemente do observado para o PRM. Tal comportamento se deve ao fato de o número de soluções não dominadas nas instâncias do PMM aqui investigadas ser muito elevado, enquanto que no PRM, a cardinalidade do Pareto não é tão significativa. Essa cardinalidade elevada no PMM, faz com que o algoritmo despenda um tempo significativo na manipulação das soluções não dominadas já encontradas. A investigação de alguma estratégia de limitação no tamanho do conjunto corrente de soluções não dominadas, que elimine soluções de forma eficiente é importante para reduzir o tempo de execução do algoritmo sem afetar muito a qualidade das soluções em relação ao Pareto real. Nessa investigação, estratégias usuais de métodos evolutivos podem ser avaliadas (ex: truncamento, clusterização, pontos de referência, etc).

- ❏ Investigar outros testes de hipótese além do Z-teste como forma de avaliação dos resultados obtidos por cada algoritmo, principalmente testes não paramétricos que permitem a comparação de mais de duas abordagens.

- ❏ O MACO/NDS e o MOACS executam muito mais rápido que o AEMMD em alguns cenários do PRM, mas produzem um conjunto de soluções com qualidade inferior. Em nossos experimentos, o número de avaliações de novos indivíduos foi utilizado como parâmetro para determinar uma comparação equilibrada entre os indivíduos. Por outro lado, os protocolos de internet exigem que na prática o roteamento seja realizado em um tempo bastante restrito (na ordem de grandeza de milisegundos). Seria interessante investigar o desempenho dos três algoritmos quando fossem executados sujeitos a uma restrição de tempo, independentemente do número de avaliações efetuadas de novos indivíduos.

- ❏ Nos cenários mais complexos do PRM, o MACO/NDS apresentou resultados significativamente melhores que o MOACS. Uma nova investigação utilizando redes ainda mais complexas (maior número de nós e arestas) e um número maior de objetivos seria interessante para verificar se essa tendência continuaria nos problemas mais difíceis de resolver, bem como tentar identificar se ela tem maior relação com a complexidade da rede ou ao número maior de objetivos.

- ❏ Em relação ao problema de roteamento multicast, foram investigadas algumas combinações de objetivos relacionados ao roteamento com qualidade de serviço, apresentados na seção 4.2.1. Essas combinações (2 a 6 objetivos) foram definidas em função das formulações utilizadas em trabalhos anteriores - Lafetá et al. (2016), Bueno e

Oliveira (2010). Como continuidade dessa investigação, formulações adicionais podem ser avaliadas, realizando-se uma análise de correlação entre os objetivos para a definição das novas combinações.

- O problema da mochila multiobjetivo investigado inclui múltiplos objetivos, mas apenas uma restrição. Essa característica faz com que o crescimento do número de soluções não-dominadas seja muito significativo quando o número de itens a serem alocados é aumentado. E esse crescimento é ainda intensificado quando um número maior de objetivos é utilizado. Por exemplo, pela Tabela 8, observa-se que o Pareto aproximado no PMM com 6 objetivos subiu de 6491 para 55471, quando o número de itens subiu de 40 para 50. Essa característica tornou a aproximação do Pareto inviável para 100 e 200 de objetos. Por outro lado, existem outras variações do PMM que trabalham com múltiplos objetivos e múltiplas restrições (ISHIBUCHI; AKEDO; NOJIMA, 2015; ALAYA; SOLNON; GHEDIRA, 2007). Acredita-se que nessa variação do PMM, o crescimento do número de soluções não dominadas com o aumento do número de itens e/ou objetivos seja mais controlada. Seria interessante uma nova implementação utilizando-se essa variação do PMM, para verificar se o comportamento dos algoritmos mudaria muito. Além disso, com uma menor quantidade de elementos no Pareto, onde os algoritmos levariam muito menos tempo para executar, seria interessante avaliar se haveria alguma modificação na análise comparativa dos tempos de execução entre os métodos.

- O MACO/NDS foi avaliado em apenas dois problemas *many-objective* nessa dissertação: PMM e PRM. Para validá-lo como um *framework* mais geral, seria relevante a sua aplicação em outros problemas de otimização discreta com muitos objetivos. Como problemas em potencial para essa investigação futura, podemos destacar as funções discretas apresentadas em (DEB et al., 2002b) e (HUBAND et al., 2006), o problema do caixeiro viajante (RIVEROS et al., 2016) e o problema de escalonamento *job shop* (MASOOD et al., 2016), que também são investigados na literatura de otimização discreta com muitos objetivos.

- Avaliar a influência dos feromônios e da heurística nos resultados a fim de identificar as características dos problemas que se beneficiam do uso de um ou de outro. Estudar possíveis dinamizações dos parâmetros α e β para que possam ajustar a influência dos dois componentes (feromônio e heurística) da melhor forma possível no decorrer do algoritmo.

- Embora existam adaptações de *frameworks* ACOs para domínios contínuos, entendemos que a maior aplicabilidade do MACO/NDS seja em problemas de otimização combinatória, pela maior adaptabilidade da construção iterativa de soluções a problemas discretos. Entretanto, existe um outro método baseado em inteligência co-

letiva mais adequado a problemas contínuos que também é bastante investigado na literatura de otimização multiobjetivo: o PSO. Assim, um possível desdobramento da nossa pesquisa é a adaptação das ideias dos algoritmos baseados na decomposição em subproblemas multiobjetivos (MACO/NDS e AEMMD) para um *framework* baseado em PSO e sua posterior aplicação a funções e problemas contínuos, além de compará-lo a outras abordagens similares (FREIRE; OLIVEIRA; PIRES, 2017).

7.2 Contribuições em Produção Bibliográfica

A produção bibliográfica originada neste trabalho compreende os seguintes artigos:

1. "*A Comparative Analysis of MOEAs Considering Two Discrete Optimization Problems*" por Tiago Peres França, Thiago Fialho de Q. Lafetá, Luiz G. A. Martins e Gina M. B. de Oliveira. Publicado e apresentado no evento *Brazilian Conference on Intelligent Systems* (BRACIS) de 2017. Este artigo compreende os experimentos realizados na etapa 1 (seção 6.2). (FRANÇA et al., 2017).

2. "*MACO/NDS: Many-objective Ant Colony Optimization based on Decomposed Pheromone*" por Tiago Peres França, Luiz G. A. Martins e Gina M. B. de Oliveira. Publicado no evento *Congress on Evolutionary Computation* (CEC) de 2018. Este artigo propõe o algoritmo MACO/NDS (capítulo 5) e compreende os experimentos realizados na etapa 3 (seção 6.4). (FRANçA; MARTINS; OLIVEIRA, 2018).

Referências

AGUIRRE, H.; TANAKA, K. Many-objective optimization by space partitioning and adaptive ε-ranking on mnk-landscapes. In: SPRINGER. **International Conference on Evolutionary Multi-Criterion Optimization**. 2009. p. 407–422. Disponível em: <https://doi.org/10.1007/978-3-642-01020-0_33>.

ALAYA, I.; SOLNON, C.; GHÉDIRA, K. Ant algorithm for the multi-dimensional knapsack problem. In: **International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004**. [s.n.], 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.3439>.

ALAYA, I.; SOLNON, C.; GHEDIRA, K. Ant colony optimization for multi-objective optimization problems. In: **19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)**. [s.n.], 2007. v. 1, p. 450–457. ISSN 1082-3409. Disponível em: <https://doi.org/10.1109/ICTAI.2007.108>.

BADER, J.; ZITZLER, E. Hype: An algorithm for fast hypervolume-based many-objective optimization. **Trans. Evol. Comput.**, v. 19(1), p. 45–76, 2011. Disponível em: <https://doi.org/10.1162/EVCO_a_00009>.

BARÁN, B.; SCHAERER, M. A multiobjective ant colony system for vehicle routing problem with time windows. In: **Applied informatics**. [s.n.], 2003. p. 97–102. Disponível em: <https://www.researchgate.net/publication/221277596_A_Multiobjective_Ant_Colony_System_for_Vehicle_Routing_Problem_with_Time_Windows>.

BAZGAN, C.; HUGOT, H.; VANDERPOOTEN, D. Solving efficiently the 0-1 multi-objective knapsack problem. **Comput. Oper. Res.**, Elsevier Science Ltd., Oxford, UK, UK, v. 36, n. 1, p. 260–279, jan. 2009. ISSN 0305-0548. Disponível em: <https://doi.org/10.1016/j.cor.2007.09.009>.

BEUME, N.; NAUJOKS, B.; EMMERICH, M. Sms-emoa: Multiobjective selection based on dominated hypervolume. **European Journal of Operational Research**, v. 181, n. 3, p. 1653 – 1669, 2007. ISSN 0377-2217. Disponível em: <https://doi.org/10.1016/j.ejor.2006.08.008>.

BRASIL, C. R. S.; DELBEM, A. C. B.; SILVA, F. L. B. da. Multiobjective evolutionary algorithm with many tables for purely ab initio protein structure prediction. **Comput. Chem.**, v. 34(20), p. 1719–1734, 2013. Disponível em: <https://doi.org/10.1002/jcc.23315>.

BUENO, M. L. d. P. **Heurísticas e algoritmos evolutivos para formulações mono e multiobjetivo do problema do roteamento multicast**. Dissertação (Mestrado) — Programa de Pós-graduação em Ciência da Computação, Universidade Federal de Uberlândia, 2010. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/12501/1/Diss%20Marcos.pdf>.

BUENO, M. L. P.; OLIVEIRA, G. M. B. Multicast flow routing: Evaluation of heuristics and multiobjective evolutionary algorithms. In: **Congress Evol. Comput.** [s.n.], 2010. p. 1–8. Disponível em: <https://doi.org/10.1109/CEC.2010.5585942>.

CHANGDAR, C.; MAHAPATRA, G.; PAL, R. K. An ant colony optimization approach for binary knapsack problem under fuzziness. **Applied Mathematics and Computation**, Elsevier, v. 223, p. 243–253, 2013. Disponível em: <https://doi.org/10.1016/j.amc.2013.07.077>.

CHU, P. C.; BEASLEY, J. E. A genetic algorithm for the multidimensional knapsack problem. **J. heuristics**, v. 4(1), p. 63–86, 1998. Disponível em: <https://doi.org/10.1023/A:1009642405419>.

CORMEN, T. H. et al. Greedy algorithms. In: FAGERBERG, J.; MOWERY, D. C.; NELSON, R. R. (Ed.). **Introduction to Algorithms, Third Edition**. MIT: MIT Press, 2009. cap. 16.

CRICHIGNO, J.; BARÁN, B. Multiobjective multicast routing algorithm. In: SPRINGER. **International Conference on Telecommunications**. 2004. p. 1029–1034. Disponível em: <https://doi.org/10.1007/978-3-540-27824-5_134>.

DARWIN, C. On the origin of species by means of natural selection. **Murray, London**, 1859.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. **Trans. Evol. Comput.**, v. 18(4), p. 577–601, 2014. Disponível em: <https://doi.org/10.1109/TEVC.2013.2281535>.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **Trans. Evol. Comput.**, v. 6(2), p. 182–197, 2002. Disponível em: <https://doi.org/10.1109/4235.996017>.

_____. Scalable multi-objective optimization test problems. In: **Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on**. [s.n.], 2002. v. 1, p. 825–830. Disponível em: <https://doi.org/10.1109/CEC.2002.1007032>.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, v. 1, p. 269–271, 1959. Disponível em: <https://doi.org/10.1007/BF01386390>.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. **IEEE Trans. on Systems, Man, and Cybernetics, Part B**, v. 26, n. 1, p. 29–41, 1996. Disponível em: <https://doi.org/10.1109/3477.484436>.

FIDANOVA, S. Aco algorithm for mkp using various heuristic information. In: DIMOV, I. et al. (Ed.). **Numerical Methods and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 438–444. ISBN 978-3-540-36487-0. Disponível em: <https://doi.org/10.1007/3-540-36487-0_49>.

FRANçA, T. P.; MARTINS, L. G. A.; OLIVEIRA, G. M. B. Maco/nds: Many-objective ant colony optimization based on non-dominated sets. In: **2018 IEEE Congress on Evolutionary Computation (CEC)**. [s.n.], 2018. p. 1–8. Disponível em: <https://doi.org/10.1109/CEC.2018.8477958>.

FRANÇA, T. P. et al. A comparative analysis of moeas considering two discrete optimization problems. In: **2017 Brazilian Conference on Intelligent Systems (BRACIS)**. [s.n.], 2017. p. 402–407. Disponível em: <https://doi.org/10.1109/BRACIS.2017.76>.

FREIRE, H.; OLIVEIRA, P. B. M.; PIRES, E. J. S. From single to many-objective pid controller design using particle swarm optimization. **International Journal of Control, Automation and Systems**, v. 15, n. 2, p. 918–932, Apr 2017. ISSN 2005-4092. Disponível em: <https://doi.org/10.1007/s12555-015-0271-0>.

GOLDBERG, D. E. Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman, 1989. Disponível em: <https://dl.acm.org/citation.cfm?id=534133>.

HRISTAKEVA, M.; SHRESTHA, D. Solving the 0-1 knapsack problem with genetic algorithms. In: **Midwest Instruction and Computing Symposium**. [s.n.], 2004. Disponível em: <http://www.sc.ehu.es/ccwbayes/docencia/kzmm/files/AG-knapsack.pdf>.

_____. Different approaches to solve the 0/1 knapsack problem. In: **The Midwest Instruction and Computing Symposium**. [s.n.], 2005. Disponível em: <https://www.researchgate.net/publication/228894571_Different_Approaches_to_Solve_the_01_Knapsack_Problem>.

HUBAND, S. et al. A review of multiobjective test problems and a scalable test problem toolkit. **IEEE Transactions on Evolutionary Computation**, v. 10, n. 5, p. 477–506, Oct 2006. ISSN 1089-778X. Disponível em: <https://doi.org/10.1109/TEVC.2005.861417>.

IREDI, S.; MERKLE, D.; MIDDENDORF, M. Bi-criterion optimization with multi colony ant algorithms. In: ZITZLER, E. et al. (Ed.). **Evolutionary Multi-Criterion Optimization**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 359–372. Disponível em: <https://doi.org/10.1007/3-540-44719-9_25>.

ISHIBUCHI, H.; AKEDO, N.; NOJIMA, Y. A study on the specification of a scalarizing function in moea/d for many-objective knapsack problems. In: NICOSIA, G.; PARDALOS, P. (Ed.). **Learning and Intelligent Optimization**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 231–246. ISBN 978-3-642-44973-4. Disponível em: <https://doi.org/10.1109/TEVC.2014.2315442>.

_____. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. **Trans. Evol. Comput.**, v. 19(2), p. 264–283, 2015. Disponível em: <https://doi.org/10.1109/TEVC.2014.2315442>.

ISHIBUCHI, H. et al. Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems. In: **2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)**. [s.n.], 2014. p. 170–177. Disponível em: <https://doi.org/10.1109/MCDM.2014.7007204>.

KANN, V. **On the approximability of NP-complete optimization problems**. Tese (Doutorado) — Royal Institute of Technology Stockholm, 1992. Disponível em: <https://www.researchgate.net/publication/267553084_On_the_Approximability_of_NP-complete_Optimization_Problems>.

KE, L. et al. An ant colony optimization approach for the multidimensional knapsack problem. v. 16, p. 65–83, 2010. Disponível em: <https://doi.org/10.1007/s10732-008-9087-x>.

KE, L.; ZHANG, Q.; BATTITI, R. Moea/d-aco: A multiobjective evolutionary algorithm using decomposition and antcolony. **IEEE Transactions on Cybernetics**, v. 43, n. 6, p. 1845–1859, Dec 2013. ISSN 2168-2267. Disponível em: <https://doi.org/10.1109/TSMCB.2012.2231860>.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Proceedings of ICNN'95 - International Conference on Neural Networks**. [s.n.], 1995. v. 4, p. 1942–1948 vol.4. Disponível em: <https://doi.org/10.1109/ICNN.1995.488968>.

KONG, M.; TIAN, P. Introducing a binary ant colony optimization. In: DORIGO, M. et al. (Ed.). **Ant Colony Optimization and Swarm Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 444–451. ISBN 978-3-540-38483-0. Disponível em: <https://doi.org/10.1007/11839088_44>.

KONG, M.; TIAN, P.; KAO, Y. A new ant colony optimization algorithm for the multidimensional knapsack problem. **Computers & Operations Research**, Elsevier, v. 35, n. 8, p. 2672–2683, 2008. Disponível em: <https://doi.org/10.1016/j.cor.2006.12.029>.

LAFETÁ, T. F. d. Q. **Algoritmos evolutivos many objectives aplicados ao problema de roteamento Multicast com qualidade de serviço**. Dissertação (Mestrado) — Programa de Pós-graduação em Ciência da Computação, Universidade Federal de Uberlândia, 2016. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/18093/1/AlgoritmosEvolutivosMany.pdf>.

LAFETÁ, T. F. d. Q. et al. Many-objective evolutionary algorithms for multicast routing with quality of service problem. In: **Braz. Conf. on Intelligent Systems**. [s.n.], 2016. p. 187–192. Disponível em: <https://doi.org/10.1109/BRACIS.2016.043>.

LAFETÁ, T. F. de Q. et al. Meands: A many-objective evolutionary algorithm based on non-dominated decomposed sets applied to multicast routing. **Appl. Soft Comput.**, v. 62, p. 851–866, 2018. Disponível em: <https://doi.org/10.1016/j.asoc.2017.09.017>.

LEGUIZAMON, G.; MICHALEWICZ, Z. A new version of ant system for subset problems. In: **Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)**. [s.n.], 1999. v. 2, p. 1464 Vol. 2. Disponível em: <{https://doi.org/10.1109/CEC.1999.782655},ISSN={},mon>.

LI, M.; YANG, S.; LIU, X. Shift-based density estimation for pareto-based algorithms in many-objective optimization. **IEEE Transactions on Evolutionary Computation**, v. 18, n. 3, p. 348–365, June 2014. ISSN 1089-778X. Disponível em: <https://doi.org/10.1109/TEVC.2013.2262178>.

LOPEZ-IBANEZ, M.; STUTZLE, T. The automatic design of multiobjective ant colony optimization algorithms. **IEEE Transactions on Evolutionary Computation**, v. 16, n. 6, p. 861–875, Dec 2012. ISSN 1089-778X. Disponível em: <https://doi.org/10.1109/TEVC.2011.2182651>.

LUST, T.; TEGHEM, J. The multiobjective traveling salesman problem: A survey and a new approach. In: _____. **Advances in Multi-Objective Nature Inspired Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 119–141. ISBN 978-3-642-11218-8. Disponível em: <https://doi.org/10.1007/978-3-642-11218-8_6>.

MASOOD, A. et al. Many-objective genetic programming for job-shop scheduling. In: **2016 IEEE Congress on Evolutionary Computation (CEC)**. [s.n.], 2016. p. 209–216. Disponível em: <https://doi.org/10.1109/CEC.2016.7743797>.

MURATA, T.; ISHIBUCHI, H. Moga: Multi-objective genetic algorithms. In: IEEE. **Evolutionary Computation, 1995., IEEE International Conference on**. 1995. v. 1, p. 289. Disponível em: <https://doi.org/10.1109/ICEC.1995.489161>.

NÉIA, S. S. et al. Roteamento de veículos utilizando otimização por colônia de formigas e algoritmo genético. **Meta-heurísticas em pesquisa operacional. cap**, v. 14, p. 219–236, 2013. Disponível em: <https://doi.org/10.7436/2013.mhpo.14>.

OMBUKI, B.; ROSS, B. J.; HANSHAR, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. **Applied Intelligence**, v. 24, n. 1, p. 17–30, Feb 2006. ISSN 1573-7497. Disponível em: <https://doi.org/10.1007/s10489-006-6926-z>.

PINTO, D.; BARAN, B. Solving multiobjective multicast routing problem with a new ant colony optimization approach. In: **Int. Latin American Conf. on Networking**. [s.n.], 2005. p. 11–19. Disponível em: <https://doi.org/10.1145/1168117.1168120>.

PRIM, R. C. Shortest connection networks and some generalizations. **The Bell System Technical Journal**, v. 36, n. 6, p. 1389–1401, Nov 1957. ISSN 0005-8580. Disponível em: <https://doi.org/10.1002/j.1538-7305.1957.tb01515.x>.

RAI, D.; TYAGI, K. Bio-inspired optimization techniques: A critical comparative study. **SIGSOFT Softw. Eng. Notes**, ACM, New York, NY, USA, v. 38, n. 4, p. 1–7, jul. 2013. ISSN 0163-5948. Disponível em: <https://doi.org/10.1145/2492248.2492271>.

RIVEROS, F. et al. A many-objective ant colony optimization applied to the traveling salesman problem. **J. of Computer Science & Technology**, v. 16(2), p. 89–94, 2016. Disponível em: <http://journal.info.unlp.edu.ar/JCST/article/view/496>.

SCHAFFER, J. D. Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition). Vanderbilt University, 1984. Disponível em: <https://dl.acm.org/citation.cfm?id=912151>.

SOUZA, M. Z. d.; POZO, A. T. R. Multiobjective binary aco for unconstrained binary quadratic programming. In: **2015 Brazilian Conference on Intelligent Systems (BRACIS)**. [s.n.], 2015. p. 86–91. Disponível em: <https://doi.org/10.1109/BRACIS.2015.15>.

SRINIVAS, N.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. **Trans. Evol. Comput.**, v. 2(3), p. 221–248, 1994. Disponível em: <https://doi.org/10.1162/evco.1994.2.3.221>.

TOTH, P. Dynamic programming algorithms for the zero-one knapsack problem. **Computing**, v. 25, n. 1, p. 29–45, Mar 1980. ISSN 1436-5057. Disponível em: <https://doi.org/10.1007/BF02243880>.

USLU, F. S. Solving knapsack problem with genetic algorithm. In: **2015 23nd Signal Processing and Communications Applications Conference (SIU)**. [s.n.], 2015. p. 1062–1065. ISSN 2165-0608. Disponível em: <https://doi.org/10.1109/SIU.2015.7130016>.

WHILE, L.; BRADSTREET, L.; BARONE, L. A fast way of calculating exact hypervolumes. **IEEE Transactions on Evolutionary Computation**, v. 16, n. 1, p. 86–95, Feb 2012. ISSN 1089-778X. Disponível em: <https://doi.org/10.1109/TEVC.2010.2077298>.

ZHANG, Q.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. **Evol. Comput.**, v. 11(6), p. 712–731, 2007. Disponível em: <https://doi.org/10.1109/TEVC.2007.892759>.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. **TIK-report**, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), v. 103, 2001. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.4617>.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. **Trans. Evol. Comput.**, v. 3(4), p. 257–271, 1999. Disponível em: <https://doi.org/10.1109/4235.797969>.

Apêndices

APÊNDICE **A**

Tabelas de resultados

Este apêndice traz todos os resultados mostrados nos gráficos do capítulo 6 desta dissertação na forma de tabelas.

Tabela 14 – Valores referentes aos experimentos para o PMM na seção 6.2

		30 itens			50 itens			100 itens			Análise consolidada		
	Algoritmo	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**
2 objetivos	NSGA-II	4.55	6.11	13.77	22.75	6.70	31.75	66.77	11.25	22.06	31.36	8.02	22.53
	NSGA-III	15.49	16.55	7.40	42.15	19.40	7.03	64.11	36.90	4.06	40.58	24.28	6.16
	SPEA2	3.28	5.04	14.18	27.41	6.15	34.40	70.65	9.78	22.25	33.78	6.99	23.61
	MOEA/D	19.66	22.42	10.96	72.20	21.60	9.22	94.61	33.69	2.45	62.16	25.90	7.54
	AEMMT	3.05	5.19	14.47	59.08	18.60	15.91	99.56	117.25	0.19	53.90	47.01	10.19
3 objetivos	NSGA-II	4.84	4.40	74.92	28.78	6.84	93.06	86.46	20.23	21.49	40.02	10.49	63.16
	NSGA-III	20.01	15.50	25.93	33.77	13.19	31.57	70.69	19.05	22.56	41.49	15.91	26.69
	SPEA2	5.32	4.38	76.46	26.06	6.30	110.88	75.56	18.36	36.66	35.65	9.68	74.67
	MOEA/D	21.81	13.87	42.87	46.81	12.57	59.20	80.00	10.85	88.06	49.54	12.43	63.38
	AEMMT	4.35	4.66	78.73	46.23	10.08	84.13	94.67	33.71	14.35	48.41	16.15	59.07
4 objetivos	NSGA-II	20.83	13.94	116.06	29.20	13.08	105.67	98.26	63.87	3.11	49.43	30.30	74.95
	NSGA-III	13.01	15.26	59.29	19.90	15.12	55.29	67.45	18.89	43.10	33.45	16.42	52.56
	SPEA2	7.79	8.72	138.32	25.28	13.04	112.08	97.77	70.31	3.35	43.61	30.69	84.58
	MOEA/D	9.92	9.29	145.89	27.51	12.31	118.44	83.06	8.60	202.68	40.16	10.07	155.67
	AEMMT	2.77	4.53	219.68	14.49	6.65	211.99	80.50	14.90	100.19	32.59	8.70	177.29
5 objetivos	NSGA-II	34.49	25.19	112.11	54.63	23.11	74.87	98.13	110.25	3.55	62.42	52.85	63.51
	NSGA-III	20.60	16.47	97.66	22.96	16.90	66.68	67.62	24.72	51.42	37.06	19.36	71.92
	SPEA2	22.88	24.23	115.68	53.07	26.93	70.40	97.85	129.53	3.23	57.93	60.23	63.10
	MOEA/D	11.97	6.31	450.46	26.24	9.37	233.34	67.11	6.99	618.23	35.11	7.55	434.01
	AEMMT	2.10	3.16	535.16	13.47	5.70	346.11	70.44	12.04	244.94	28.67	6.97	375.40
6 objetivos	NSGA-II	52.26	33.83	85.61	86.80	88.97	25.28	95.25	161.17	9.21	78.10	94.65	40.03
	NSGA-III	31.01	22.84	85.43	50.91	25.97	84.30	61.71	28.52	64.16	47.88	25.78	77.96
	SPEA2	48.00	37.71	78.00	86.36	108.10	20.46	97.93	203.68	3.11	77.43	116.50	33.86
	MOEA/D	22.67	8.04	480.24	39.40	8.66	724.20	58.41	7.38	965.35	40.16	8.03	723.26
	AEMMT	5.77	4.23	594.85	18.32	6.77	757.59	52.54	10.30	536.91	25.55	7.10	629.78

Tabela 15 – Valores referentes aos experimentos para o PRM na seção 6.2

		Rede 1			Rede 2			Rede 3			Análise consolidada		
	Algoritmo	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**
2 objetivos	NSGA-II	3.96	0.11	13.01	0.22	0.00	8.92	6.82	0.09	4.73	3.67	0.07	8.89
	NSGA-III	0.07	0.00	13.85	0.00	0.00	9.00	2.65	0.06	4.89	0.91	0.02	9.25
	SPEA2	1.80	0.82	13.42	0.74	0.32	8.98	11.65	12.84	4.21	4.73	4.66	8.87
	MOEA/D	9.27	0.50	11.98	7.84	0.04	7.96	31.52	0.61	3.10	16.21	0.38	7.68
	AEMMT	0.77	0.45	13.43	0.00	0.00	9.00	15.75	22.32	3.82	5.51	7.59	8.75
3 objetivos	NSGA-II	2.73	2.22	27.87	0.18	0.07	16.91	9.36	13.20	13.95	4.09	5.16	19.58
	NSGA-III	1.34	1.55	28.61	0.00	0.00	17.00	4.18	3.34	14.89	1.84	1.63	20.17
	SPEA2	3.31	2.73	25.74	2.36	0.89	15.41	19.77	11.24	9.28	8.48	4.95	16.81
	MOEA/D	11.80	11.29	21.03	9.57	2.62	13.69	16.83	10.24	9.51	12.73	8.05	14.74
	AEMMT	3.66	2.77	26.57	0.39	0.14	16.51	9.18	6.25	11.10	4.41	3.05	18.06
4 objetivos	NSGA-II	11.11	3.55	80.03	29.47	8.02	30.88	8.92	4.20	44.72	16.50	5.26	51.88
	NSGA-III	8.23	2.86	82.95	22.17	5.47	34.61	2.51	1.01	50.51	10.97	3.11	56.02
	SPEA2	13.84	4.00	51.59	33.53	7.96	27.55	14.30	3.61	37.55	20.56	5.19	38.90
	MOEA/D	19.71	4.54	66.58	31.79	10.36	19.69	27.93	7.92	23.97	26.48	7.61	36.75
	AEMMT	17.78	4.26	74.61	24.40	6.84	29.38	11.65	4.08	37.70	17.94	5.06	47.23
5 objetivos	NSGA-II	25.86	7.09	73.18	26.97	6.24	69.46	35.18	4.99	66.25	29.34	6.11	69.63
	NSGA-III	20.46	5.29	77.31	16.86	4.15	77.46	22.65	3.83	76.34	19.99	4.42	77.04
	SPEA2	25.62	7.06	66.94	27.52	4.29	65.23	23.19	3.40	69.13	25.44	4.92	67.10
	MOEA/D	16.33	2.87	152.52	24.46	5.05	85.60	15.52	2.24	140.80	18.77	3.39	126.31
	AEMMT	12.93	3.19	108.79	16.57	3.70	80.85	9.15	2.01	107.75	12.88	2.97	99.13
6 objetivos	NSGA-II	36.32	7.26	68.54	31.41	7.20	69.15	48.37	9.89	56.20	38.70	8.11	64.63
	NSGA-III	29.83	5.45	71.06	17.36	4.53	79.87	16.32	3.41	85.17	21.17	4.46	78.70
	SPEA2	34.46	6.13	58.99	32.87	5.81	60.42	36.73	4.96	56.94	34.69	5.63	58.78
	MOEA/D	12.40	1.57	287.23	16.26	3.44	155.38	11.02	1.87	232.68	13.23	2.29	225.10
	AEMMT	3.59	1.10	86.77	5.76	1.71	84.82	0.14	0.04	89.87	3.16	0.95	87.15

Tabela 16 – Valores referentes às métricas ER, GD e PS dos experimentos para o PMM na seção 6.4

		30 itens			**40 itens**			**50 itens**			**Análise consolidada**		
	Algoritmo	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**
4 objetivos	NSGA-III	14.59	115.63	57.89	20.01	76.96	56.76	19.47	80.22	57.26	18.03	90.94	57.30
	MOEA/D	10.02	97.27	150.52	20.17	45.64	173.62	26.87	45.80	123.32	19.02	62.90	149.15
	AEMMT	2.57	179.23	220.45	10.20	56.14	251.43	14.19	56.13	216.82	8.99	97.17	229.57
	AEMMD	7.40	130.01	220.97	30.40	34.94	201.67	58.68	435.44	81.03	32.16	200.13	167.89
	MACO/NDS	2.69	116.18	283.48	6.90	41.31	653.38	5.96	54.65	568.53	5.18	70.71	501.80
5 objetivos	NSGA-III	19.37	88.61	99.44	22.95	68.25	91.00	22.38	76.05	69.19	21.57	77.64	86.54
	MOEA/D	11.16	57.30	455.61	20.72	37.79	330.42	29.76	33.82	226.62	20.55	42.97	337.55
	AEMMT	1.62	187.66	541.93	8.87	59.03	409.77	14.16	44.43	342.52	8.22	97.04	431.41
	AEMMD	10.45	49.86	792.00	32.23	29.05	443.68	48.78	43.25	233.93	30.49	40.72	489.87
	MACO/NDS	3.33	61.51	1184.18	8.35	32.12	1437.39	11.29	29.73	1279.29	7.66	41.12	1300.29
6 objetivos	NSGA-III	30.30	76.86	86.50	28.01	80.29	79.45	50.31	48.54	85.42	36.20	68.56	83.79
	MOEA/D	22.00	35.85	495.49	19.05	40.86	485.22	39.55	19.02	721.76	26.87	31.91	567.49
	AEMMT	5.52	76.36	602.56	5.13	78.96	648.50	18.74	31.11	757.45	9.79	62.14	669.50
	AEMMD	15.15	25.80	1247.95	21.50	31.31	746.57	48.48	13.78	892.45	28.38	23.63	962.32
	MACO/NDS	6.36	36.53	1665.34	8.67	32.16	1956.80	30.21	11.85	3368.55	15.08	26.85	2330.23

Tabela 17 – Valores referentes à métrica Tempo dos experimentos para o PMM na seção 6.4

		30 itens	**40 itens**	**50 itens**	**Análise consolidada**
	Algoritmo	**Tempo**	**Tempo**	**Tempo**	**Tempo**
4 objetivos	NSGA-III	4.21	4.26	4.23	4.23
	MOEA/D	1.07	1.24	1.07	1.13
	AEMMT	5.19	5.24	5.19	5.21
	AEMMD	1.32	1.29	1.17	1.26
	MACO/NDS	4.80	7.69	9.30	7.26
5 objetivos	NSGA-III	4.43	4.42	4.47	4.44
	MOEA/D	2.16	1.77	1.54	1.82
	AEMMT	11.08	10.77	10.73	10.86
	AEMMD	5.28	4.56	3.56	4.47
	MACO/NDS	10.76	12.11	12.56	11.81
6 objetivos	NSGA-III	4.68	4.66	4.70	4.68
	MOEA/D	2.47	2.20	3.25	2.64
	AEMMT	23.15	23.01	24.41	23.52
	AEMMD	17.51	12.75	17.32	15.86
	MACO/NDS	14.35	17.41	31.14	20.97

Tabela 18 – Valores referentes às métricas *ER*, *GD* e *PS* dos experimentos para o PRM na seção 6.4

		Rede 1			**Rede 2**			**Rede 3**			**Análise consolidada**		
	Algoritmo	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**	**ER**	**GD**	**PS**
4 objetivos	NSGA-III	17.10	1.44	36.23	9.07	0.70	13.17	18.89	1.34	24.66	15.02	1.16	24.69
	MOEA/D	27.83	1.43	42.38	26.10	1.51	11.40	23.57	1.14	18.05	25.83	1.36	23.94
	AEMMT	13.66	1.50	73.93	17.74	1.04	28.46	15.17	0.88	27.43	15.52	1.14	43.27
	AEMMD	2.82	1.92	102.28	5.47	0.76	30.91	33.99	0.73	17.84	14.09	1.14	50.34
	MACO/NDS	17.19	1.19	71.37	33.95	0.74	37.28	50.69	0.66	16.54	33.95	0.86	41.73
5 objetivos	NSGA-III	23.96	0.71	26.25	27.13	0.82	13.87	22.45	0.46	42.26	24.51	0.66	27.46
	MOEA/D	24.47	0.56	73.95	19.45	0.70	41.17	19.70	0.37	121.57	21.21	0.55	78.90
	AEMMT	16.88	0.45	118.51	13.63	0.68	85.51	15.73	0.43	108.83	15.41	0.52	104.28
	AEMMD	7.34	0.53	255.81	5.43	0.70	169.02	17.95	0.27	229.30	10.24	0.50	218.04
	MACO/NDS	20.75	0.31	198.28	15.87	0.37	164.25	29.48	0.29	171.53	22.03	0.32	178.02
6 objetivos	NSGA-III	20.49	0.74	44.85	32.87	0.56	17.47	52.69	0.16	29.46	35.35	0.49	30.59
	MOEA/D	19.70	0.37	121.57	18.52	0.93	56.01	11.50	0.38	183.90	16.58	0.56	120.49
	AEMMT	8.70	0.58	192.85	23.12	0.41	113.67	37.19	0.11	121.19	23.00	0.36	142.57
	AEMMD	5.82	0.46	492.88	14.54	0.26	266.29	33.11	0.09	299.11	17.82	0.27	352.76
	MACO/NDS	13.42	0.34	404.36	18.01	0.24	275.22	43.56	0.19	188.77	25.00	0.26	289.45

Tabela 19 – Valores referentes à métrica Tempo dos experimentos para o PRM na seção 6.4

		Rede 1	**Rede 2**	**Rede 3**	**Análise consolidada**
	Algoritmo	**Time**	**Time**	**Time**	**Time**
4 objetivos	NSGA-III	8.57	12.87	21.70	14.38
	MOEA/D	3.53	5.71	10.46	6.57
	AEMMT	11.99	16.94	26.46	18.46
	AEMMD	7.58	11.95	19.44	12.99
	MACO/NDS	4.34	10.11	9.96	8.14
5 objetivos	NSGA-III	8.78	13.58	21.55	14.64
	MOEA/D	3.71	6.50	11.43	7.21
	AEMMT	18.13	23.33	35.44	25.63
	AEMMD	11.86	18.98	29.45	20.10
	MACO/NDS	4.63	9.30	9.31	7.75
6 objetivos	NSGA-III	9.03	13.43	22.24	14.90
	MOEA/D	3.98	6.37	11.85	7.40
	AEMMT	28.38	35.79	51.71	38.63
	AEMMD	25.54	34.30	47.82	35.89
	MACO/NDS	4.36	8.89	9.14	7.46

Tabela 20 – Valores referentes aos experimentos para o PMM na seção 6.5

		50 itens		100 itens		200 itens	
	Algoritmo	**Hipervolume**	**Tempo**	**Hipervolume**	**Tempo**	**Hipervolume**	**Tempo**
4 objetivos	NSGA-III	1.72176e+16	60.83	1.1154e+18	279.03	1.34228e+19	380.75
	MOEA/D	2.02321e+16	1.04	1.04208e+18	1.70	5.39396e+18	1.83
	AEMMT	2.38188e+16	5.74	8.93346e+17	9.77	3.74724e+18	11.77
	AEMMD	1.33635e+16	1.13	5.28118e+17	3.46	1.90786e+18	3.44
	MOEA/D-ACO	6.96528e+16	6.38	3.62735e+18	11.06	4.12522e+19	27.30
	MOACS	5.813e+16	7.41	1.74587e+18	14.26	1.9033e+19	30.66
	MACO/NDS	5.7667e+16	7.51	2.36262e+18	12.13	2.59024e+19	28.28
5 objetivos	NSGA-III	7.70349e+20	334.16	1.04171e+23	1833.71	2.77075e+24	1612.17
	MOEA/D	6.18603e+20	1.62	4.15004e+22	2.23	6.08667e+23	2.32
	AEMMT	7.77715e+20	13.19	4.35547e+22	19.51	5.94057e+23	23.12
	AEMMD	7.14715e+20	3.94	3.83953e+22	10.08	2.70665e+23	7.09
	MOEA/D-ACO	4.46128e+21	7.62	2.65165e+23	14.51	8.32665e+24	30.52
	MOACS	2.46506e+21	13.43	1.18187e+23	28.36	2.51017e+24	52.77
	MACO/NDS	3.3126e+21	9.92	1.74575e+23	18.71	3.93817e+24	34.03
6 objetivos	NSGA-III	2.28385e+26	4144.64	8.34169e+27	2109.95	4.68628e+29	1848.96
	MOEA/D	8.56771e+25	2.93	2.52205e+27	2.72	5.38171e+28	2.64
	AEMMT	8.75267e+25	21.53	3.80597e+27	46.80	7.75364e+28	49.83
	AEMMD	1.03279e+26	20.04	3.48037e+27	38.87	4.62281e+28	24.53
	MOEA/D-ACO	5.36626e+26	12.41	2.57833e+28	14.41	1.60168e+30	29.60
	MOACS	2.78176e+26	28.72	6.07046e+27	41.06	2.762e+29	58.97
	MACO/NDS	4.18841e+26	22.48	1.32966e+28	19.28	6.33141e+29	34.47

Tabela 21 – Valores referentes aos experimentos para o PRM na seção 6.5

		Rede 3		Rede 4		Rede 5	
	Algoritmo	**Hipervolume**	**Tempo**	**Hipervolume**	**Tempo**	**Hipervolume**	**Tempo**
4 objetivos	NSGA-III	1594.33	12.70	972.16	7.02	1035.87	9.00
	SPEA2-SDE	2206.90	15.07	997.61	9.96	4985.00	12.42
	MOEA/D	1029.35	9.87	1231.78	5.61	4292.38	6.46
	AEMMT	2140.68	15.47	189.30	11.02	5305.32	11.93
	AEMMD	922.32	11.20	873.43	6.10	4755.68	8.71
	MOEA/D-ACO	1494.31	4.11	7502.47	4.35	3132.88	6.27
	MOACS	2547.58	6.32	8330.36	7.11	5903.63	9.47
	MACO/NDS	2364.66	8.15	5952.99	9.51	5927.17	13.43
5 objetivos	NSGA-III	9001.57	42.06	6553.66	21.33	8416.00	10.49
	SPEA2-SDE	66647.20	102.89	122748.86	42.95	34976.66	15.78
	MOEA/D	17483.11	10.98	65579.40	6.31	24611.74	7.09
	AEMMT	7529.99	20.73	45310.37	16.46	38578.48	17.53
	AEMMD	65747.37	17.02	94005.70	10.02	47296.57	12.75
	MOEA/D-ACO	27420.78	4.07	63565.14	4.54	13491.39	6.25
	MOACS	44303.80	6.75	87680.73	7.30	23750.76	9.66
	MACO/NDS	46033.28	7.78	75276.40	8.26	43665.57	10.99
6 objetivos	NSGA-III	319506.59	83.15	129486.91	48.85	53130.23	34.09
	SPEA2-SDE	4.768243269e+6	392.13	8.073723833e+6	179.57	4.731148423e+6	104.41
	MOEA/D	665174.63	11.97	3.801447428e+6	6.79	2.385996998e+6	7.60
	AEMMT	244973.65	30.40	1.202940102e+6	24.52	3.954286392e+6	26.75
	AEMMD	4.39478916e+6	26.57	7.329716622e+6	17.76	7.577439767e+6	21.75
	MOEA/D-ACO	945138.72	4.40	5.875876067e+6	4.48	743365.30	6.46
	MOACS	4.162784128e+6	7.28	7.816353705e+6	7.97	4.273903197e+6	10.11
	MACO/NDS	2.538590639e+6	8.44	8.384286786e+6	8.52	7.402986048e+6	11.25