Um Modelo para Tese, Qualificações e Dissertações
Exemplo de título Longo

Tiago Peres França

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

Uberlândia
2018

Tiago Peres França

Um Modelo para Tese, Qualificações e Dissertações
Exemplo de título Longo

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.

A474m Sobrenome, Nome do aluno, 1979-

2014 Título do Trabalho / Nome e Sobrenome do aluno. - 2014.
81 f. : il.

Orientador: Nome do Orientador.

Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1.Computação - Teses. 2. Simulação (Computadores) - Teses. I. Sobrenome, Nome do orientador. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

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 "**Título do trabalho**" por **Nome do aluno** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação.**

Uberlândia, ___ de _____ de _____

Orientador: _____
Prof. Dr. Nome do orientador
Universidade Federal de Uberlândia

Coorientador: _____
Prof. Dr. Nome do coorientador
Universidade Federal de Uberlândia
(quando houver)

Banca Examinadora:

_____
Prof. Dr. Membro da banca 1
Instituição de Ensino Superior

_____
Prof. Dr. Membro da banca 2
Instituição de Ensino Superior

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Faça os agradecimentos àqueles que direta ou indiretamente contribuíram para que você tivesse obtido êxito. Inclua na sua lista agradecimentos aos órgãos de fomento, quando for o caso.

"Sua vida pode ser dividida em dois períodos: antes de agora e a partir de agora."
(Prof. Obvious Stating)

Resumo

Segundo a ABNT (2003, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Para auxiliá-lo com o latex, o Apêndice **??** apresenta os resultados dos comandos incluídos no arquivo ape_comandos/abntex2-modelo-include-comandos.tex

Palavras-chave: Latex. Abntex. Normas USP.

Abstract

This is the english abstract.

Keywords: Latex. Abntex..

Lista de ilustrações

Lista de tabelas

Lista de siglas

Sumário

Capítulo 1

Introdução

Qual o menor caminho para atingir um destino? Qual o melhor carro para se comprar dado um orçamento? Qual a forma mais rápida de se executar uma tarefa? Enfim, dado um conjunto de soluções possíveis, qual melhor resolve um dado problema? A pergunta é simples, mas encontrar a resposta correta em um tempo viável é complexo e configura um dos principais campos de pesquisa da computação: busca e otimização.

Só há uma forma de se saber qual é, indubitavelmente, a melhor solução para um problema: gerando e comparando todas as possibilidades. Para a maioria dos problemas interessantes, essa é uma tarefa difícil, se não impossível. Para um grafo com muitas arestas, por exemplo, combinar e testar todas as possibilidades é um problema NP-completo, ou seja, não será resolvido em tempo hábil considerando o atual estado da arte da computação. Sendo assim, a busca e otimização se concentra em desenvolver meios para se encontrar soluções suficientemente próximas à melhor possível, ou seja, trabalha-se com estratégias de aproximação. Dentre as estratégias de aproximação, destacam-se os algoritmos gulosos [] e os algoritmos evolutivos [].

Os problemas de otimização se tornam ainda mais complexos quando deseja-se otimizar múltiplas métricas, por exemplo, ao comprar um carro, não deseja-se apenas o mais barato ou o mais potente, mas considera-se preço, potência, aparência, eficiência, quilometragem, etc. Isto é, não basta otimizar um único objetivo, mas sim, vários. A maneira mais simples de se contornar este problema é transformar as diversas métricas em uma única função, fazendo uma média ponderada dos objetivos. Infelizmente, reduzir os múltiplos objetivos em uma única função não é ideal, pois é necessário ter um conhecimento prévio do problema para se decidir os pesos de cada métrica e um único peso mal escolhido pode impedir as melhores soluções de serem encontradas. Na realidade, se todos os objetivos são considerados igualmente importantes, existirão várias soluções possíveis, algumas terão melhor desempenho em uma métrica enquanto outras se darão melhor em outras. Devido à natureza dos problemas multiobjetivos (PMOs) de envolverem diversas soluções, se tornou natural a escolha dos algoritmos genéticos, já que envolvem evoluir uma população de soluções de acordo com um objetivo.

Neste trabalho investigou-se diversos algoritmos genéticos para os PMOs e afim de colocá-los à prova utilizou-se dois problemas discretos bem conhecidos na literatura de busca e 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, onde ao invés de um único valor de lucro, todo item possui m valores, onde m é o número de objetivos. O PMM é 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 prático geralmente encontrado em comunicações de rede, nele deseja-se transmitir uma mensagem de um dispositivo fonte para múltiplos dispositivos em uma rede de computadores de forma a utilizar os recursos disponíveis da forma mais eficiente possível, o que é de extrema importância considerando o grande número de transmissões multimídia e aplicações em tempo real que se beneficiariam muito de algoritmos eficientes para cálculos de rota.

Problemas com dois e três objetivos são considerados bem resolvidos, mas a partir de quatro, torna-se difícil encontrar boas soluções com os algoritmos evolutivos multiobjetivos (AEMOs) clássicos. Chama-se de "many-objectives"os problemas com 4 ou mais funções de otimização e, para resolvê-los, necessita-se de métodos mais robustos que consideram o acréscimo no número de objetivos com estratégias de decomposição em funções escalares, evolução de indicadores, cálculos de distância mais eficientes, etc. Neste trabalho testamos vários algoritmos em diversas formulações de objetivos, variando entre 2 e 6 funções, analisando a qualidade das soluções obtidas em cada um dos casos e discutindo as características que permitem um ou outro atingir determinado resultado.

A maior parte dos trabalhos em busca e otimização multiobjetivo utilizam algoritmos genéticos, mas na mesma linha existem os métodos baseados em colônias de formigas (ACO) e os algoritmos inspirados em inteligência de enxame (PSO). Os PSO's são indicados especialmente para problemas contínuos e a utilização de seus cálculos vetoriais se torna difícil em problemas discretos. Por outro lado, os ACOs lidam especialmente com problemas discretos e representações em grafos. Como ambos os problemas estudados neste trabalho (PMM e PRM) são discretos, decidiu-se concentrar apenas nos algoritmos genéticos e nas colônias de formigas, dando prioridade a este último a fim de se propor um novo modelo para a resolução de problemas many-objectives que fugisse um pouco da já extensa lista de proposições baseadas em algorítimos genéticos. Com uma proposta que utiliza uma metodologia relativamente pouco explorada, surge a esperança de se desenvolver um modelo único que aborda de forma diferente o espaço de busca, possibilitando tanto ganhos de desempenho em tempo quanto em qualidade das soluções.

O modelo baseado em ACO proposto foi chamado de Many-Objective Ant Colony Optimization Based on Decomposed Pheromones (MACO/D), em português, otimização para muitos objetivos com colônia de formigas baseada em decomposição de feromônios.

O algoritmo proposto foi comparado tanto com algoritmos genéticos quanto outros ACOs encontrados na literatura. A partir dos resultados dos experimentos mostrados mais adiante no texto, foi possível comprovar a eficácia do método na maior parte dos cenários e analisar possíveis melhorias para pesquisas futuras.

1.1 Objetivos

Esta dissertação tem como objetivo principal estender os trabalhos de [Fialho] e [Bueno] sobre o problema do roteamento multicast (PRM), mas dando ênfase maior nos algoritmos e modelos multiobjetivos em sí, utilizando o PRM como exemplo de aplicação. Afim de enriquecer o estudo, introduz-se uma nova aplicação, o problema da mochila multiobjetivo. Em linhas gerais, esta pesquisa objetiva:

- ❑ Adotar de um novo problema suficientemente diferente do roteamento multicast a fim de melhor suportar os resultados até então obtidos para os algoritmos estudados e propostos. o novo problema introduzido por este trabalho é o Problema da Mochila Multiobjetivo (PMM) e, apesar de ter aplicações mais restritas, apresenta diferenças interessantes em sua formulação capazes de aprofundar a análise do comportamento dos vários algoritmos.
- ❑ Introduzir e estudar os resultados obtidos pelo hipervolume, uma métrica de desempenho de algoritmos multiobjetivos até então não explorada pelo grupo de pesquisa.
- ❑ Analisar em ambos PMM e PRM o comportamento de cada algoritmo em relação à complexidade do espaço de busca e ao número de objetivos.
- ❑ Propor um modelo para a construção de soluções em algoritmos baseados em colônias de formigas para ambos os problemas da mochila e do roteamento multicast.

1.2 Contribuições

Este trabalho contribui para os campos de busca e otimização multiobjetivo e comunicações em rede (através do problema do roteamento multicast). os principais resultados desta pesquisa são resumidos nos seguintes tópicos:

- ❑ O AEMMT foi proposto originalmente para sequenciamento de proteínas [] e em [] foi utilizado para resolver o problema do roteamento multicast (PRM). Neste trabalho exploramos uma terceira aplicação do algoritmo, aplicando-no sobre o problema da mochila objetivo.
- ❑ O AEMMD foi proposto e analisado em [] para resolver o PRM. Sua eficácia como algoritmo multiobjetivo separado do problema do roteamento ainda não havia sido

comprovada. Neste trabalho mostramos como é possível aplicar o AEMMD ao PMM e fazemos uma análise sobre os limites do algoritmo em relação ao tamanho do espaço de busca.

- Através da execução de vários algoritmos multi e many-objectives 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, desta forma, possibilitando uma comparação profunda entre os principais algoritmos da literatura multi-objetivo.

- Neste trabalho propõe-se um algoritmo eficiente para a construção de soluções para o PRM a partir de uma tabela de feromônios, parte essencial de qualquer algoritmo embasado em colônia de formigas. A fim de propor tal estratégia, apresenta-se mais a frente no texto outros métodos e diversos testes que permitiram o desenvolvimento do algoritmo proposto.

- A principal contribuição desta dissertação está no novo algoritmo proposto Many-objective Ant Colony Optimization with Decomposed Pheromones (MACO/D). Uma estratégia baseada em colônia de formigas e decomposição de objetivos para de forma rápida e eficiente resolver problemas com muitos objetivos. O novo método foi aplicado aos problemas PPM e PRM e comparado com os demais algoritmos.

1.3 Organização do texto

Este trabalho está dividido em capítulos divididos da seguinte forma:

- Capítulo 2: ...
- Capítulo 3: ...
- Capítulo 4: ...
- Capítulo 5: ...
- Capítulo 6: ... ...

Capítulo 2

Otimização bio-inspirada

Grande parte dos problemas de otimização envolvem encontrar a melhor opção num conjunto de possibilidades que cresce de maneira exponencial. Tais problemas são impossíveis de se resolver em tempo viável e necessitam de estratégias inteligentes para aproximar-se da solução ótima em tempo hábil. Dentre essas estratégias destacam-se os algoritmos gulosos [] e os algoritmos de busca bio-inspirados [].

Os algoritmos gulosos são algoritmos relativamente simples que, apesar de nem sempre obterem a solução ótima, normalmente aproximam-se bem dela [14]. Tais métodos são geralmente utilizados quando apenas um critério é envolvido na otimização, quando mais de um objetivo deve ser analisado, o problema se torna bem mais complexo e essa abordagem deixa de ser indicada.

A otimização bio-inspirada lança mão de estratégia baseadas na natureza para se encontrar boas soluções de forma eficiente. Assim como os algoritmos gulosos, não é possível garantir que se encontre exatamente a solução ótima, mas com uma boa modelagem do problema, é possível encontrar soluções suficientemente próximas. Na natureza, o processo de encontrar uma melhor solução aparece a todo momento, desde a forma como os primeiros animais surgiram, 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 matemática [].

Os principais métodos bio-inspirados na literatura de otimização matemática são os algoritmos genéticos (AG), as colônias de formigas (ACO) e os enxames de partículas (PSO). Os algoritmos genéticos se inspiram na teoria da evolução de Darwin. Segundo [], 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 ainda podem ocorrer mutações aleatórias que podem tanto gerar características benéficas quanto maléficas. 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, o

processo de evolução das espécies nada mais faz do que gerar indivíduos e selecionar os melhores, repetindo o processo 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. 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 a chance do mesmo ser tomado. Tal comportamento serviu de inspiração para os algoritmos de otimização bio-inspirados, dando origem ao ACO. Os PSOs, assim como os ACOs são inspirados no comportamento emergente de populações de animais simples. A otimização por enxame de partículas se inspira na navegação de pássaros, onde cada elemento da formação se guia através dos pássaros à frente. O algorítimo 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, que não é o caso dos problemas explorados nesta dissertação.

Em geral, todo algoritmo de otimização bio-inspirado inicia sua busca através da geração de soluções aleatórias e então entra em um laço, onde 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, o que oferece ao usuário uma gama de boas solução para que possa ele mesmo decidir qual utilizar.

2.1 Algoritmos Genéticos (AGs)

Os algoritmos genéticos são métodos de busca baseados na teoria da evolução de Charles Darwin [6]. 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 mutação, cruzamento e seleção natural. De maneira aleatória, um indivíduo em uma população pode ter alguma característica alterada, esse novo atributo pode ajudá-lo a sobreviver em seu habitat ou atrapalhá-lo. Os indivíduos com mutações favoráveis têm maiores chances de sobreviver e reproduzir, passando suas características benéficas 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, a ideia é partir de soluções aleatórias e após várias iterações de mutação, crossover e seleção encontrar um conjunto de soluções que resolvem bem o problema. Nesta ideia, o indivíduo na população representa uma solução, o meio representa o problema e as operações de mutação e crossover devem ser definidas, respectivamente, de forma a permitir uma alteração aleatória em uma solução e a combinação de duas soluções.

Em um algoritmo genético, primeiramente gera-se uma população inicial com indivíduos aleatórios e calcula-se a aptidão de cada um. 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 execução do laço do AG. O laço do AG consiste em:

1. Sortear os pares de pais;
2. Aplicar o crossover 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 os melhores entre pais e filhos para formar a população da iteração seguinte (elitismo).

O laço do AG termina quando uma condição de término estabelecida pelo usuário é atingida, e.g., 100 iterações.

2.1.1 Representação do indivíduo

A principal dificuldade ao se elaborar um algoritmo genético é definir a representação do indivíduo, cada um representa uma possível solução para o problema e deve ser codificado em uma estrutura que permita a mutação e a troca de características com outro indivíduo (cruzamento). Na proposição original do AG [], o indivíduo é representado de forma binária, ou seja, a solução para o problema é codificada em bits, que podem facilmente ser invertidos (mutação) ou copiados de um elemento 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 o 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 que não binárias também são possíveis, mas apresentam novos desafios. Em problemas de menores caminhos, por exemplo, normalmente

trabalha-se com grafos. Logo, um indivíduo é um grafo e tanto a mutação quanto o cruzamento deverão ser operações em grafos.

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 num algoritmo genético. Além disso, é preciso se certificar que cada solução pode ser representada de uma única maneira e que cada indivíduo pode representar uma única solução (relação um-para-um).

2.1.2 Operadores Genéticos

Nos algoritmos genéticos destacam-se três operações principais: cruzamento (ou crossover), mutação e seleção. O cruzamento e a mutação estão diretamente ligados com a escolha da representação do indivíduo, enquanto a seleção opera sobre a avaliação (fitness) de cada elemento da população.

Num algoritmo genético, cada iteração do laço principal é chamada de de geração. No início de cada geração, sorteia-se pares de pais de acordo com suas aptidões para que seja gerada uma nova população de filhos. Cada par é composto de duas cadeias genéticas, uma correspondente a cada indivíduo do par. Para gerar o filho, cruza-se os dois materiais a fim de obter uma nova solução que compartilhe características de ambos genitores. Este processo é conhecido como cruzamento, ou crossover. Existem várias maneira de se cruzar cadeias genéticas, a escolha depende principalmente da estrutura de dados usada para representar um indivíduo. A estrutura mais comum e proposta no artigo original de Goldberg [] é a cadeia binária, onde uma solução é codificada em bits. Dispondo de uma sequência de bits, a forma mais trivial de se efetuar o crossover é gerando um novo *array* onde a primeira metade dos bits pertence a um pai e a segunda pertence a outro. Existem diversas formas de se combinar duas cadeias de bits, dentre elas pode-se destacar:

- ❏ Ponto de cruzamento único: uma posição i de uma das cadeias é sorteada. O filho herda os i primeiros genes do pai 1 e restante do pai 2.

- ❏ 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. O filho herda a primeira parte do pai 1, a segunda parte do pai 2 e a terceira novamente do pai 1.

- ❏ 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 bit na mesma posição do pai 1 para o filho. Para cada bit 1, copia-se o bit do pai 2.

- Cruzamento aritmético: realiza-se uma operação binária entre a cadeia do pai 1 e do pai 2, e.g. AND, OR, XOR, etc.

Após gerar os materiais genéticos 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 em uma cadeia binária é simples, basta sortear um bit e invertê-lo.

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. Independente do número de filhos, o fato é que, após o processo de crossover, a população será maior que seu limite máximo e por esta razão deve-se eliminar parte dela. A seleção natural sem elitismo, simplesmente elimina a população mais velha (pais). A seleção natural com elitismo é normalmente mais interessante, pois permite a sobrevivência dos indivíduos mais aptos considerando a totalidade da população, sendo assim, avalia-se todos os pais e filhos e mantém-se aqueles com melhor aptidão. A aptidão de cada indivíduo é dada por seu desempenho relativo à função de otimização.

2.2 Colônia de formigas (ACO)

A otimização por colônia de formigas (ACO) 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 colocadas em conjunto. 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 sentida 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, criando um ciclo vicioso.

Ao trazer o conceito de colônia de formigas para a computação, observa-se um grande potencial para se resolver problemas em grafo. Por exemplo, para descobrir o caminho de um vértice A a um vértice B que visita o menor número de vértices em um grafo G sem pesos, bastaria simular várias formigas que partem de A e chegam em B fazendo um caminho baseado na quantidade de feromônios em cada aresta. Ao final de cada iteração, atualiza-se o valor do feromônio em cada aresta de acordo com a evaporação e com as arestas que foram de fato escolhidas pelas formigas. Dessa forma, espera-se que, após

várias iterações, a quantidade de feromônios sejam suficientes para guiar uma formiga pelo melhor caminho entre A e B.

O primeiro passo de um ACO é inicializar as arestas do grafo com quantidade zero de feromônios. O passo seguinte é o laço principal, onde a cada iteração cada formiga na colônia percorre 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 atualiza-se os feromônios em cada aresta. O laço termina quando uma condição de parada pré-determinada é atingida, normalmente um número máximo de iterações.

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. No caso onde o problema é encontrar o menor caminho, o indivíduo pode ser uma lista de vértices. Caso deseja-se encontrar caminhos entre a raiz e diversos destinos, pode-se representar a solução como uma árvore. Nem sempre é claro decidir a codificação da solução, como, por exemplo, no problema da mochila, onde não é trivial a visualização de um grafo no processo da construção da solução.

2.2.2 Construção da solução

Independente da representação escolhida, deve-se ser possível relacionar cada partícula da solução a um depósito de feromônio 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 a próxima geração de formigas. Em cada época (iteração do laço principal) constrói-se um número pré-determinado de soluções, onde cada formiga, a partir dos valores de feromônios, decide as partículas que vão compor sua solução. 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.

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 j.

$$p(i,j) = \frac{\tau_{i,j}^{\alpha} * \eta_{i,j}^{\beta}}{\sum_{v \in vizinhança(i)} \tau_{i,v}^{\alpha} * \eta_{i,v}^{\beta}}$$

Onde:

- $\tau_{i,j}^{\alpha}$: feromônio na aresta (i, j) elevado à constante pré-determinada α. α representa a importância que deve ser dada ao valor do feromônio.
- $\eta_{i,j}^{\beta}$: heurística da aresta (i, j) elevada à constante pré-determinada β. β representa a importância que deve ser dada à heurística. A heurística de uma aresta é dada em função do peso, normalmente 1/peso.
- $vizinhança(i)$: todo vértice em G para o qual é possível construir um caminho para i com apenas uma aresta.

2.2.3 Atualização dos feromônios

Existem duas ocasiões onde o feromônio das arestas 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 que se avalie as soluções e que se incremente o feromônio de acordo com os desempenhos. Ao fim de cada época, utiliza-se a seguinte fórmula para atualizar a estrutura de feromônios τ:

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

Onde:

- ρ: 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 i, j.

A quantidade de feromônio depositada por uma formiga k em uma aresta i, j é dada por:

$$\Delta\tau_{i,j}(k) = \begin{cases} \frac{Q}{L_k}, & \text{if } x \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

Onde:

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

O ACO é normalmente utilizado para encontrar o menor caminho ou árvore de menor custo em um grafo. A esperança é que após várias gerações, as formigas percorram sempre o melhor caminho possível.

Capítulo 3

Otimização multiobjetivo

A otimização multiobjetivo consiste em selecionar as melhores soluções de acordo com múltiplos critérios ao invés de apenas um. Por exemplo, ao estabelecer um 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. A otimização de apenas um objetivo é simples, para que uma solução seja considerada melhor que a outra, basta que ela tenha uma melhor avaliação. Por outro lado, quando se trabalha com mais de uma função de otimização, é preciso usar o conceito de dominância de Pareto.

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

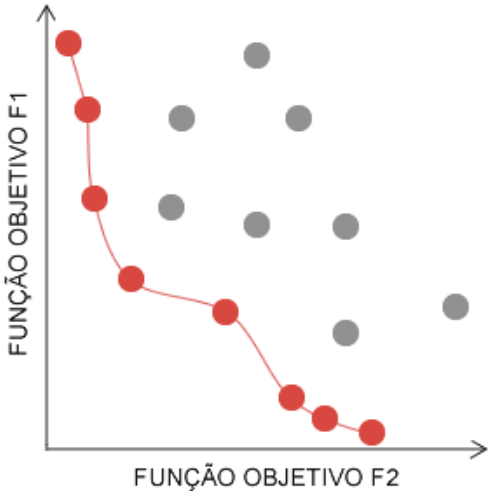
- A é melhor avaliado que B em pelo menos um dos objetivos;
- A não tem avaliação pior que B em nenhum dos objetivos.

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

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

Em problemas de otimização multiobjetivo, o interesse está em encontrar o conjunto de todas as soluções que não são dominadas por nenhuma outra, ou seja, a fronteira de Pareto. Graficamente, a fronteira de Pareto representa a linha formada pelas soluções não-dominadas existentes para o problema. Na figura 3 apresenta-se um exemplo de uma fronteira de Pareto para um problema de minimização com dois objetivos ($F1$ e $F2$), a fronteira de Pareto está representada em vermelho. Observe que nenhum círculo vermelho possui ambos F1 e F2 menores que alguma outra solução em vermelho, ou seja, são não-dominadas. Em contra-partida, toda solução acima da fronteira, em cinza, é dominada, pois existe alguma solução em vermelho que possui ambos valores de F1 e F2 menores. Caso o problema em questão fosse de maximização, a fronteira de Pareto estaria acima de qualquer solução não-dominada ao invés de abaixo.

Figura 1 – Fronteira de Pareto



Não existe limite para o número de funções objetivo em um problema de otimização, mas quanto maior ele for, mais complexa é a busca. Os algoritmos clássicos de otimização multiobjetivo NSGA-II e SPEA2 lidam bem com até dois objetivos, mas a partir de quatro critério de otimização, ambos os métodos sofrem para encontrar soluções relevantes. Desta forma, criou-se a classificação "many-objective". Problemas many-objectives (4 ou mais objetivos) apresentam diversas novas dificuldades e precisam de novas técnicas para que sejam resolvidos eficientemente. Como observado por Deb em [nsga3], os problemas trazidos pelo alto número de objetivos são:

1. Grande parte da população é não dominada: a maioria dos algoritmos multiobjetivos classifica a população de acordo com a dominação de Pareto. Se existem muitas funções objetivo, se torna muito comum que uma solução seja melhor que outra em pelo menos uma das funções. 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: afim de garantir uma boa diversidade populacional, alguns algoritmos medem alguma espécie de distância entre as soluções e removem as que são consideradas mais similares. A maior dimensionalidade traz consequentemente um maior impacto 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 distante 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 afim 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 obter-se bons resultados, é necessário que se manipule grandes populações de indivíduos, o que é computacionalmente caro e dificulta o trabalho do usuário que deverá escolher uma única solução ao final do processo.

5. Métricas de avaliação se tornam difíceis de se calcular: a avaliação das soluções 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. Com 4 funções em diante, se torna difícil tal visualização.

A maior parte dos algoritmos many-objectives (todos mencionados neste trabalho) lidam apenas com os três primeiros problemas. Sobre a quarta dificuldade fazemos um breve estudo mais adiante [citar seção] e as duas últimas não são responsabilidade dos algoritmos de otimização em sí.

3.1 Algoritmos Multiobjetivo

3.1.1 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

O NSGA-II [] é o algoritmo evolutivo multiobjetivo (AEMO) mais frequente na literatura. A atribuição de aptidão (fitness) se dá pela classificação da população em rankings de dominância (fronteiras), de forma que o primeiro contenha todas as soluções não dominadas, o segundo todos os indivíduos não-dominados excluindo a 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, pertencem à mesma fronteira, a fim de diferenciá-las utiliza-se um cálculo de distância (crowding distance), o qual confere melhor avaliação às soluções mais diferenciadas umas das outras, garantindo assim a diversidade da população.

O processo do NSGA-II 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. O primeiro passo continua sendo a geração aleatória dos indivíduos, em seguida classifica-se a população em ranks de dominância e inicia-se o laço principal, o qual termina assim que a condição de parada é atingida. O laço principal do NSGA-II é dado pelo pseudo-código do algoritmo 1.

A seleção de pais utiliza torneio simples para sorteá-los, ou seja, dois elementos da população são escolhidos de forma aleatória, o indivíduo com melhor avaliação é selecio-

Algoritmo 1 Laço principal do NSGA-II

1: **enquanto** número máximo de gerações não for atingido **faça**
2: selecione os pares de pais para o crossover
3: efetue o cruzamento para cada par de pais, gerando os filhos
4: combine a população de pais com a população de filhos
5: classifique todas as soluções em fronteiras (ranks) de dominância
6: calcule a crowding distance para cada uma das soluções
7: aplique a seleção natural sobre a totalidade da população, preservando os indivíduos de melhor rank e, em segundo lugar, crowding distance.
8: **fim enquanto**

nado como um dos pais, sorteia-se mais dois indivíduos, e o melhor dentre eles se torna o segundo pai.

Na linha 2 do pseudo-código, através dos pares de pais, gera-se os filhos através do crossover e da mutação. Após a geração dos filhos, a população corrente e o conjunto de filhos são concatenados (linha 3) e submetidos à classificação em ranks de dominância (linha 4).

A classificação em ranks de dominância recebe um conjunto de soluções e verifica quais dentre elas não são dominadas. O conjunto de soluções não dominadas forma o primeiro rank de dominância. Do conjunto restante (excluindo o primeiro rank), retira-se as soluções não dominadas para formar o segundo rank. Esse processo se repete até que todos os indivíduos tenham sido classificados.

Após toda a população ter sido classificada em ranks, antes de selecionar os indivíduos que vão compor a população na próxima iteração, deve-se calcular a distância de aglomeração (crowding distance) para cada indivíduo em cada rank de dominância. O cálculo de distância, para cada objetivo, ordena o conjunto de soluções e faz uma relação entre as distâncias de cada indivíduo para os vizinhos imediatamente anterior e posterior. Soma-se as distâncias obtidas em cada objetivo para cada solução e define-se aquelas com maior valor de distância como as mais diferentes entre sí.

Com toda a população classificada em ranks e todas as distâncias calculadas, basta formar a nova população com os melhores indivíduos. 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 $tamanho(rank) + tamanho(nova_população) < tam_max_pop$: adiciona-se todos os membros do rank à nova população.
- Caso contrário, se $tamanho(nova_população) < tam_max_pop$: adiciona-se à nova população os $tam_max_pop - tamanho(nova_população)$ elementos do rank com os maiores valores de distância. Termine o processo, a nova população está formada.
- Caso contrário, termine o processo, a nova população está formada.

Desta forma, ao final do algoritmo obtém-se a fronteira de Pareto aproximada no primeiro rank da população gerada na última iteração do algoritmo.

3.1.2 Strength Pareto evolutionary algorithm 2 (SPEA2)

O SPEA2 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. As principais diferenças entre o SPEA2 e um AG comum estão no cálculo de aptidão e na utilização de uma população extra: o arquivo.

O arquivo é responsável por guardar as melhores soluções já encontradas até o momento, funciona como uma espécie de elitismo. Os pais, no cruzamento, 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 compõe o arquivo da geração seguinte. A quantidade de indivíduos no repositório de soluções não-dominadas é limitada e, portanto, quando se excede o tamanho máximo, deve-se executar um processo de truncamento.

O processo de truncamento do arquivo ocorre na seleção natural, a última função executada na iteração do laço principal de um AG. A seleção no SPEA2 se dá pelo cálculo do arquivo da próxima geração: ambas as populações da iteração corrente (população e arquivo) são submetidas à seleção, extrai-se do conjunto total de soluções 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 $tamanho(n_d) = capacidade\_arquivo$, o novo arquivo é formado por n_d;
- Caso contrário, se $tamanho(n_d) < capacidade\_arquivo$, o novo arquivo é formado pela união de n_d com os $capacidade\_arquivo - tamanho(n_d)$ indivíduos restantes com melhor aptidão;
- Caso contrário, se $tamanho(n_d) > capacidade\_arquivo$, o novo arquivo é formado por n_d e deve-se truncá-lo em $tamanho(n_d) - capacidade\_arquivo$ passos, onde em cada passo elimina-se o indivíduo com menor variabilidade genética em relação aos demais.

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*), do *raw fitness* e densidade.

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

$$s(i) = |j| : j \in P \cup E \wedge i \prec j$$

Tendo calculado a força de cada indivíduo, parte-se para o *raw fitness*. O *raw fitness* de um indivíduo i ($r(i)$) é dado pela soma das forças de cada elemento que o domina. Veja a fórmula a seguir:

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

Observe que, caso o indivíduo seja não-dominado, seu *raw fitness* será o menor possível: zero. Após determinar o *raw fitness*, para finalizar o cálculo de aptidão deve-se descobrir a densidade de cada indivíduo ($d(i)$). A densidade é computada de acordo com a distância da solução para seus vizinhos e é dada pela seguinte fórmula:

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

Na fórmula acima, σ_i^k é a k-ésima menor distância entre o indivíduo i e o restante da população. k é a raiz quadrada do tamanho do conjunto de soluções em avaliação, i.e. $k = \sqrt{|P \cup E|}$. O valor de $d(i)$ sempre está no intervalo (0,1). Referencia-se o leitor ao artigo original do SPEA2 [] para mais detalhes sobre o cálculo de densidade.

Finalmente, a aptidão do indivíduo ($f(i)$) é dada pela soma do *raw fitness* e a densidade: $f(i) = r(i) + d(i)$. Como, $\forall i, d(i) < 1$ e, $\forall i, r(i) = 0$ se a solução é não-dominada, $f(i) < 1$ sempre que i é não-dominado.

O laço principal do SPEA2 é explicitado no pseudo-código 2 e como resposta para o problema, retorna-se o arquivo da última geração computada. Espera-se que, após as diversas iterações, o algoritmo tenha conseguido uma boa aproximação da fronteira de Pareto.

Algoritmo 2 Laço principal do SPEA2

```
1: enquanto número máximo de gerações não for atingido faça
2:     a partir do arquivo, selecione os pares de pais para o crossover
3:     efetue o cruzamento para cada par de pais, gerando os filhos
4:     substitua a população corrente pelos filhos
5:     calcule o fitness de todos indivíduos no arquivo e na população
6:     aplique a seleção natural e trunque o arquivo, caso necessário
7: fim enquanto
```

3.2 Algoritmos Many-objectives

3.2.1 Multiobjective evolutionary algorithm based on decomposition (MOEA/D)

O MOEA/D é um algoritmo que avalia os objetivos através de uma função escalarizadora, se baseando na dominância de Pareto apenas para atualizar o conjunto de soluções não dominadas geradas em cada iteração (arquivo). No MOEA/D, um problema multiobjetivo é decomposto em múltiplos problemas mono-objetivos chamados de células. 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, ou indivíduo, 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 levada em consideração para a escolha do pai e seleção natural.

O primeiro passo do MOEA/D é gerar a estrutura de células e vizinhanças, para isso, sorteia-se os vetores de pesos (a soma de cada vetor deve ser igual a um) e para cada um deles, calcula-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. Antes de começar o laço principal, gera-se aleatoriamente uma solução para cada célula e calcula-se as aptidões.

Uma parte fundamental do MOEA/D é a escolha da função escalarizadora, ela é a principal responsável pelo cálculo de aptidão. Em todos experimentos realizados neste trabalho, foi utilizada a soma ponderada, mas outras estratégia como *Penalty-Based Boundary Intersection* e Tchebycheff também podem ser utilizadas [moead]. 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 $2 * 0.3 + 9 * 0.2 + 5 * 0.5 = 4.9$.

No laço principal do MOEA/D, seleciona-se os pais e gera-se os filhos. Para cada célula c_i, dois pais são selecionados aleatoriamente em sua vizinhança. Sempre que um filho é gerado, o processo de seleção é realizado logo em seguida. A aptidão do filho é calculada para cada uma das células na vizinhança de c_i, substituindo a solução anterior da célula caso seu fitness seja melhor. Após o processo de geração de filhos e seleção, atualiza-se o arquivo com as soluções novas soluções não-dominadas.

3.2.2 Non-dominated Sorting Genetic Algorithm III (NSGA-III)

O NSGA-III é uma extensão do NSGA-II que permite o *framework* funcionar melhor para mais de três objetivos. Ele se diferencia do original apenas na fase de seleção, onde ao invés de usar a distância de aglomeração para diferenciar soluções em uma mesma fronteira, utiliza um método de clusterização, onde os indivíduos são divididos em nichos de acordo com suas similaridades. 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 é distribuído sobre a figura e passa a representar cada um, um nicho. Para classificar uma solução, define-se como nicho o ponto de referência mais próximo. Ao final, toma-se como sobreviventes os pontos nas regiões menos lotadas do espaço de busca. Para mais detalhes sobre o processo de clusterização, referencia-se o leitor ao artigo original [NSGA-III].

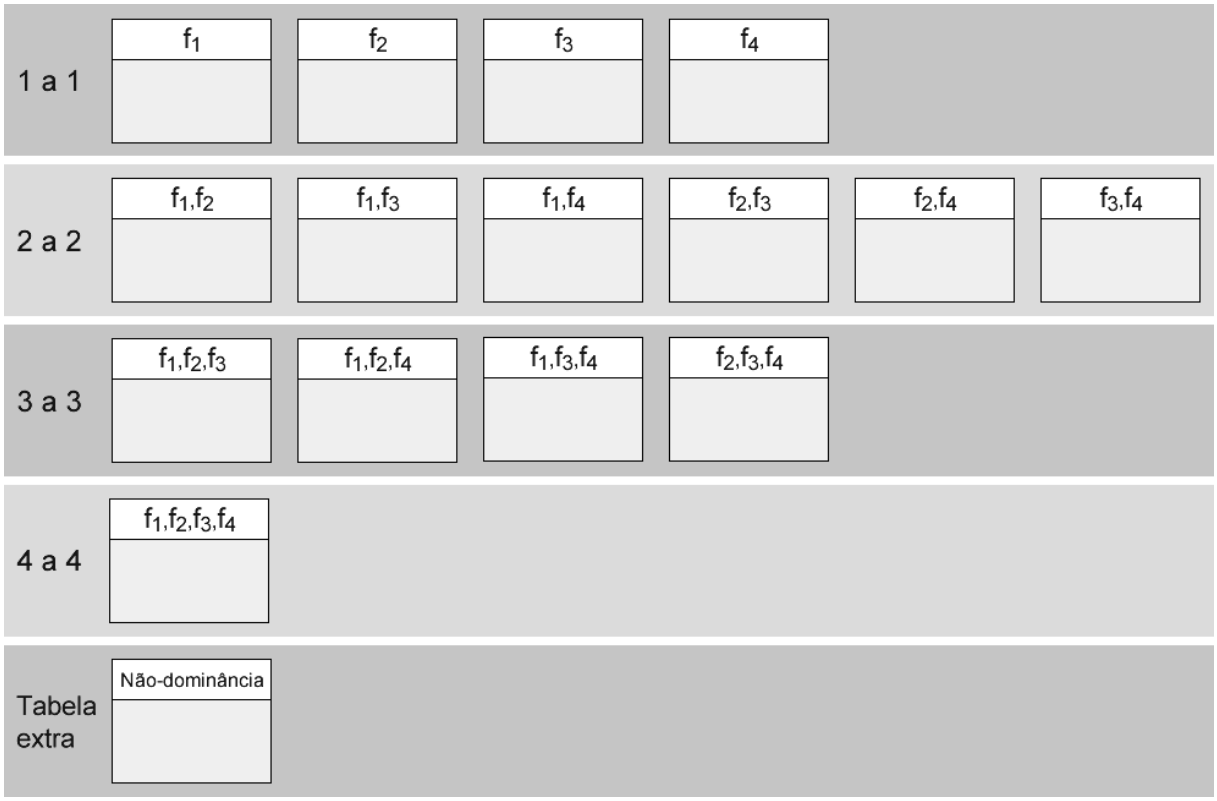
3.2.3 Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas (AEMMT)

O AEMMT, assim como o MOEA/D, decompõe o problema multiobjetivo em subproblemas menores e para isso utiliza um esquema de tabelas, onde cada tabela representa uma combinação diferente de objetivos. A função que transforma os múltiplos objetivos em um valor escalar é sempre a média e cada tabela mantém os melhores indivíduos considerando a média dos objetivos que 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, que será testado em todas as tabelas, entrando naquelas em que representar uma melhor aptidão em relação aos demais indivíduos. Naturalmente, como um único *crossover* é realizado a cada iteração, o AEMMT precisa de mais gerações para efetuar o mesmo número de comparações que os algoritmos citados nas seções anteriores.

A quantidade de tabelas é determinada pelo número de combinações possíveis de objetivos. Para quatro objetivos (f_1, f_2, f_3, f_4), por exemplo, como ilustrado na figura 3.2.3 serão criadas 15 tabelas de combinações mais uma tabela extra, usada para guardar os indivíduos não-dominados.

Cada tabela possui um limite máximo de indivíduos e no início do algoritmo gera-se soluções aleatórias de forma que todas as tabelas sejam completamente preenchidas. No laço principal, um indivíduo só entra em uma tabela t se for melhor que a pior solução na população de t. Com relação a tabela de dominância, sempre que um filho é gerado e não-dominado por nenhum outro indivíduo na tabela, ele é incluído. A restrição no

Figura 2 – Tabelas do AEMMT



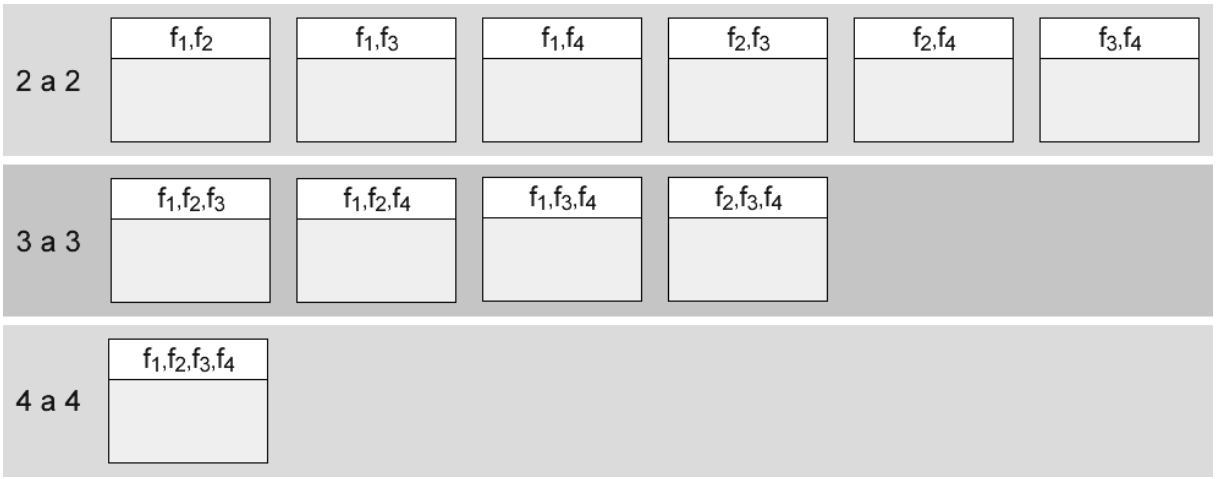
tamanho da tabela de dominância é independente das demais e sempre que o limite for atingido, é feito um truncamento priorizando a permanência das soluções com maior valor de média aritmética entre todos os objetivos.

O primeiro passo do AEMMT é gerar as tabelas e preenchê-las com soluções aleatórias. Em seguida, inicia-se o laço principal, onde em cada iteração são escolhidas duas tabelas vai torneio duplo de acordo com suas pontuações. A pontuação tem valor inicial zero e sempre que uma tabela gera um filho que sobrevive para a geração seguinte, sua pontuação é incrementada. As pontuações são zeradas a cada 100 gerações. Considerando as duas tabelas que vencem os torneios, sorteia-se um indivíduo de cada e efetua-se o cruzamento entre os dois. O filho gerado é então comparado tabela à tabela, entrando naquelas em que representar uma melhoria. Após a execução de todas as gerações, espera-se que a população da tabela extra de não-dominância tenha convergido para a fronteira de Pareto.

3.2.4 Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias (AEMMD)

O AEMMD é uma modificação do AEMMT que, apesar de usar o mesmo processo de divisão do problema multi-objetivo, abandona a ideia de escalarização e volta a utilizar o conceito de dominância dos métodos mais antigos (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

Figura 3 – Tabelas do AEMMD



mão da relação de dominância de Pareto, um indivíduo novo s só entra na tabela t, se s não for dominado por nenhuma solução em t considerando apenas os objetivos de t. Além disso, se s entra em t, todas as soluções em t dominadas por s são removidas.

O primeiro passo do AEMMD é gerar o conjunto de tabelas, que é composto por 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 apenas para a partir de dois valores. Diferentemente do AEMMT, as tabelas não possuem limite de tamanho e podem crescer indefinidamente. Para um problema de quatro objetivos (f_1, f_2, f_3, f_4), por exemplo, 11 tabelas seriam geradas, veja a figura 3.2.4.

Com as tabelas criadas, gera-se um número pré-definido de soluções aleatórias, distribuindo-nas pelas tabelas de acordo com a relação de dominância de Pareto. O próximo passo é o laço principal, onde a cada iteração, através de um torneio duplo, sorteia-se duas tabelas de acordo com suas pontuações. A pontuação das tabelas no AEMMD é diferente do AEMMT, ao invés de conceder um ponto sempre que se gera um indivíduo sobrevivente, pontua-se quando uma tabela recebe um indivíduo. Tendo escolhido as duas populações pais, sorteia-se um representante de cada e gera-se um único filho, o qual é comparado tabela à tabela e entra naquelas onde representa uma solução não dominada. Outra diferença em relação ao método original, é que o AEMMD não reinicia as pontuações em momento algum do algoritmo. Espera-se que ao final das gerações, a população da tabela principal, com todos os objetivos, tenha convergido para a fronteira de Pareto.

3.3 Algoritmos many-objectives baseados em colônias de formigas

A maior parte dos métodos de busca multiobjetivo são baseados em algoritmos genéticos, mas uma boa alternativa pouco explorada é a inteligência coletiva, representada por

estratégias como as colônias de formigas (ACOs) e enxame de partículas (PSOs). Neste trabalho, devido ao fato de explorar-se dois problemas discretos (problema da mochila multiobjetivo e problema do roteamento multicast), optou-se por utilizar as colônias de formigas, que foram desenvolvidas especialmente para lidar com esse tipo de problema. Como explicado em [secao_formigas], ao invés de utilizarem os operadores genéticos para gerarem e evoluírem a população, os ACOs lançam mão de um processo de construção de solução baseado em feromônios, o qual decide, a cada passo, uma partícula da solução de acordo seu valor de feromônio e heurística. Dentre os ACOs multiobjetivos propostos na literatura, destacam-se o MOACS, ...

3.3.1 Multi-Objective Ant Colony Optimization Algorithm (MOACS)

O MOACS foi proposto pela primeira vez em [Baran 2003] para o problema de roteamento de veículos com janelas de tempo e posteriormente foi aplicado no problema do roteamento multicast [Baran 2005]. A última versão do algoritmo foi proposta em [Baran 2016] e essa é a variação utilizada neste trabalho. O MOACS 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. Veja o código do algoritmo 3.

Algoritmo 3 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 da solução depende do problema e, no MOACS, os principais componentes para o 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;

- 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 peso, distância e tráfego na aresta.

- Peso máximo de uma heurística (w_{max}): representa o valor máximo que o peso de uma heurística pode atingir. Em [Baran 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 é gerado aleatoriamente em cada iteração. Cada função de heurística recebe um peso variando no intervalo $[0, w_{max}[$.

Ao construir uma solução, utiliza-se o mesmo processo de decisão do ACO original, explicado na seção [secao aco]. A única diferença é que as múltiplas heurísticas do MOACS (H) deve ser unificada em uma única função $h(x)$, para isso utiliza-se o vetor de pesos W para se aplicar uma média ponderada. Veja a equação a seguir:

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

Em cada época (iteração do laço principal), atualiza-se o arquivo de soluções não dominadas com as novas soluções geradas. Se o arquivo foi atualizado após criar-se todas as soluções, reinicia-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 seja não-dominada, atualiza-se as quantidades de feromônio na estrutura de acordo com as soluções no arquivo.

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

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

Onde:

- ρ: coeficiente de evaporação;

- $\Delta\tau(s)$: Quantidade de feromônios depositados pela solução s.

A quantidade de feromônio depositado pela solução s ($\Delta\tau(s)$) é definida por:

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

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, para problemas de maximização, basta inverter a equação. Se os objetivos são reduzir o custo, o tráfego e o delay de uma rede, por exemplo, $performance(s) = custo(s) + trafego(s) + delay(s)$.

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

Capítulo 4

Problemas de teste

São vários os problemas em que se pode aplicar os algoritmos multiobjetivos e pode-se dividi-los em duas categorias: contínuos ou discretos. O comportamento e até a própria possibilidade de se aplicar o algoritmo depende dessa classificação. A fim de testar os métodos multiobjetivos, normalmente se utiliza problemas de teste, dentre os contínuos, destacam-se: SCH [], FON [], POL [], KUT [] e ZDT []. Os problemas contínuos são funções contínuas e não necessariamente representam um problema real. Os problemas discretos, por outro lado, possuem um enunciado bem definido e nem todas as soluções possíveis são válidas, ou seja, existem buracos no contradomínio das funções. Exemplos de problemas discretos comummente usados na literatura multiobjetivo são: cacheiro viajante, roteamento de veículos com janelas de tempo, problema da mochila, sequenciamento de proteínas e problemas de roteamento em redes. Neste trabalho focou-se em dois problemas discretos: o problema da mochila multiobjetivo (PMM) e o problema do roteamento multicast (PRM).

4.1 Problema da mochila multiobjetivo

O problema da mochila (PM) é um problema teórico muito conhecido na computação e geralmente utilizado para se introduzir o conceito de otimização. Apesar disso, existem problemas reais equivalentes que podem ser resolvidos com as mesmas técnicas, como o escalonamento de tarefas em um sistema operacional.

O problema da mochila consiste em arranjar um conjunto de itens em uma mochila de forma a não exceder a capacidade da mesma e ao mesmo tempo maximizar o valor (lucro) dos objetos carregados. Matematicamente, dado 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 $lucro(O_i)$, encontrar o conjunto $S \subset O$, tal que $\sum_{o \in S} peso(o) <= C$ e $\sum_{o \in S} lucro(o)$ seja o maior possível.

Existem diversas estratégias para se resolver o problema da mochila, dentre elas as mais usadas são algoritmos gulosos e algoritmos genéticos. Uma coletânea de algoritmos gulosos

para o PM são explicados, implementados e analisados em [Bracis 15]. Em [Bracis 21], um AG é proposto e se demonstra o potencial da estratégia para a resolução de problemas de otimização NP-completos com restrições. Apesar de existirem múltiplas estratégias, para se resolver o problema, os algoritmos gulosos são os mais rápidos e eficientes para resolver o PM mono-objetivo, em contra-partida, a complexidade adicionada pela versão multiobjetivo do problema inviabiliza a utilização dos mesmos, tornando os AG's e demais métodos bio-inspirados as melhores opções.

O problema da mochila multiobjetivo (PMM) é similar ao original, 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 escalar, onde cada componente representa o valor do item O_i em um dos objetivos. Por exemplo, no PMM de 3 objetivos, cada $O_i \in O$ possui um vetor tri-dimensional de lucros. O objetivo do problema passa a ser maximizar todos os lucros ao invés de um único valor.

O PMM já foi utilizado várias vezes para avaliar algoritmos multi-objetivos, podendo-se destacar os trabalhos de [SPEA] [SPEA2] [MOEA/D].

4.2 Problema do roteamento multicast (PRM)

O problema do roteamento multicast aparece na engenharia de tráfego em redes de computadores e consiste em transmitir uma mensagem multicast. Uma transmissão de rede pode ser do tipo unicast, multicast ou broadcast. Em transmissões unicast conecta-se um ponto da rede a um outro ponto qualquer, para fazer isso de forma eficiente basta encontrar o melhor caminho entre os dois pontos. 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, basta verificar a árvore geradora de custo mínimo. As transmissões multicast desejam, a partir de um nó da rede, transmitir o conteúdo para alguns outros, o que apresenta maior complexidade, 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 [1-plano].

O PRM é um problema além de ser um problema prático, é muito importante, pois significaria um grande avanço na geração de rotas em redes de computadores, proporcionando uma comunicação mais rápida, menos custosa e mais confiável entre dispositivos, o que é de essencial em uma era onde a maioria das pessoas consomem informação e entretenimento pela Internet. Dado que deseja-se transmitir um conteúdo via uma rede de computadores, o problema consiste em encontrar a melhor rota possível entre a fonte de dados e o destino. Matematicamente, dado uma rede representada pelo grafo $G = (V, E)$, um nó raiz $s \in V$ (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. Veja o exemplo da figura 4.2.

Figura 4 – Exemplo de rede retirado de [1-plano]

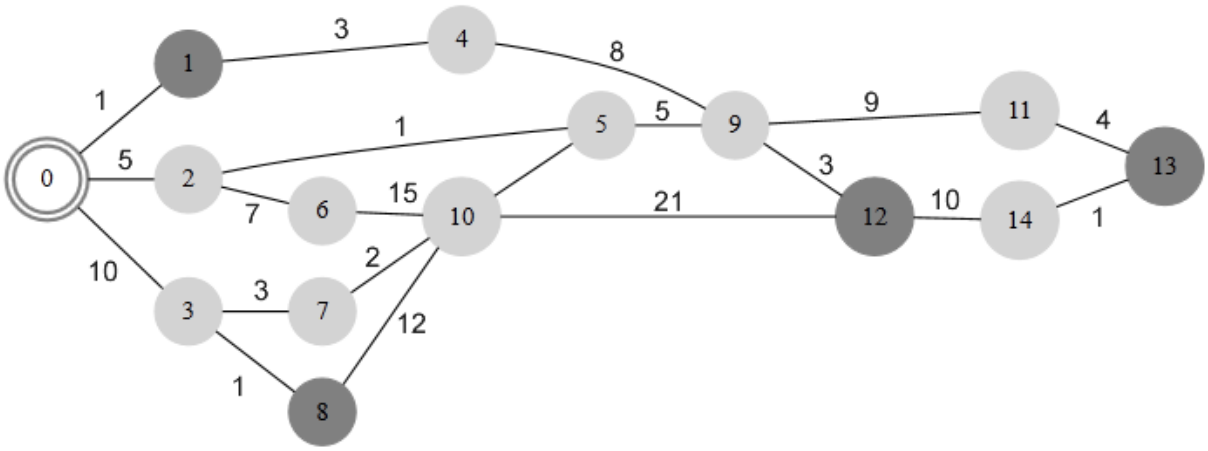


Figura 5 – Exemplos de árvores multicast relativos ao grafo da figura 4.2. Retirado do trabalho de [2-plano]

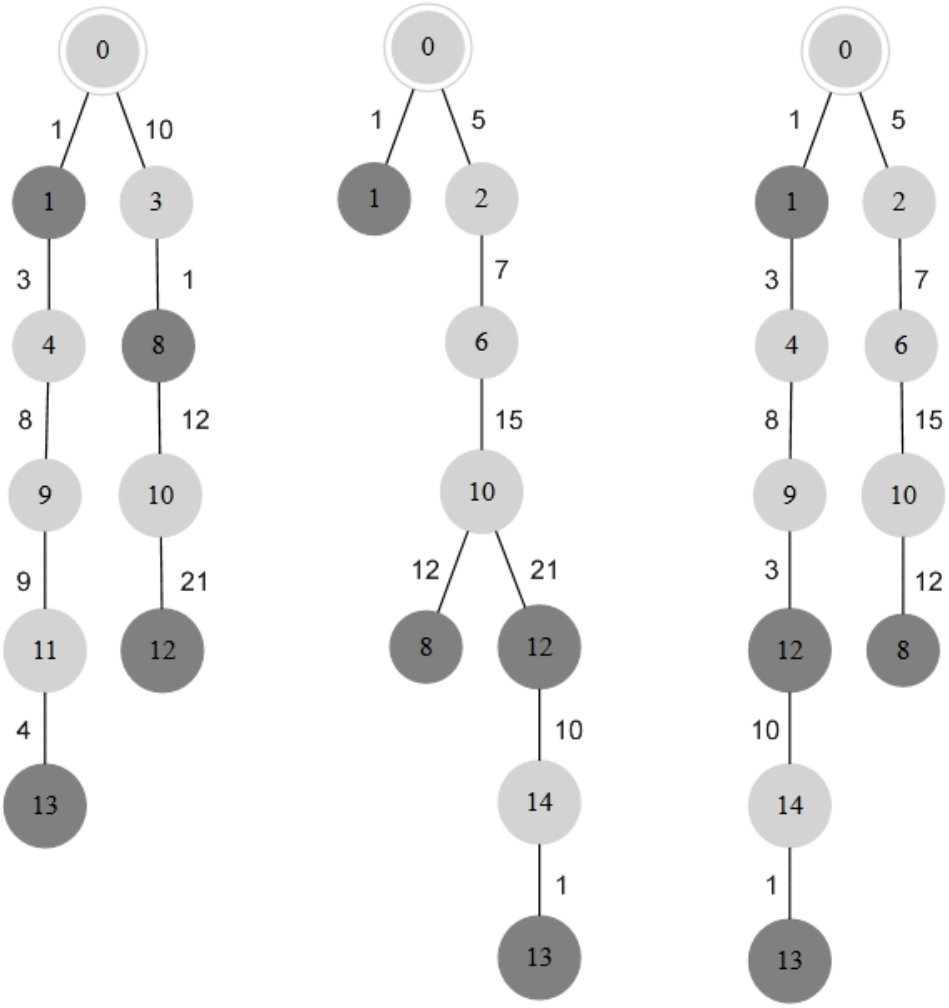
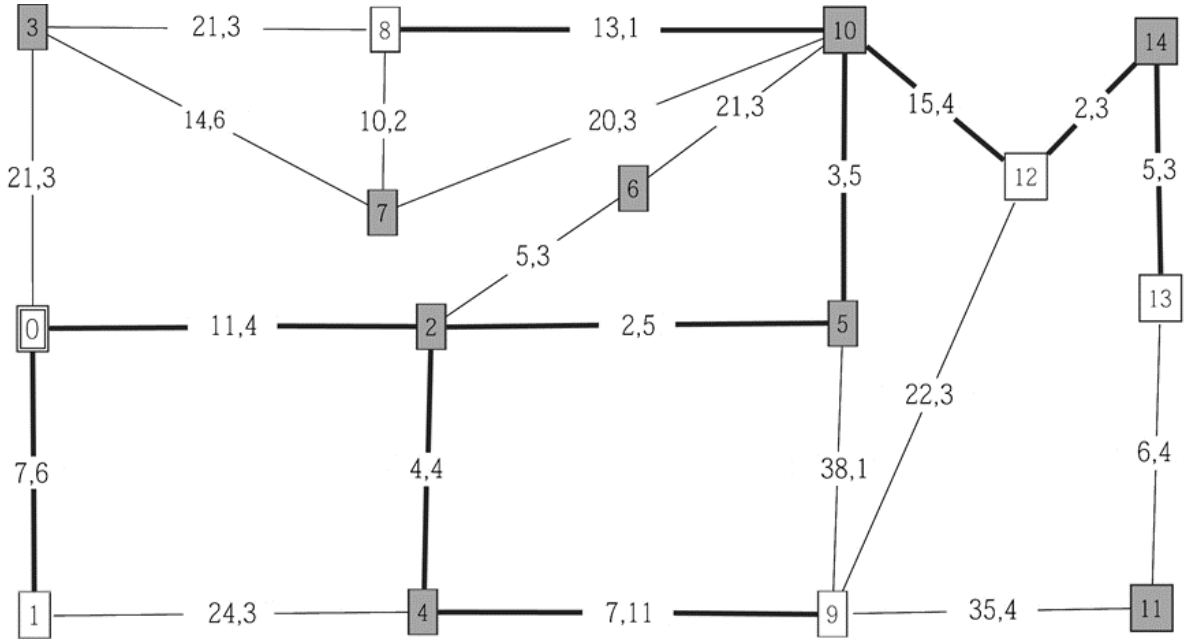


Figura 6 – Exemplo de árvore multicast no PRM multiobjetivo



Na figura 4.2 apresenta-se exemplos de árvores multicast criadas a partir do grafo mostrado na figura 4.2 considerando a raiz (r) como sendo o vértice 0 e os nós destinos (D) igual a 1, 8, 12, 13. 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, mas o intuito deste trabalho é utilizar uma versão mais realista do problema. A qualidade de um enlace de rede não pode ser medida através de uma única métrica, um custo genérico não é capaz de dizer se um link é bom ou ruim, características como distância, delay, capacidade de tráfego e uso do tráfego são melhores indicadores, portanto, propõe-se como objeto de estudo deste trabalho, o problema do roteamento multicast multiobjetivo. Nesta versão do problema, as árvores apresentadas como solução devem representar o melhor compromisso entre as métricas utilizadas. Veja um exemplo para a otimização de "custo" e "delay" na figura 4.2.

Na figura 4.2 apresenta-se um exemplo de rede com as métricas "custo" (primeiro valor) e "delay" (segundo valor) nas arestas. As arestas em negrito representam uma árvore multicast ótima (não-dominada) para o seguinte conjunto de objetivos:

1. Minimizar custo total: soma dos valores de custo para cada aresta da árvore;

2. Maximizar delay fim-a-fim atendidos: número de ramos da árvore em que a soma dos delays nas arestas não ultrapassa um valor d_{max} pré-definido, neste caso 25. Em outras palavras, quantidade de conexões cliente-servidor que mantém limite aceitável de atraso.

Neste trabalho considera-se até quatro valores de peso para um enlace rede: custo,

delay, capacidade de tráfego e tráfego corrente, representados nas fórmulas a seguir respectivamente pelas funções: $c()$, $d()$, $z()$ e $t()$. Através dessas medidas são formulados os seguintes objetivos:

1. Custo total: soma dos valores de custo para cada aresta da árvore;
2. Delay fim-a-fim atendidos: número de ramos da árvore em que a soma dos delays nas arestas não ultrapassa um valor d_{max} pré-definido;
3. Delay total: soma dos valore de delay para cada aresta da árvore;
4. 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;
5. Delay fim-a-fim máximo: maior valor para a soma de delays dentre todos os ramos da árvore;
6. Hops count: número de vértices na árvore;
7. 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, $\max_{e \in E} \frac{t(e)+\phi}{z(e)}$;
8. Utilização média dos enlaces: média entre a utilização de banda entre todas as arestas da árvore. Similar à definição anterior.

Afim de possibilitar diversos cenários de teste para o PRM, os objetivos acima podem ser combinados de diversas maneiras, criando vários ambientes multi-objetivos. Tais combinações são exploradas na seção de experimentos.

Capítulo 5

Estratégias evolutivas para o PMM

Uma das partes mais importantes de um algoritmo de busca bio-inspirado é a modelagem da solução. Para os algoritmos genéticos é necessário definir a representação da solução, a geração aleatória de indivíduos, o cruzamento e a mutação. Para os ACOs, é possível utilizar a mesma representação de solução do AG, mas deve-se desenvolver um algoritmo que constrói a solução a partir de uma estrutura de feromônios e uma heurística.

A implementação de um AG para o problema da mochila mono-objetivo é trivial, pois a solução é representada por um vetor binário e a literatura está repleta de exemplos que podem ser resolvidos dessa maneira []. Um AG para o problema da mochila original pode ser encontrado em []. A versão many-objective do problema não requer nenhuma modificação no modelo, fazendo com que o mesmo processo de cruzamento e mutação possam ser utilizados. No entanto, a implementação de um ACO para o mesmo problema pode ser desafiador, já que as colônias de formigas esperam trabalhar com grafos e não arrays de bits. Um estudo extensivo relativo ao uso de ACOs para a resolução do problema da mochila com ACOs é apresenta em []. Nas seções a seguir detalhe-se a representação da solução, os operadores genéticos e a construção de soluções para o PMM usados neste trabalho.

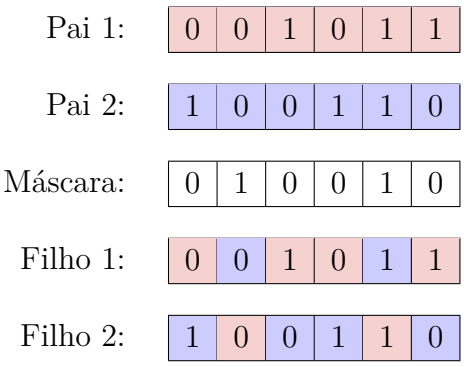
5.1 Representação da solução

Em ambas as estratégias bio-inspiradas, AGs e ACOs, uma solução para o PMM é representada da mesma maneira: 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 onde o valor é 0 dizem que aquele item não será incluído na mochila, enquanto as posições que valem 1, dizem que o item será incluído na mochila. E.g. se a solução é representada pelo vetor [1,0,0,1,0,1,1,0,0,0], apenas os itens 0, 3, 5 e 6 serão colocados na mochila, os outros ficarão de fora.

5.2 Cruzamento e mutação (AGs)

O cruzamento entre duas soluções binárias, como explicado em [ref. secao ags], 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. Veja o exemplo da figura 5.2.

Figura 7 – Exemplo de crossover uniforme



Como pode ser visto na figura 5.2, o crossover uniforme pode ser implementado com uma máscara, que é um vetor aleatório de bits que controla os genes herdados de cada filho. Se o bit na posição i da máscara vale 0, então o filho na posição i herda o valor do pai 1, caso contrário, o pai 2 fornece o valor. Dessa forma, ainda é possível gerar 2 filhos, um com a regra de que o bit 0 da máscara representa o pai 1 e o bit 1 representa o pai 2 (filho 1 na imagem), e outro com a regra inversa (filho 2).

Após o crossover, existe uma chance definida pelo AG de se mutar a solução. A mutação utilizada para o PMM foi o processo mais simples possível para vetores binários, a inversão de bit: sorteia-se uma posição aleatória do vetor, se o valor for 0, troca-se para um, caso contrário, troca-se para 0.

5.3 Construção da solução (ACOs)

As colônias de formigas foram propostas inicialmente para problemas em grafos, portanto, soa contra-intuitivo utilizá-las para o problema do mochila. Mas, como mostrado em [mkp-aco-ke], não é necessário ter um grafo para se utilizar a técnica, é possível manipular os feromônios de diversas outras formas. Para o PMM, a literatura traz três principais formas de lidar com o feromônio:

1. Depositar feromônios em cada um dos itens. Sempre que se escolher um objeto para compor a solução, incrementa-se a quantidade de feromônios nele presente. Dessa forma, a quantidade de feromônio em cada item representa a preferência para se escolhê-lo em relação aos demais. [Leguizamon and Michalewicz 1999].

2. Criar um grafo direcionado que representa a preferência de se incluir um item b logo após ter incluído um item a. 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 incluído e todas as arestas no grafo que partem dele, o destino com a maior quantidade de feromônios representa o item preferível. [Fidanova 2002].

3. A terceira estratégia proposta por [Alaya et al. 2004] utiliza um conceito semelhante a ideia anterior, mas ao invés de depositar feromônios apenas em pares consecutivos, os deposita para todos os pares de objetos existentes na solução. Por exemplo, se na solução os itens a, d e f já foram incluídos na mochila e no passo corrente 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 adicionado. Assim, ao construir a solução, deve-se analisar todas as arestas com origem em algum objeto já presente na solução, o destino da aresta com maior quantidade de feromônios representa o item mais desejável.

Neste trabalho utilizou-se a estratégia 1, os feromônios são depositados nos items. Além dos feromônios, outra importante fonte de informação para se construir uma solução no ACO é a heurística, que estima o quão vantajoso é escolher um item em relação a outro. Tomando como inspiração o estudo de [mkp-aco-ke], propôs-se como heurísticas 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: $h_k(item, cr) = lucro_k(item) * (1 - peso(item)/cr)$.

- Uma heurística extra é usada exclusivamente para se referir ao peso do item: $h_{peso}(item) = 1 - peso(item)/peso\_maximo$.

Logo, o número de heurísticas no PMM é sempre o número de objetivos + 1. Tendo definido a estrutura de feromônios e as heurísticas, cabe ao algoritmo baseado em colônia de formigas construir a solução.

Capítulo 6

Estratégias evolutivas para o PRM

A modelagem de um algoritmo genético para o problema do roteamento multicast não é trivial, pois a solução não pode ser representada por um vetor. De fato, como deve representar os caminho entre o servidor e os múltiplos destinos em uma rede de computadores, a solução para o PRM é uma árvore. Dessa forma, é preciso desenvolver o processo de crossover e de mutação de acordo com a estrutura. O cruzamento deve receber duas árvores e gerar uma nova que compartilha características de ambos os pais, enquanto a mutação precisa criar uma pequena alteração na árvore que permita melhor explorar o espaço de busca, mas que não a descaracterize completamente.

Em contrapartida, o PRM se aproxima da definição original do ACO, pois trabalha com grafos. A diferença está no fato de que a solução é representada por uma árvore ao invés de um simples caminho, fazendo com que o depósito de feromônio e a escolha de arestas sejam desenvolvidas conforme a nova estrutura.

6.1 Representação da solução

Como mostrado na seção [seção do prm], 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 a solução:

1. Representação em árvore: o AG evolui a própria árvore que se deseja encontrar como solução. É um processo mais complicado que a alternativa a seguir, mas nenhum pós-processamento é necessário.

2. Representação em conjunto: o AG evolui um conjunto de caminhos C, ou seja, para cada nó destino d no problema deve existir uma lista de nós $L \in C$ que contém a sequência de nós, onde o primeiro elemento é o servidor e o último é d. A representação em conjuntos é mais fácil de se gerenciar, mas exige a transformação

em árvore ao final do processo. Como diferentes árvores podem ser formadas a partir de um 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 1, árvores.

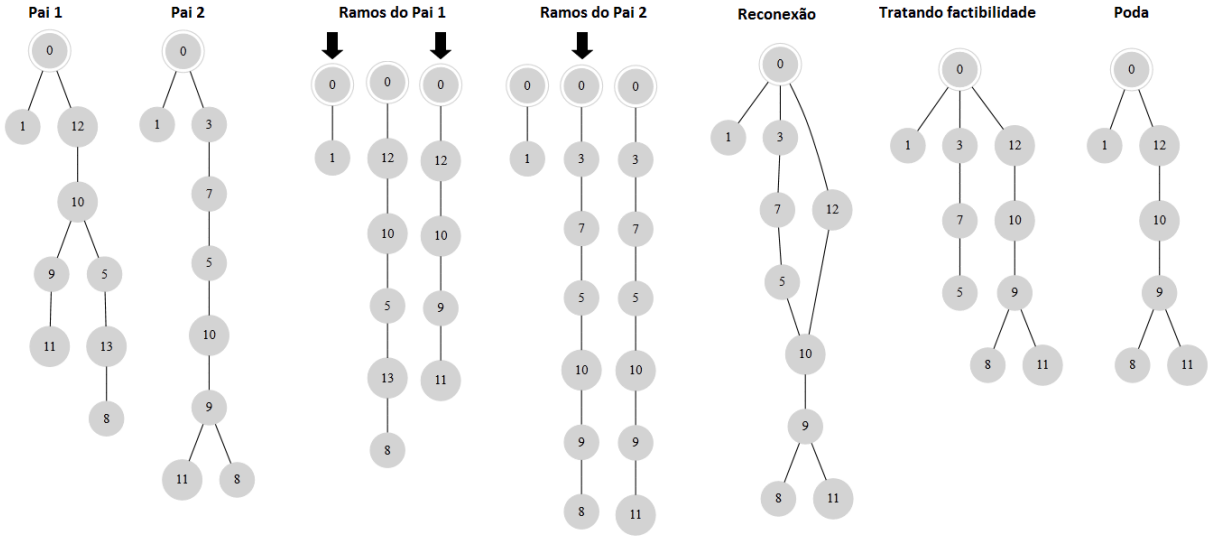
6.2 Inicialização dos indivíduos

6.3 Cruzamento (AG)

O modelo de cruzamento para o PRM utilizado neste trabalho é chamado de cruzamento por caminho e foi proposto em [dissertacao do Fialho] como uma alternativa ao cruzamento por similaridade utilizado em trabalhos anteriores [Bueno e Oliveira, 2010].

O cruzamento por caminho 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 do PRM, acrescentar a F ou o ramo de P_1 que leva a d, ou o ramo de P_2. A escolha entre P_1 e P_2 é feita de forma aleatória. Assim que todos os nós destinos são atingíveis em F, para-se a seleção de ramos, remove-se os ciclos que possivelmente foram incluídos e poda-se a árvore, excluindo qualquer nó folha que não seja um destino.

Figura 8 – Exemplo de cruzamento por caminho



A figura 6.3, retirada do trabalho de [dissertacao Fialho], representa o processo de cruzamento por caminho entre duas árvores (Pai 1 e Pai 2). No exemplo, o nó raiz (servidor) é o vértice 0 e os 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, e como foi gerado um ciclo, deve-se removê-lo afim de obter-se uma árvore

válida. Em "tratando factibilidade" apresenta-se o filho após a remoção dos ciclos, como o nó folha "5" não é um destino, deve-se podá-lo, resultando na última árvore, "poda", que é o filho retornado pelo processo.

Durante o processo de cruzamento por caminho, para remover os ciclos, percorre-se a árvore em largura removendo qualquer aresta que adicione ciclo. No processo de poda, verifica-se todas as folhas, se alguma não for um destino, remove-se o nó, repetindo o processo até que todos os nós folhas sejam destinos.

6.4 Mutação (AG)

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. Veja o algoritmo 4.

Algoritmo 4 Mutação para uma árvore $(A, G, qte_{arestas}, r, D)$

1: Selecione aleatoriamente $qte_{arestas}$ e remova-as de A
2: Retire de A a componente conexa que contém a raiz e chame-a de C
3: Crie um grafo vazio M para guardar o resultado da mutação
4: Adicione todas as arestas de C a M
5: **enquanto** $|D| > 0$ **faça**
6: Selecione aleatoriamente um destino $d \in D$ e remova-o da lista D
7: Remova de A a componente conexa correspondente ao destino d e coloque-a em C
8: **se** M não possui o vértice d **então**
9: Tendo G como referência, crie um caminho aleatório P entre M e a componente conexa C
10: Adicione todas as arestas de P a M
11: **fim se**
12: Adicione todas as arestas de C a M
13: **fim enquanto**
14: Remova os ciclos em M, caso existam
15: Pode a árvore M, removendo todos os nós folhas que não são destinos
16: **retorne** M

No algoritmo 4 recebe-se como parâmetros a árvore a se mutar (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 7, se não existe componente conexa com o vértice d, C será um grafo com um único nó (d) e nenhuma aresta. Na linha 9, o caminho aleatório é construído nó a nó até se encontrar uma sequência de arestas entre as duas componentes. Ao final, o mesmo pós-processamento do cruzamento por caminho é realizado: remove-se os ciclos e poda-se a árvore.

6.5 Construção da solução (ACO)

O processo de construção da solução por um ACO no 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 [Baran PRM], 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 (E) que contém um único elemento: a raiz. Sorteia-se um vértice $i \in E$ e atribui-se probabilidades a todas as arestas (i, j), onde j é qualquer vértice não-visitado na vizinhança de i. i é removido de E caso não possua vizinhos factíveis. Um vértice v é escolhido de acordo com as probabilidades calculadas, a aresta (i, v) é adicionada à solução e v é inserido na lista de exploração E. 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 solução pode ser vista como os caminhos entre a raiz e cada um dos destinos, portanto, outra estratégia é imaginar que $|D|$ formigas partirão da raiz e cada uma deve encontrar um vértice $d \in D$ diferente, onde D é o conjunto de nós destinos. Com cada um dos caminhos em mãos, monta-se a árvore com o cuidado de não se incluir ciclos. O último passo é a poda, que exclui qualquer vértice folha que não seja um destino.

3. Uma terceira estratégia, proposta neste trabalho, representa uma formiga fictícia que pode estar em vários locais do grafo ao mesmo tempo. 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 tende o resultado 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 (E) que contém todas as arestas com uma extremidade na raiz. A cada passo, calcula-se as probabilidades para toda aresta de E e, de acordo com os valores obtidos, escolhe-se $e \in E$ para compor a solução. e é então removido de E, adicionado à solução e tem todas as arestas com que compartilha um vértice 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 algoritmo.

4. Uma última estratégia é fazer um processo inverso para se construir a solução, ao invés de partir da raiz e chegar nos destinos, este modelo propõe que se utilize $|D|$ formigas, onde cada uma tem como posição inicial um destino $d \in D$ diferente. D é o conjunto de vértices destinos. 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 meio 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 linguagem matemática, 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. Em cada componente, calcula-se as probabilidades das arestas na vizinhança do nó sendo explorado e escolhe-se, de acordo com os valores obtidos, uma novo vértice v para ser adicionado à componente. Se não há arestas factíveis na vizinhança, deve-se rebobinar a exploração para um vértice anterior. Caso o nó incluído v pertença a uma outra componente conexa, une-se as duas estruturas. O processo termina quando existe apenas uma componente conexa e o vértice raiz foi encontrado. Uma poda é realizada como pós-processamento da árvore.

Cada uma das estratégias mencionadas foi implementada e testada a fim de determinar aquela que produz as soluções de melhor qualidade. O PRM com apenas um objetivo foi o problema escolhido para avaliá-las. A fim de obter um valor como parâmetro de qualidade, também foi implementada uma versão modificada do algoritmo de Prim que aproxima a árvore multicast de menor custo. Como esperado, o algoritmo de Prim é uma opção melhor que o ACO quando se trata do PRM mono-objetivo, mas a intenção deste estudo não foi produzir um algoritmo melhor para o problema, mas sim testar as estratégias de construção de solução com o fim de utilizá-las em versões mais complexas (multi-objetivas) do PRM.

Os resultados do testes, que podem ser encontrados no apêndice [], mostraram que a estratégia número 3 representa a melhor relação entre qualidade do resultado e tempo de execução. Portanto, para todos os experimentos no PRM da seção [], esse foi o método de construção da solução utilizado.

Na lista acima, a estratégia número 3, escolhida como a melhor forma de se construir uma solução para o PRM no ACO, é explicada de maneira geral e omite alguns detalhes que são expressos no algoritmo 5. A ideia principal se mantem a mesma, mas se implementada da maneira como foi escrita, se torna um processo muito lento. É possí-

vel agilizar o algoritmo, sem afetar muito a qualidade das soluções, através de algumas técnicas de amostragem.

Algoritmo 5 Geração de solução no ACO $(G, r, D, \tau, h, E'_{size})$

1: Inicie uma árvore vazia T
2: Inicie E 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 E' a partir da lista E com E'_{size} elementos
6: Calcule as probabilidades de todas as arestas em E' de acordo com τ e h
7: Escolha uma aresta $e = (i, j) \in E'$ 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 E que leva a v **então**
13: Remova a de E
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 E
16: **senão se** v não tiver sido visitado **então**
17: Inclua (j, v) em E
18: **fim se**
19: **fim para**
20: **fim enquanto**
21: Pode a árvore T
22: **retorne** T

O Algoritmo 5 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;
- E'_{size}: tamanho da amostra.

Trabalhar com todas as arestas possíveis é demasiadamente caro e inviável para um algoritmo em que se deseja boa performance em termos de tempo de execução. Por isso, trabalha-se com uma amostra da lista de exploração (linha 5 do algoritmo). Também a fim de se evitar o crescimento de E, nas linhas 12 a 16, caso um novo vértice descoberto já seja atingível a partir de alguma aresta em E, mantém-se em E apenas uma das arestas. Para escolher qual das arestas manter, calcula-se as probabilidades de acordo com os feromônios (τ) e a heurística (h).

Capítulo 7

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 Decomposed Pheromone* (MACO/D), em português, otimização em colônia de formigas para muitos objetivos baseada em decomposição de feromônios. A proposição envolve os seguintes módulos:

- Framework ACO: algoritmo geral desenvolvido para se trabalhar com qualquer problema discreto;
- Modelo PMM: construção da solução para o problema da mochila multiobjetivo, apresentado na seção [seção pmm-aco];
- Modelo PRM: construção da solução para o problema do roteamento multicast, apresentado na seção [seção prm-aco].

Esta seção apresenta o módulo principal do algoritmo, o framework. O MACO/D se baseia na ideia do AEMMD de se criar tabelas de dominância para diferentes combinações possíveis de objetivos, adaptando-na para um modelo de colônia de formigas.

A multiplicidade de objetivos impõe que algumas mudanças sejam feitas na ideia original do ACO. Como representar os diferentes objetivos no depósito de feromônio? Como compor as múltiplas heurísticas em uma única função? De acordo com [Alaya], essas questões podem ser resolvidas através de uma das seguintes opções:

1. $(m + 1, m)$: considerando m o número de objetivos, este 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, a cada passo, sorteia-se alguma das estruturas de feromônios para se utilizar. 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 as heurísticas, cada colônia utiliza

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

2. $(1, 1)$: este modelo utiliza apenas uma colônia de formigas e uma estrutura de feromônios. Assim como no modelo 1, as heurísticas são somadas para montar uma solução. A única estrutura 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 para arquivo, já que, de fato, são indiferenciáveis em termos de qualidade.

3. $(1, m)$: considerando m o número de objetivos, este modelo utiliza 1 colônia de formigas e m estruturas de feromônios. Assim como nos outros 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 época conforme a melhor solução encontrada para o mesmo.

Em [Alaya], os experimentos mostraram que o terceiro modelo $(1, m)$ gera os melhores resultados quando aplicado ao problema da mochila multiobjetivo. Dentre os ACO's vistos na seção [secao aco-multiobjetivo], o MOACS se encaixa na segunda categoria, adaptando apenas o modo de lidar com a heurística. (...). O algoritmo aqui proposto, não pertence a nenhuma das categorias apontadas por Alaya, mas se assemelha a proposição número três, pois lida com uma única colônia e múltiplos feromônios. A grande diferença é que o número de estruturas de feromônio não é igual a quantidade de objetivos, mas sim ao número de combinações de objetivo possíveis, de forma semelhante ao que ocorre no AEMMD. Além disso, o processo de cálculo da heurística e de atualização dos feromônios se difere substancialmente da ideia original do algoritmo $(1, m)$.

O MACO/D, se baseia na ideia de decomposição, ou seja, ao invés de trabalhar-se diretamente com todos os objetivos, o problema é atacado em várias frentes com quantidades reduzidas de funções. Desta forma, evita-se o problema onde não é possível classificar a população devido ao alto número de soluções não-dominadas em espaços de dimensionalidade alta.

O primeiro passo do MACO/D é criar as estruturas de feromônio $P = \{p_1, p_2, ..., p_i, ..., p_n\}$, uma para cada combinação possível de objetivos. Para um problema de seis objetivos, por exemplo, são criadas 57 estruturas ($|P| = 57$). Cada $p_i \in P$ é responsável por um subproblema e guarda as seguintes informações:

- Valores: os valores dos feromônios em sí. São inicializados com o menor valor possível;

- Objetivos: determina os objetivos do subproblema em questão. É um vetor binário de tamanho m (número de objetivos), onde cada posição 1 representa um objetivo que faz parte do problema e 0 que não faz parte;

- 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).

As estruturas de feromônios são utilizadas no processo de construção da solução e devem ser atualizadas em toda iteração do algoritmo. trabalhar com totalidade da lista P (57 estruturas, no caso de seis objetivos) é computacionalmente caro e inviável. Por essa razão, define-se um número máximo de estruturas para se trabalhar em um dado momento. Em ambas as aplicações do MACO/D utilizadas nesta pesquisa (PMM e PRM), foi utilizado um limite de cinco estruturas de feromônios ativas simultaneamente. A ordem de ativação dos elementos de P é determinada pela própria ordem dos itens em P, 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 instancia-se os subproblemas de 3 objetivos e assim por diante, até que a última estrutura de feromônios com o número total de objetivos seja criada.

Inicialmente, os cinco subproblemas mais simples são ativados ($P_{ativo} = \{p_1, p_2, ..., p_5\}$). Assim que algum deles passa a não contribuir satisfatoriamente para o conjunto de soluções, é desativado em favor do próximo problema mais complexo ainda não utilizado. Dessa forma, o conjunto de soluções do MACO/D cresce gradualmente, partindo das decomposições mais simples em direção às mais complexas. O desafio principal em uma otimização com muitos objetivos é diferenciar soluções não dominadas e é nesse ponto que se utiliza o auxílio da decomposição. Em cada iteração do laço principal, após gerar o conjunto de soluções S, extrai-se todas as que são não-dominadas de acordo com a totalidade dos objetivos e partir desse subconjunto de soluções S_{nd}, alimenta-se as cinco estruturas ativas em P_{ativo}. A distribuição das soluções não-dominadas S_{nd} entre as estruturas ativas se dá de acordo com o algoritmo 6.

O laço principal, que consiste em gerar soluções e atualizar as estruturas de feromônios ativas, termina quando o número máximo de iterações é atingido. Se todo $p \in P$ se torna ativo antes do término do programa, de maneira circular, volta-se a explorar o início da lista. O valor máximo de convergência para cada estrutura utilizado neste trabalho foi 10. Ao final, retorna-se como solução o arquivo de p_n, ou seja, todas as soluções não-dominadas encontradas para o problema completo de m objetivos. O algoritmo 7 apresenta uma visão geral do MACO/D.

Algoritmo 6 Distribuição de soluções não dominadas entre as estruturas de feromônios

1: **para** $a \in P_{ativo}$ **faça**
2: atualize o arquivo de a com as soluções em S_{nd}
3: crie um conjunto A com as soluções de S_{nd} que foram adicionadas ao arquivo
4: crie um conjunto R com as soluções do arquivo que foram removidas
5: **se** |A| > 0 **então**
6: incremente os valores de feromônio de a de acordo com as soluções em A
7: decremente os valores de feromônio de a de acordo com as soluções em R
8: **senão**
9: incremente a convergência de a em 1
10: **se** convergência de a atingiu o valor máximo **então**
11: em P_{ativo}, substitua a estrutura a pelo próximo $p_i \in P$
12: reinicie a convergência de a
13: **fim se**
14: **fim se**
15: **fim para**

Algoritmo 7 Algoritmo geral do MACO/D

1: crie as estruturas de feromônios $P = \{p_1, p_2, ..., p_i, ..., p_n\}$ em ordem crescente de número de objetivos
2: $P_{ativo} \leftarrow \{p_1, p_2, ..., p_5\}$
3: **enquanto** número máximo de iterações não for atingido **faça**
4: construa o conjunto de soluções S de acordo com os feromônios em P_{ativo}
5: atualize o arquivo de p_n com as soluções em S, chame-o de S_{nd}
6: distribua as soluções em S_{nd} entre as estruturas em P_{ativo} (algoritmo 6)
7: **fim enquanto**
8: **retorne** arquivo de p_n

No algoritmo 6, as linhas 6 e 7 fazem parte do processo de atualização de feromônios, explicado mais a frente neste capítulo. No algoritmo 7, a linha 4 representa a construção das soluções, tema da seção seguinte.

7.1 Construção das soluções

As soluções em colônias de formigas são construídas a partir dos feromônios ($Pativo$), 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 (arestas, em caso de grafos e itens em caso de vetores), α determina a importância do feromônio ao tomar uma decisão a respeito da composição da solução, e β representa a importância da heurística.

No MACO/D existem múltiplas estruturas de feromônios 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 feromônios são recebidos pelo processo de construção das soluções através da lista

de estruturas ativas do MACO/D (P_{ativo}), ou seja, 5 conjuntos de feromônios são recebidos. De maneira circular, cada solução utiliza uma única estrutura de feromônios para ser construída, ou seja, a primeira solução é criada a partir do primeiro elemento de P_{ativo}, a segunda a partir do segundo elemento de P_{ativo}, e assim por diante, até a sexta solução que utiliza novamente o primeiro elemento de P_{ativo}.

Com relação às heurísticas, o MACO/D utiliza o mesmo processo proposto em [Baran 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 com os pesos sorteados.

O processo geral para se construir as soluções é exposto no algoritmo 8. No pseudo-código, a primeira posição de um array é 0.

Algoritmo 8 Construção das soluções

1: $S \leftarrow \emptyset$
2: **para** $i \leftarrow 0$ até nº máximo de soluções **faça**
3: sorteie valores entre 0 e 2 (inclusive) para um vetor W de $|H|$ posições
4: defina h como a função a seguir: $h(x) = \sum_{i \leftarrow 0}^{|H|-1} \frac{H[i](x) * W[i]}{\sum\limits_{w \in W} w}$
5: chame de f os valores de feromônio da estrutura $P_{ativo}[i \pmod{|P_{ativo}|}]$
6: gere uma solução s de acordo com os valores de feromônio f, a heurística h, α e β
7: $S \leftarrow S \cup \{s\}$
8: **fim para**
9: **retorne** S

Na linha 6 do algoritmo 8 constrói-se a solução em sí. 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 [secao pmm construcao da solucao]. Para o problema do roteamento multicast, a estratégia de construção da solução foi apresentada em [secao prm construcao da solucao].

7.2 Atualização dos feromônios

A atualização dos feromônios no MACO/D pode ser de dois tipos:

1. Depósito: a partir de uma solução, adiciona-se uma quantidade δ ao feromônio correspondente à cada partícula da solução. No caso de um problema em grafos, por exemplo, para cada aresta da solução, incrementa-se em δ o feromônio da mesma aresta no grafo.

2. Evaporação: similar ao depósito, mais ao invés de incrementar o feromônio em $\Delta\tau$, decrementa-se.

O depósito de feromônio ocorre quando novas soluções não-dominadas são encontradas e, diferentemente da maioria dos algoritmos baseados em ACO, o MACO/D não evapora todos os feromônios em todas as iterações, ao invés disso, o feromônio só é decrementado quando uma solução deixa de ser não-dominada (ver algoritmo 7).

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 estrutura de feromônios em um fator δ. $\delta(e) = (1 - pesos(e)) * \rho$, onde $pesos(e)$ é a média dos pesos normalizados na aresta e, e ρ é a taxa de evaporação, parâmetro do MACO/D. Se s 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 da lista de itens s incrementa o valor correspondente na estrutura de feromônios em um fator δ. $\delta(i) = lucros(i) * (1 - peso(i)/peso\_max) * \rho$, 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 array de feromônios correspondentes aos itens de s devem ser decrementados em δ.

Capítulo 8

Experimentos

Os experimentos realizados neste trabalho compreendem os algoritmos NSGA-II [], NSGA-III [], SPEA2 [], MOEA/D [], AEMMT [], AEMMD [], MOACS e o ACO proposto: MACO/D. Todos os métodos são testados em dois problemas discretos multiobjetivos: o PMM e o 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 objetivos, a topologia da rede (PRM) e a quantidade de itens (PMM) também afetam a complexidade, portanto elaborou-se instâncias diferentes para cada problema: no PRM, considerou-se seis redes de diferentes complexidades e no PMM variou-se a quantidade de itens em 30, 40, 50, 100 e 200.

Os experimentos podem ser divididos em três etapas distintas:

1. Teste dos AG's 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. Esses experimentos compuseram o primeiro artigo gerado neste trabalho intitulado *"A Comparative Analysis of MOEAs Considering Two Discrete Optimization Problems"* [], em português, uma análise comparativa entre algoritmos evolutivos multiobjetivos considerando dois problemas de otimização discretos. 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. As avaliações dos algoritmos são feitas com base em métricas relacionadas ao Pareto conhecido e permitem um melhor entendimento sobre o comportamento dos algoritmos, o que facilita a identificação de pontos fortes e fracos de cada estratégia, ajudando na elaboração de um novo modelo melhor adequado aos problemas em questão (PMM e PRM).

2. Teste dos métodos de otimização many-objective NSGA-III, MOEA/D, AEMMT, AEMMD e do algoritmo proposto, MACO/D, nos problemas da mochila e do roteamento multicast. São testadas formulações com 4, 5 e 6 objetivos, avaliando-se

3 redes no PRM e problemas com 30, 40 e 50 itens no PMM. Esses experimentos foram responsáveis pela publicação do segundo artigo intitulado *"MACO/D: Many-objective Ant Colony Optimization based on Decomposed Pheromone"* [], em português, MACO/D: otimização por colônia de formigas com muitos objetivos baseada em decomposição de feromônios. Os experimentos colocam à prova pela primeira vez o algoritmo proposto neste trabalho e revelam suas vantagens e fraquezas, possibilitando a identificação dos casos em que se é uma boa ideia utilizá-lo e as formas em que se pode melhorá-lo em cenários onde que os resultados foram desfavoráveis.

3. Teste dos algoritmos de otimização many-objective NSGA-III, MOEA/D, AEMMT, AEMMD e MACO/D em instâncias complexas do PMM e PRM utilizando a métrica hipervolume, independente do Pareto. Esses experimentos são necessários, pois as duas etapas anteriores testaram apenas problemas com complexidade razoável onde se é possível estimar o Pareto. 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 faz necessária a utilização do hipervolume para avaliar os algoritmos. Nesses experimentos são testadas duas novas redes para o PRM e duas novas instâncias do PMM, com 100 e 200 itens.

Considerando as três etapas de experimentos, foram usadas 5 métricas diferentes para se avaliar os algoritmos, são elas:

- Taxa de erro (ER): porcentagem das soluções encontradas que não fazem parte do Pareto aproximado;
- *Generational distance* (GD): distância entre das soluções incorretas para a solução mais próxima no Pareto aproximado. Mede o quão distantes as soluções erradas encontradas estão de uma solução correta.
- *Pareto subset* (PS): número absoluto de soluções encontradas que fazem parte do Pareto aproximado.
- Hipervolume (HV): Única métrica, além do tempo, utilizada que independe de um Pareto pré-conhecido. Mede o volume da figura geométrica m-dimensional (m é o número de objetivos) formada pelas distâncias entre as soluções encontradas e um ponto de referência p_{ref}. 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, extrai-se os 10 resultados e coloca-se 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 p_{ref} para os valores encontrados.

p_{ref} é então utilizado como ponto de referência para o PMM de 5 objetivos e 100 itens. Para os experimentos deste trabalho foi utilizada a implementação oficial do cálculo de hipervolume em [].

- Tempo: tempo em segundos necessário para se executar o algoritmo.

As seções a seguir apresentam os experimentos e seus resultados em cada uma das etapas citadas acima.

8.1 Etapa 1: AG's multiobjetivos

Neste etapa testou-se os algoritmos NSGA-II, NSGA-III, SPEA2, MOEA/D e AEMMT. Ao todo, 30 cenários de teste foram considerados:

- PRM: 5 formulações de objetivos (P_2, P_3, P_4, P_5 e P_6) e 3 redes (R_1, R_2 e R_3). Tanto as formulações quanto às redes foram descritas na seção correspondente ao problema do roteamento multicast [].
- PMM: 5 formulações de objetivos (2 a 6) e 3 instâncias (30, 50 e 100 itens).

Para cada um dos cenários foi extraído um Pareto através de múltiplas execuções dos 5 algoritmos testados. A tabela 1 mostra a cardinalidade de cada Pareto encontrado.

Tabela 1 – Cardinalidade dos Paretos encontrados para a 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 1, a quantidade de elementos nos paretos do PMM é demasiadamente grande para as formulações de objetivos com 100 itens, isso acontece, pois o espaço de busca do problema da mochila cresce exponencialmente com o número de itens. Além disso, a situação é pior quando o número de objetivos é alto, pois, naturalmente, quanto maior a quantidade de funções objetivos, 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, cada rodada de execuções dos algoritmos encontrava novas soluções. Principalmente por essa razão, julgou-se necessária a execução da etapa 3 de experimentos, onde não se usa um Pareto pré-calculado para se avaliar os algoritmos. Apesar de não serem perfeitos, os resultados para o problema de 100 itens ainda são relevantes, pois ainda que os Paretos não sejam estáveis, compara-se as execuções de todos algoritmos

contra as melhores soluções encontradas por todos eles, o que indica o algoritmo com melhor potencial para encontrar boas soluções em problemas multiobjetivos.

Para compilar os resultados através das medidas de desempenho ER, GD e PS, foram feitas 100 execuções de cada algoritmo com os parâmetros listados na tabela **??**. As figuras **??**, **??** e **??** mostram respectivamente os resultados para o PMM de 30, 50 e 100 itens. As figuras **??**, **??** e **??** revelam respectivamente os resultados para o PRM aplicado às redes R_1, R_2 e R_3. Uma análise conjunta, com uma média entre as três instâncias de cada problema é apresenta nas figuras **??** (PRM) e **??** (PMM).

Tabela 2 – Parâmetros utilizados pelos algoritmos no PRM e PMM.

Parameter	**(A) MRP**	**(B) MKP**
Tamanho da população	90	150
Número de gerações*	100	100
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 (MEAMT)	90	150
Número de subdivisões (NSGA-III)	8	8

Na tabela **??**, o asterisco em "número de gerações" é para dizer 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 comparações. No problema da mochila com 100 itens, devido à complexidade do problema, dobrou-se a quantidade de gerações. Ainda na tabela **??**, "variável" quer dizer que a taxa de mutação foi 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 [12-Bracis].

Análise do MKP-30

Análise do MKP-50

Análise do MKP-100

Análise geral do MKP

Análise do PRM-R1

Análise do PRM-R2

Análise do PRM-R3

Análise geral do PRM

Análise geral dos resultados comparando os dois problemas.

Figura 9 – Etapa 1: resultados para o PMM com 30 itens

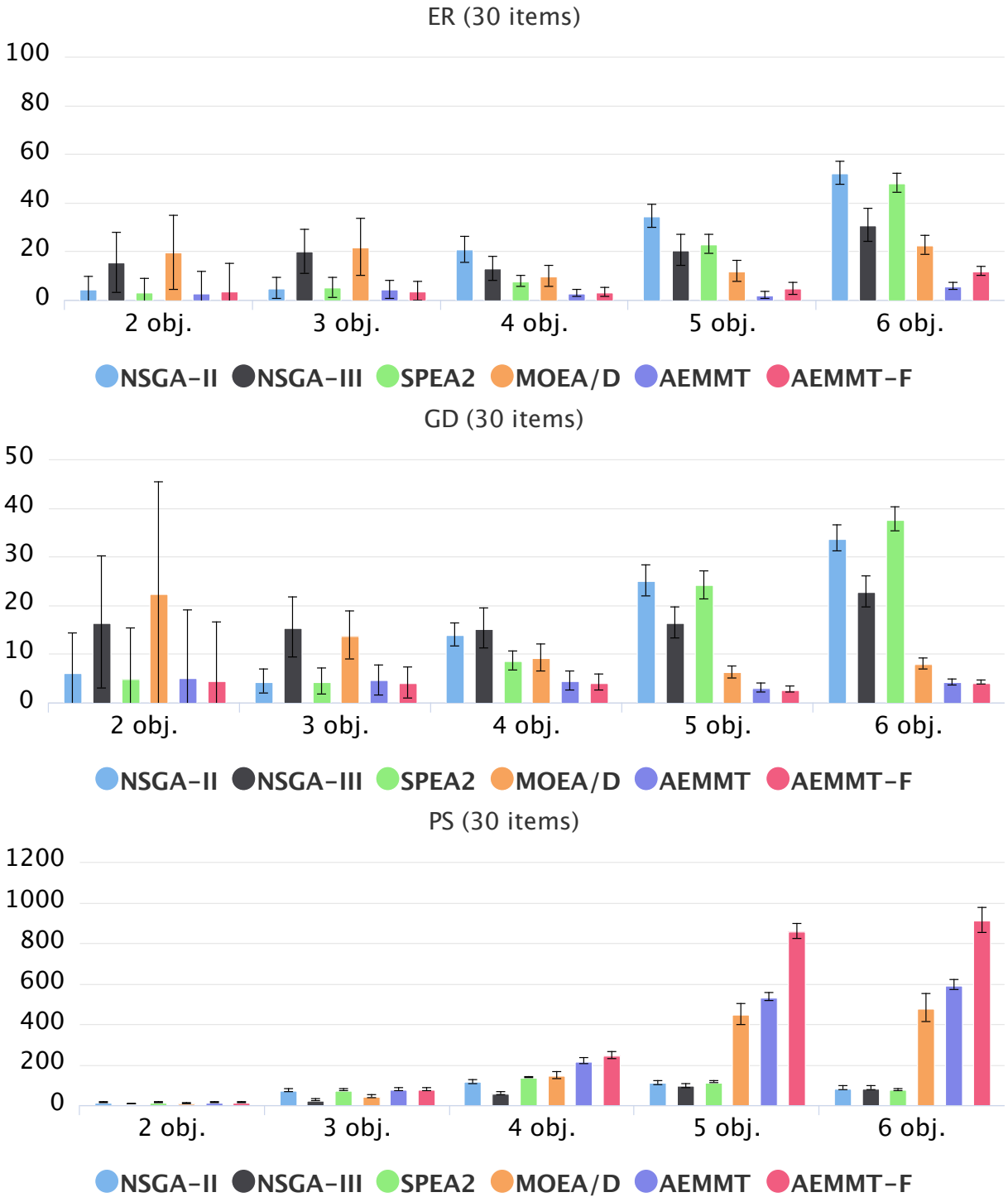


Figura 10 – Etapa 1: resultados para o PMM com 50 itens

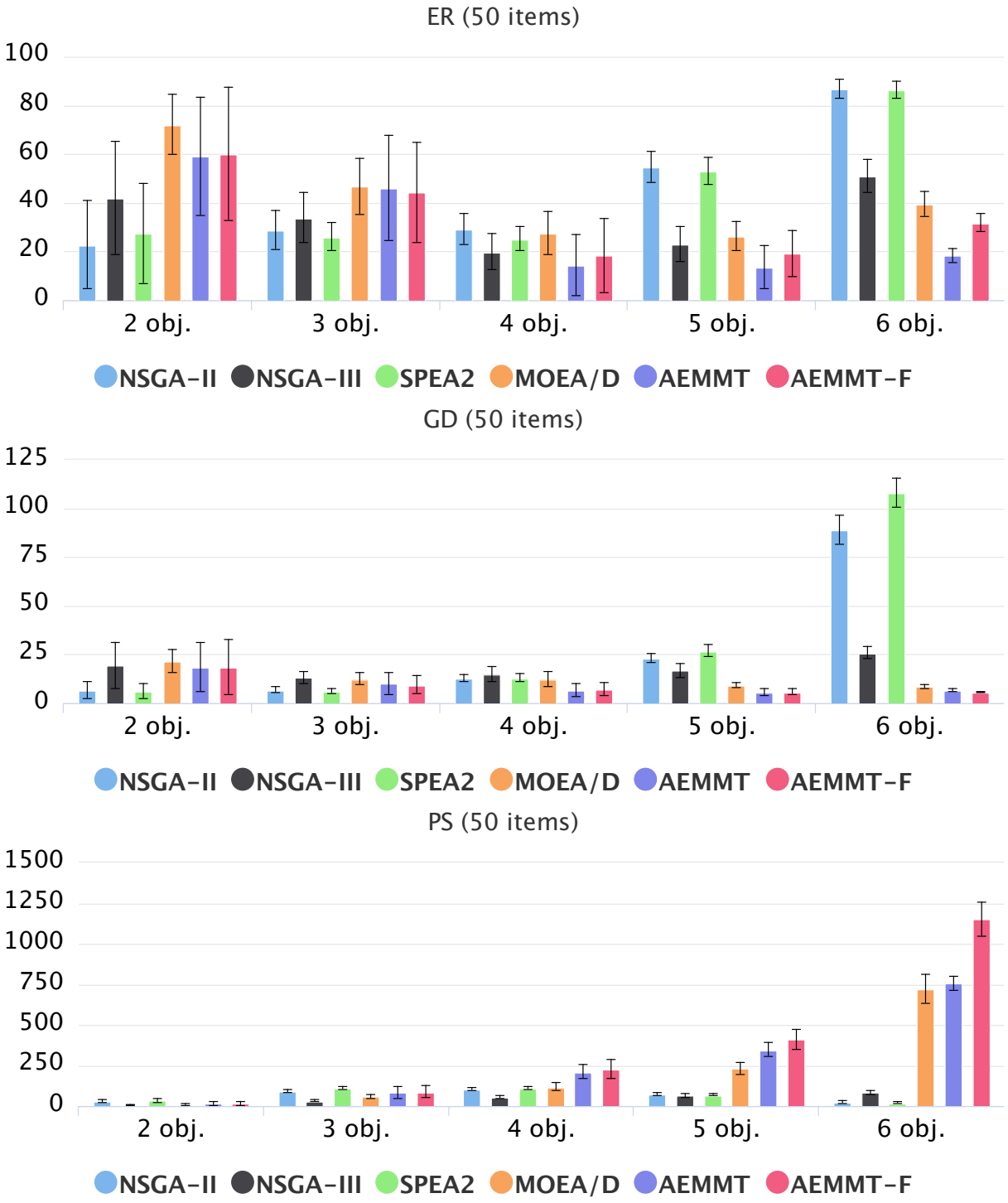


Figura 11 – Etapa 1: resultados para o PMM com 100 itens

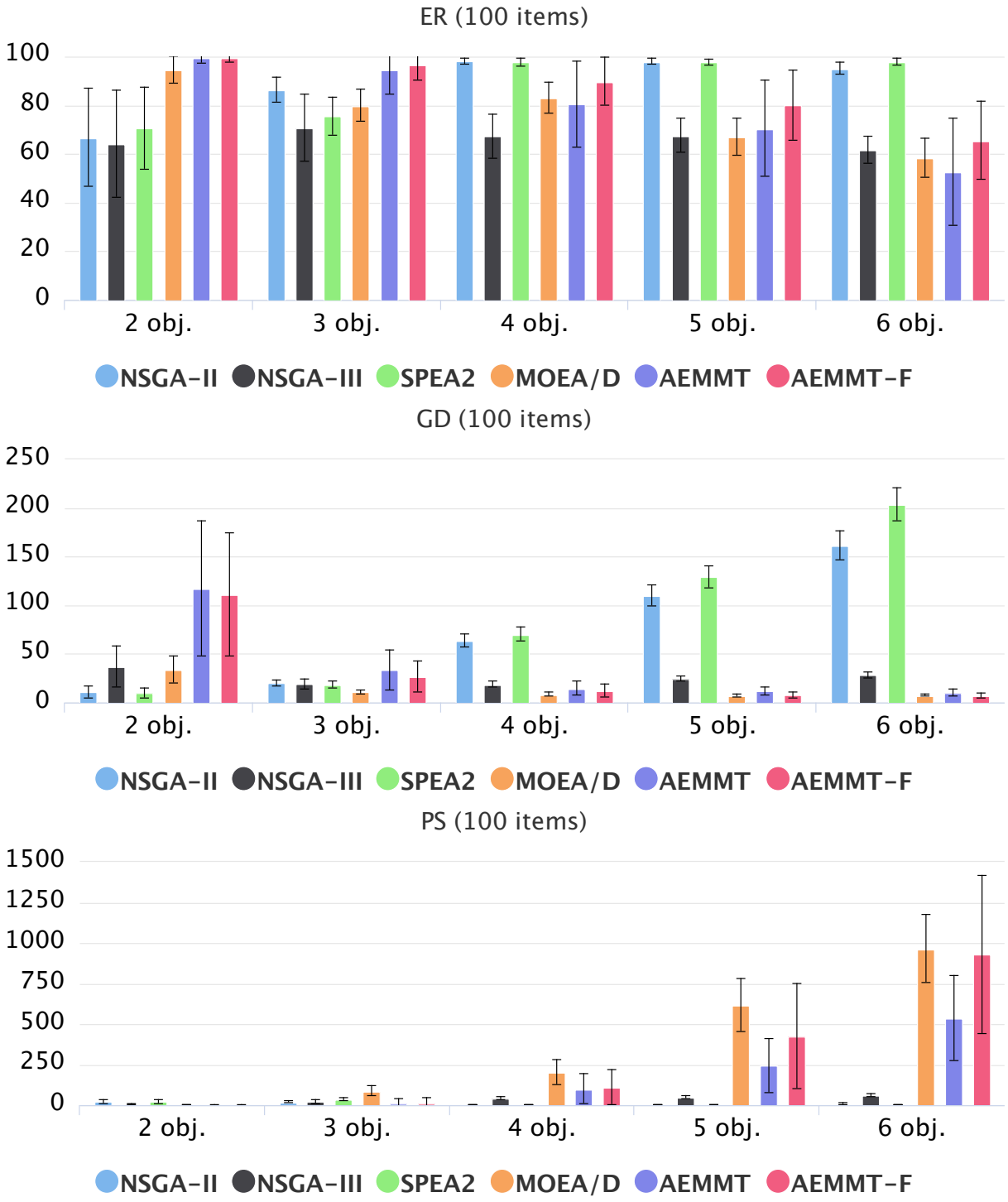


Figura 12 – Etapa 1: resultados agrupados para o PMM com 30, 50 e 100 itens

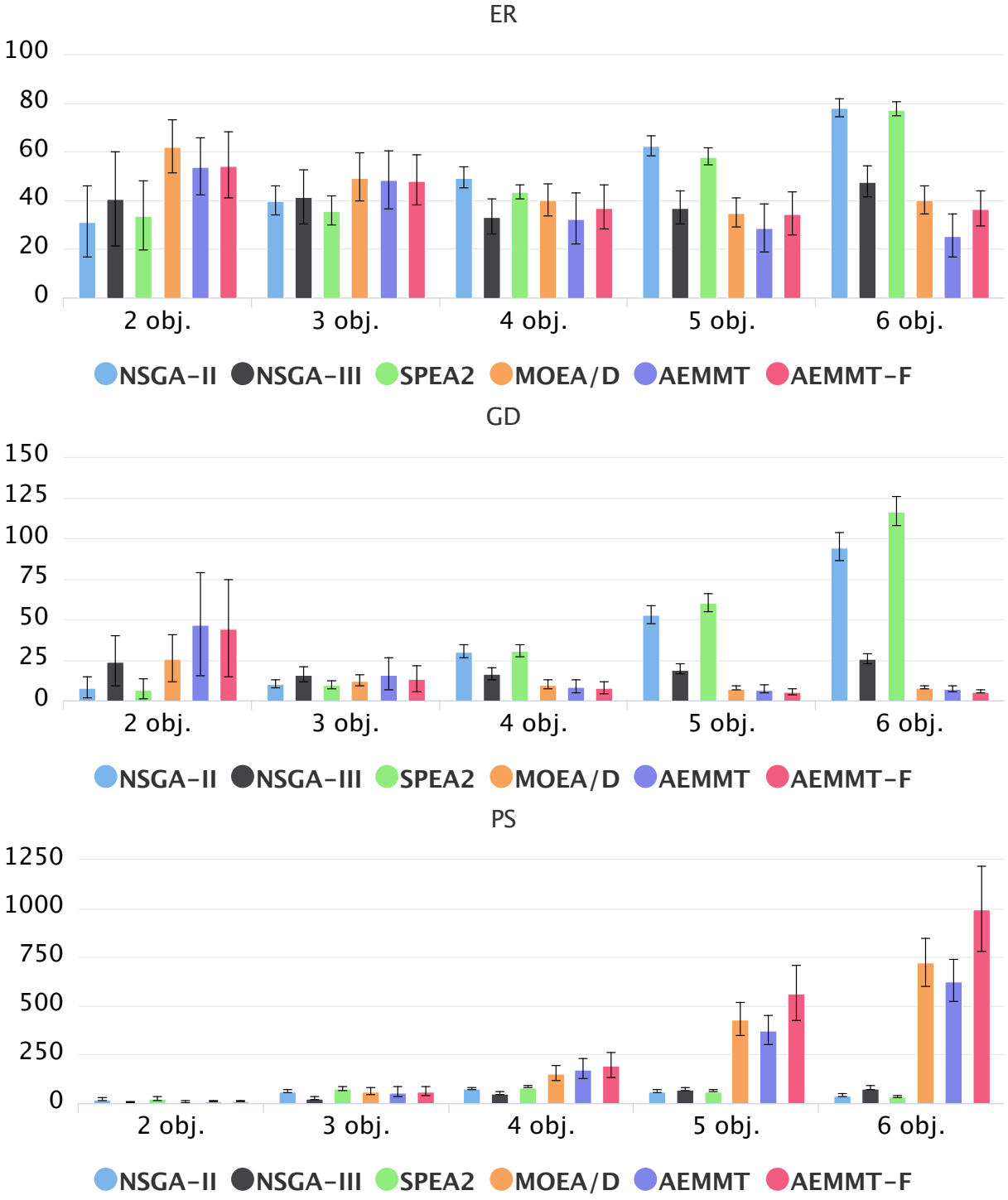


Figura 13 – Etapa 1: resultados para o PRM na rede R_1

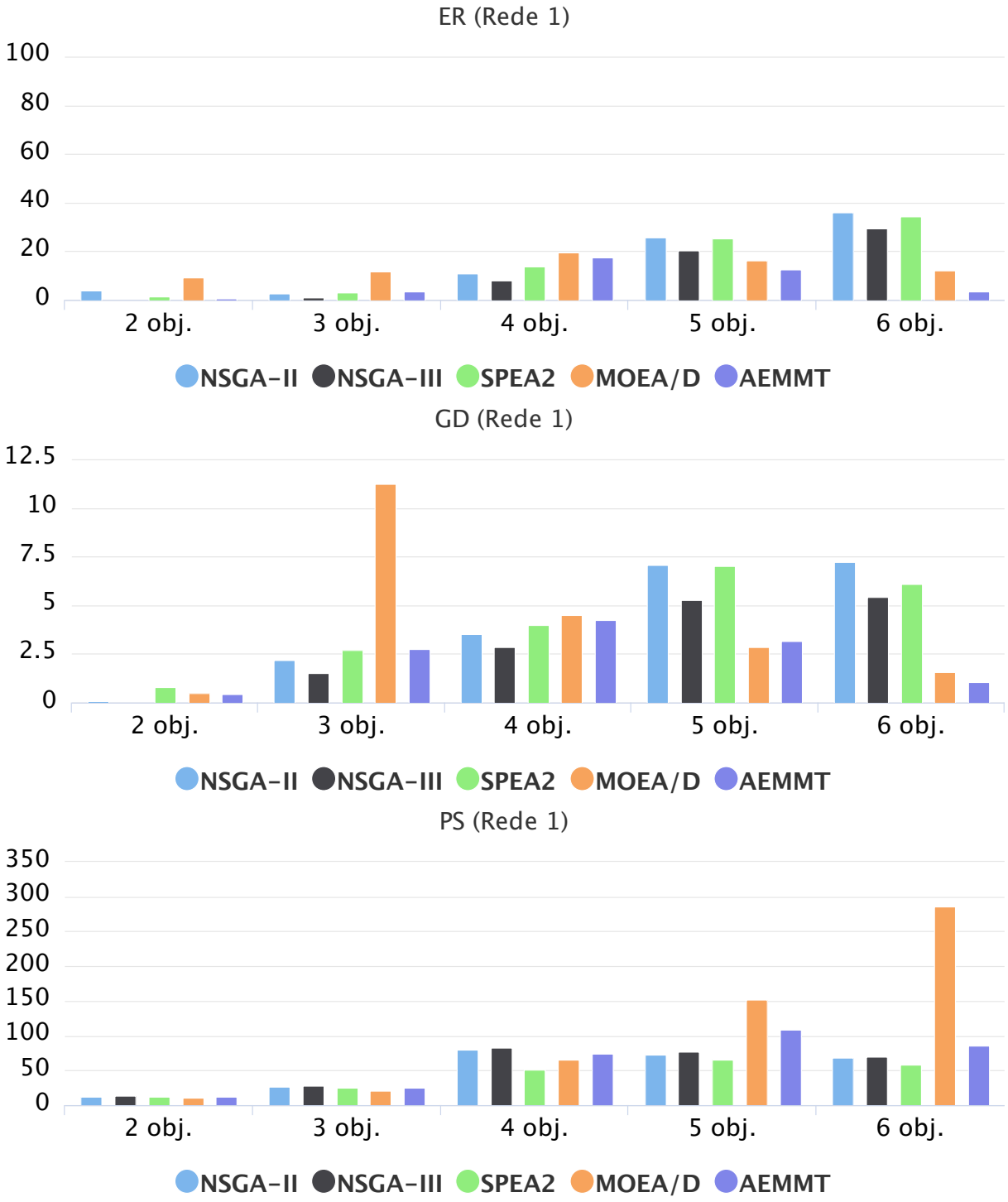


Figura 14 – Etapa 1: resultados para o PRM na rede R_2

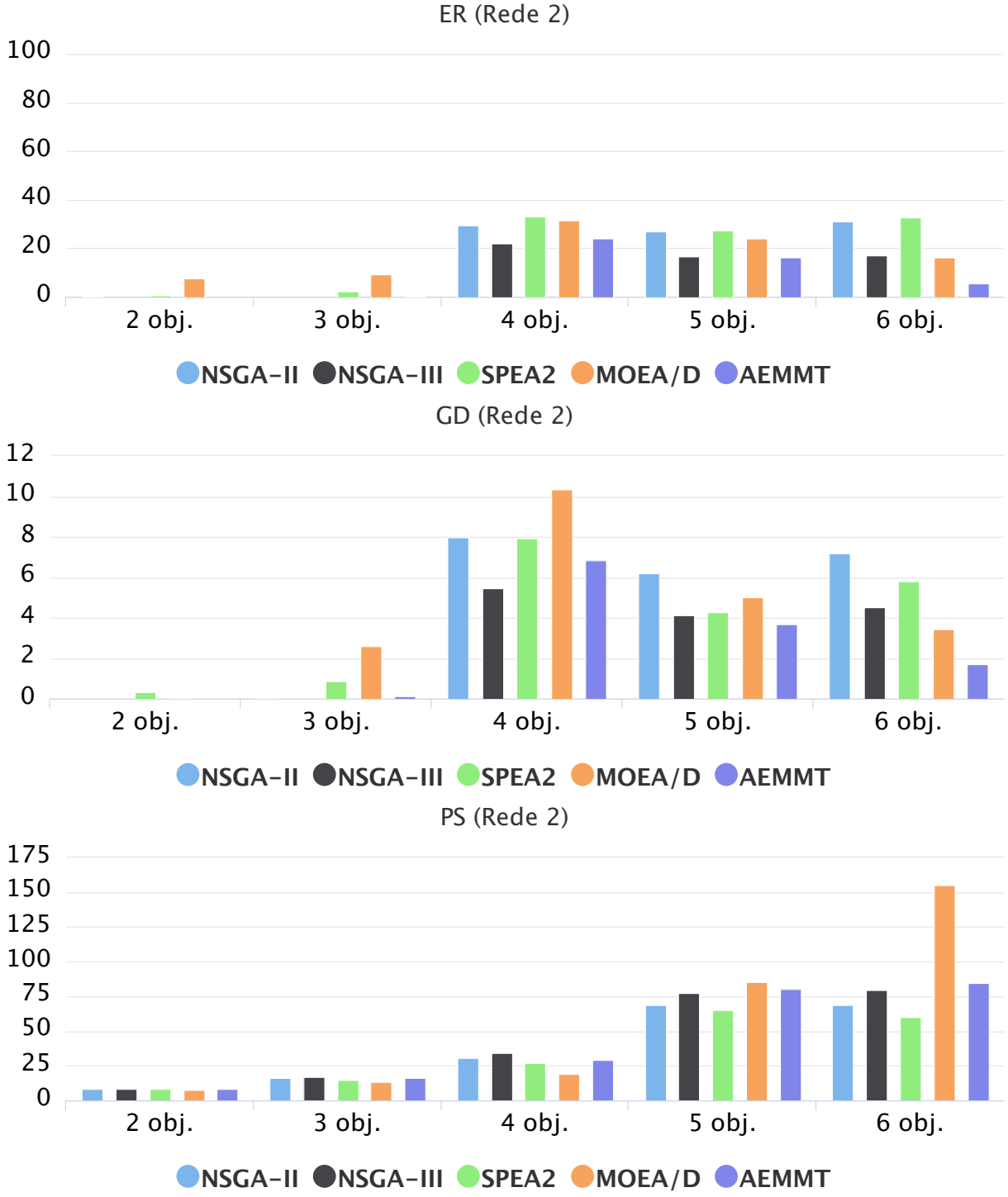


Figura 15 – Etapa 1: resultados para o PRM na rede R_3

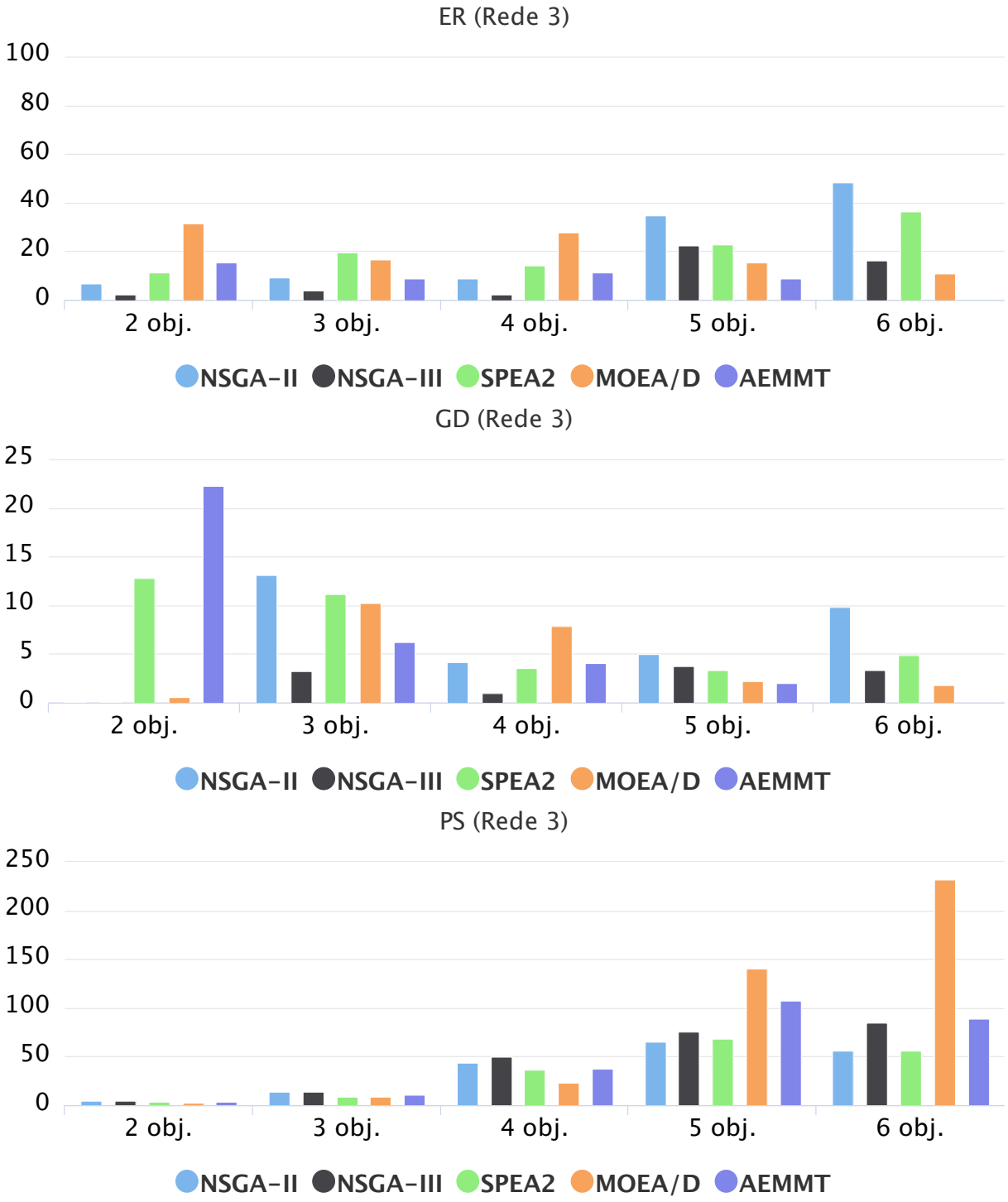
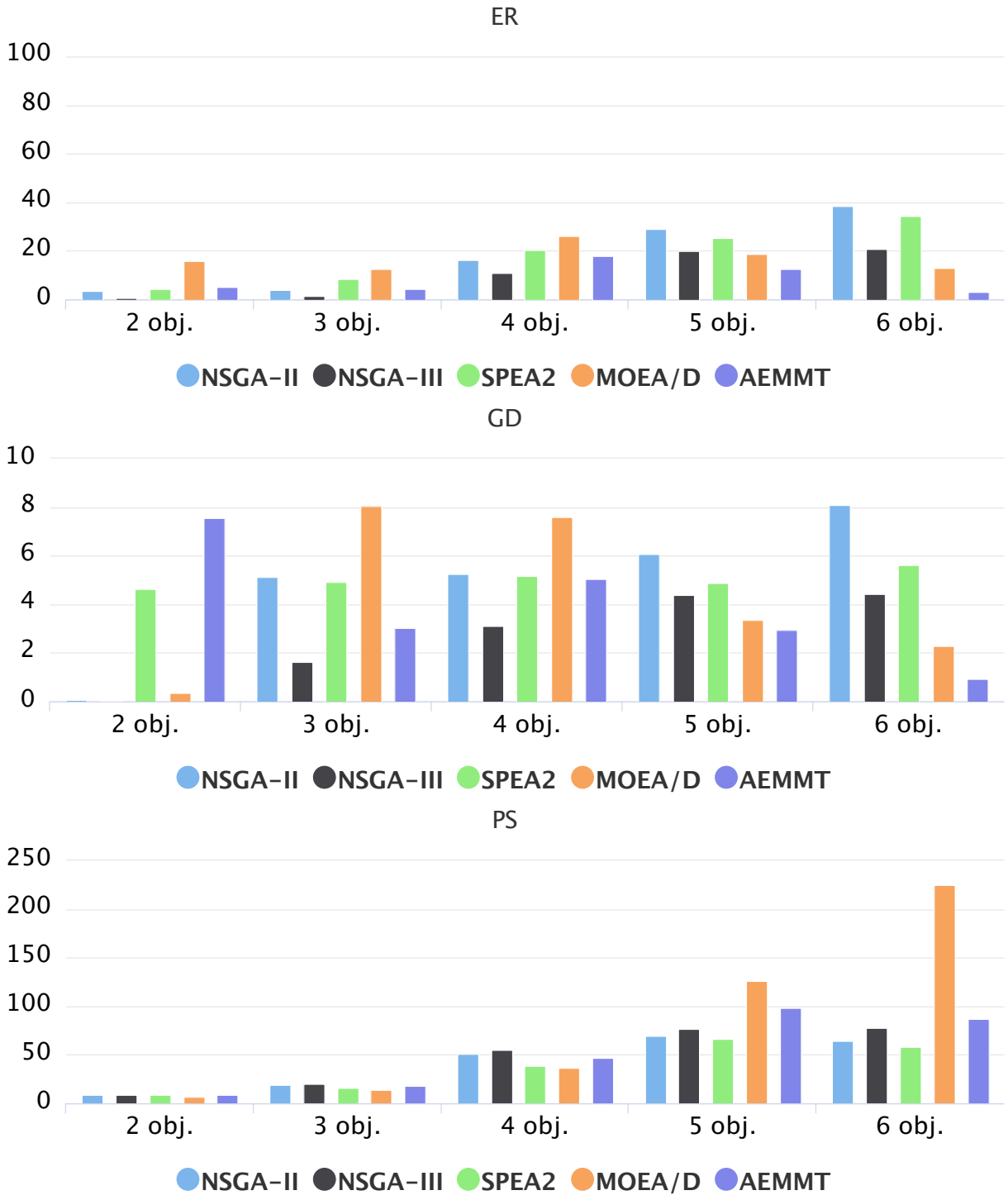


Figura 16 – Etapa 1: resultados agrupados para o PRM nas redes R_1, R_2 e R_3



8.2 Etapa 2: AG's many-objectives vs. MACO/D

Na segunda etapa de experimentos descartam-se os AEMOs clássicos NSGA-II e SPEA2 e inclui-se o AEMMD e o MACO/D, algoritmo proposto neste trabalho. Uma nova métrica de desempenho é utilizada, o tempo, e as demais continuam sendo o erro (ER), a distância (GD) e o número de soluções corretas (PS). Assim como na etapa 1, o Pareto aproximado foi pré-calculado a partir de múltiplas execuções dos algoritmos e é apresentado na tabela 3. Enfim, os resultados foram obtidos através das médias entre 100 execuções dos 30 cenários descritos na lista a seguir.

- PRM: 5 formulações de objetivos (P_4, P_5 e P_6) e 3 redes (R_1, R_2 e R_3). Tanto as formulações quanto às redes foram descritas na seção correspondente ao problema do roteamento multicast [].
- PMM: 5 formulações de objetivos (4 a 6) e 3 instâncias (30, 40 e 50 itens).

Tabela 3 – Cardinalidade dos Paretos encontrados para a 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*

A tabela 4 apresenta os parâmetros dos algoritmos utilizados neste experimento. Note que o número de gerações (marcado com asterisco) deve ser multiplicado pelo tamanho da população no AEMMT e AEMMD devido ao fato de realizarem apenas um crossover por geração.

As figuras **??**, **??** e **??** mostram respectivamente os resultados para o PMM de 30, 40 e 50 itens. As figuras **??**, **??** e **??** revelam respectivamente os resultados para o PRM aplicado às redes R_1, R_2 e R_3. Uma análise conjunta, com uma média entre as três instâncias de cada problema é apresenta nas figuras **??** (PRM) e **??** (PMM).

Análise do MKP-30

Análise do MKP-40

Análise do MKP-50

Análise geral do MKP

Análise do PRM-R1

Análise do PRM-R2

Análise do PRM-R3

Análise geral do PRM

Análise geral dos resultados comparando os dois problemas.

Tabela 4 – Parâmetros utilizados para o PRM e o PMM.

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/D)	1, 2, 0.3	1, 4.3, 0.3
Intervalo de valores para os feromônios (MACO/D)	[0.1, 0.9]	[0.1, 0.9]
Tamanho das amostras (MACO/D)	10	25% of the number of items
Tamanho do grupo de estruturas ativas (MACO/D)	5	5

Figura 17 – Etapa 2: resultados para o PMM com 30 itens

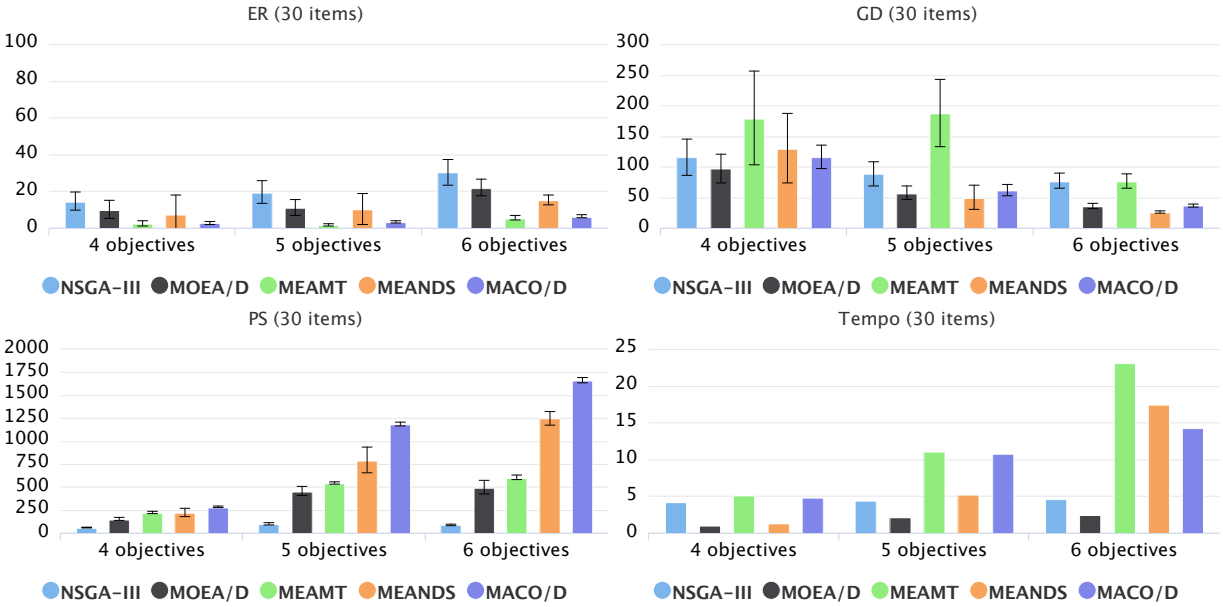


Figura 18 – Etapa 2: resultados para o PMM com 50 itens

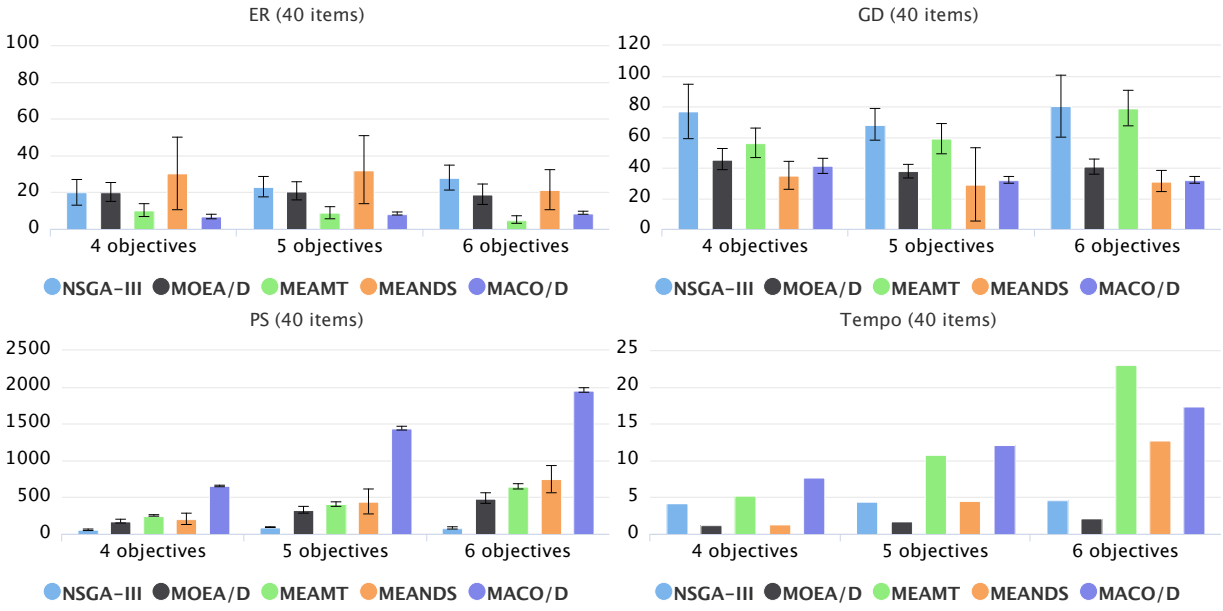


Figura 19 – Etapa 2: resultados para o PMM com 100 itens

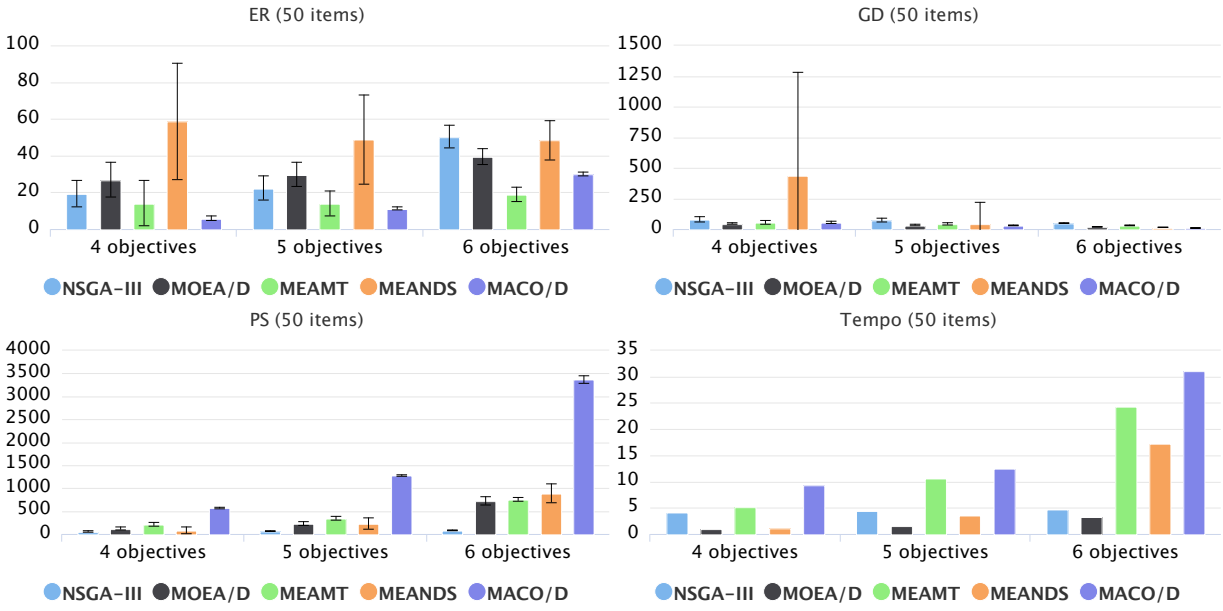


Figura 20 – Etapa 2: resultados agrupados para o PMM com 30, 40 e 50 itens

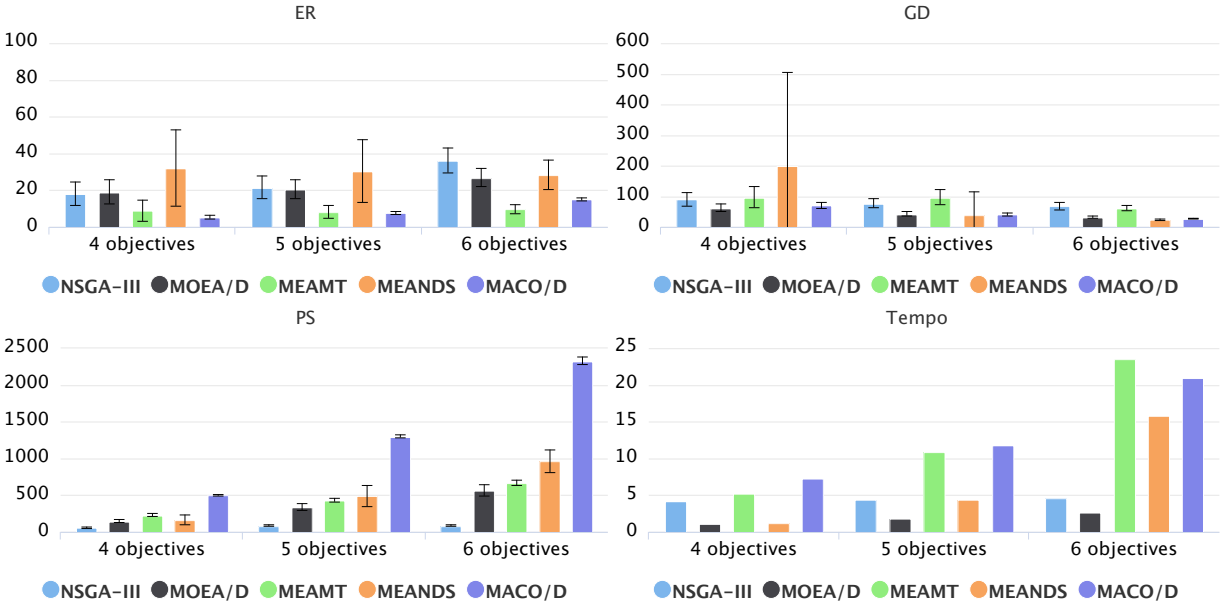


Figura 21 – Etapa 2: resultados para o PRM na rede R_1

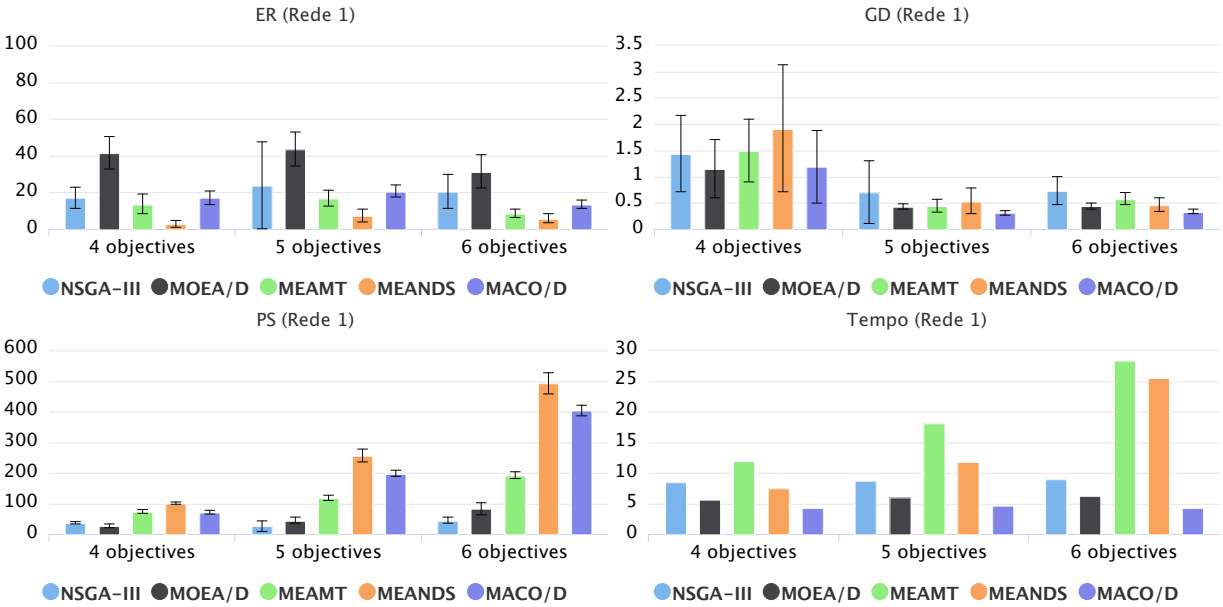


Figura 22 – Etapa 2: resultados para o PRM na rede R_2

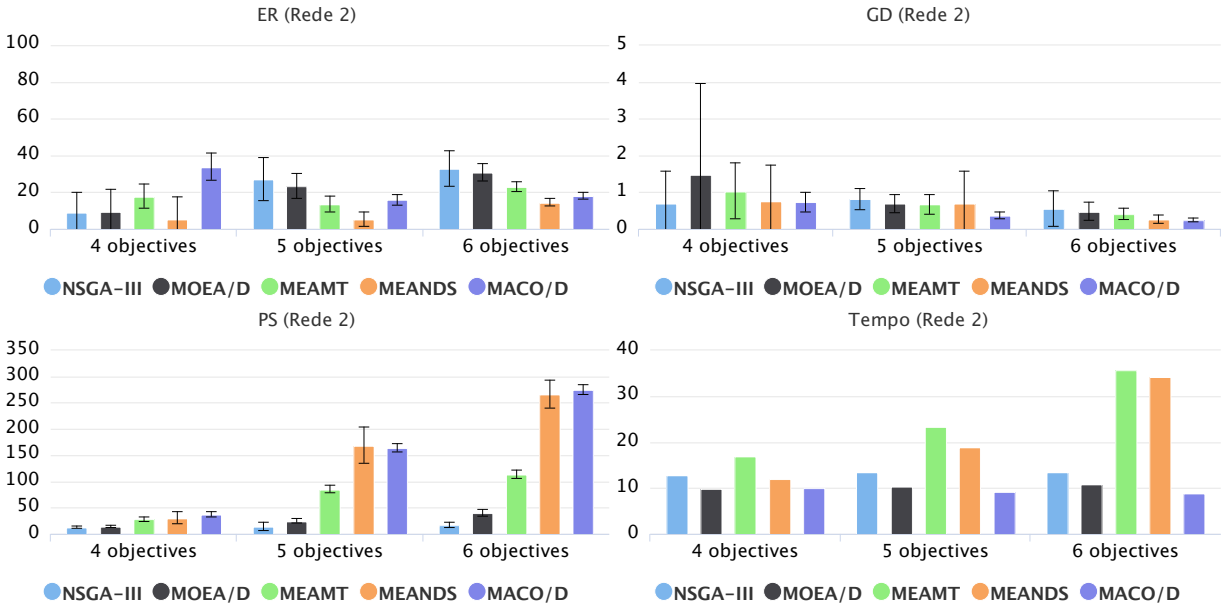


Figura 23 – Etapa 2: resultados para o PRM na rede R_3

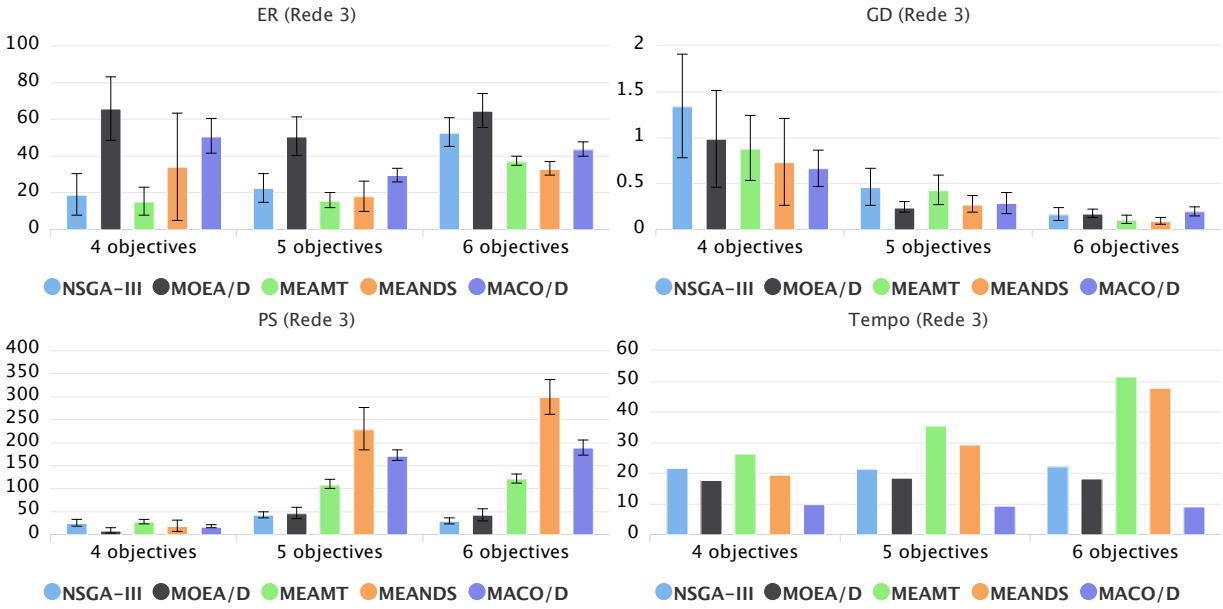
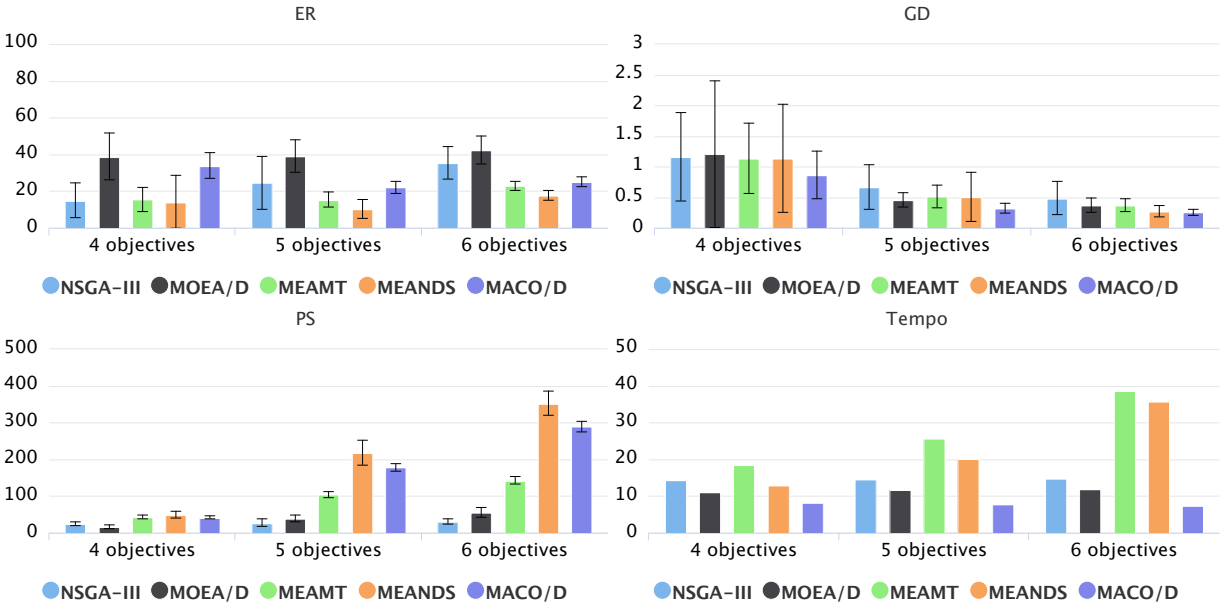


Figura 24 – Etapa 2: resultados agrupados para o PRM nas redes R_1, R_2 e R_3



8.3 Etapa 3: Análise com hipervolume

A fim de testar propriamente o comportamento dos algoritmos em espaços de busca mais complexos que os utilizados nos experimentos das etapas 1 e 2, lançou-se mão de três novas redes e instâncias do problema da mochila com 100 e 200 itens. Como não é possível extrair Paretos estáveis para as redes R_4, R_5 e R_6, nem para os problemas da mochila com 50, 100 e 200 itens, não é interessante basear-se em métricas dependentes de tais Paretos para se tirar conclusões. Por isso, nesta etapa, testa-se unicamente a métrica hipervolume, independente do Pareto.

O hipervolume foi utilizado pela primeira vez em [] e desde então, junto a *inverse generational distance* (IDG) [], é a métrica mais utilizada na literatura para se avaliar algoritmos many-objectives. O hipervolume, como explicado no início deste capítulo, calcula o volume da figura geométrica formada pelas distâncias das soluções a um ponto de referência pré-definido. Os pontos de referência utilizados em cada cenário são apresentados na tabela [].

Tabela de pontos de referência.

Nesta etapa testou-se os algoritmos NSGA-III, SPEA2-SDE, MOEA/D, AEMMT, AEMMD, MOACS e MACO/D. Foram 3 formulações de objetivo para cada problema (4, 5 e 6 objetivos) e 3 instâncias, totalizando 18 cenários de teste. Foram realizadas 10 execuções para cada algoritmo em cada cenário e os resultados foram calculados através das médias dos hivervolumes de cada execução.

Falar sobre figuras. Exibir figuras. Discutir resultados.

Capítulo 9

Conclusão

Faça uma breve introdução para o capítulo. Observe os objetivos geral e específicos do trabalho no capítulo de introdução e coloque aqui um comentário sobre como o desenvolvimento ajudou a chegar a cada um desses objetivos, ou seja, como a pesquisa permitiu concluir que cada um dos objetivos foi atingido.

9.1 Principais Contribuições

Nessa seção destaque ainda mais as suas contribuições, mostrando que sua hipótese foi validada pelos experimentos executados.

9.2 Trabalhos Futuros

Destaque nessa seção o que pode ser melhorado no método proposto para resolver as possíveis falhas que você identificou e descreveu na seção **??**. Indique quais outros projetos podem ser gerados a partir do seu trabalho.

9.3 Contribuições em Produção Bibliográfica

Liste a produção bibliográfica resultante do seu trabalho.

Referências

ABNTEX2. **A classe abntex2: Modelo canônico de trabalhos acadêmicos brasileiros compatível com as normas ABNT NBR 14724:2011, ABNT NBR 6024:2012 e outras**. [S.l.], 2013. Disponível em: <http://code.google.com/p/abntex2/>.

ARAUJO, L. C. **Configuração: uma perspectiva de Arquitetura da Informação da Escola de Brasília**. Dissertação (Mestrado) — Universidade de Brasília, Brasília, Março 2012.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10520**: Informação e documentação — apresentação de citações em documentos. Rio de Janeiro, 2002. 7 p.

_____. **NBR 6028**: Resumo - apresentação. Rio de Janeiro, 2003. 2 p.

_____. **NBR 14724**: Informação e documentação — trabalhos acadêmicos — apresentação. Rio de Janeiro, 2011. 15 p. Substitui a Ref. **??**).

_____. **NBR 6024**: Numeração progressiva das seções de um documento. Rio de Janeiro, 2012. 4 p.

van GIGCH, J. P.; PIPINO, L. L. In search for a paradigm for the discipline of information systems. **Future Computing Systems**, v. 1, n. 1, p. 71–97, 1986.