Algoritmos evolutivos multiobjetivo baseados em tabelas para escalonamento de tarefas em ambientes multiprocessados

Johnata Ferreira Santos

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

Uberlândia
2023

Johnata Ferreira Santos

Algoritmos evolutivos multiobjetivo baseados em tabelas para escalonamento de tarefas em ambientes multiprocessados

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: Prof. Dr. Paulo Henrique Ribeiro Gabriel
Coorientador: Prof. Dr. Murillo Guimarães Carneiro

Uberlândia
2023

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S237 Santos, Johnata Ferreira, 1988-
2023 Algoritmos evolutivos multiobjetivo baseados em tabelas para escalonamento de tarefas em ambientes multiprocessados [recurso eletrônico] / Johnata Ferreira Santos. - 2023.

Orientador: Paulo Henrique Ribeiro Gabriel.
Coorientador: Murillo Guimarães Carneiro.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2023.163
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Gabriel, Paulo Henrique Ribeiro, 1984-, (Orient.). II. Carneiro, Murillo Guimarães,1988-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado 3/2023, PPGCO				
Data:	28 de fevereiro de 2023	Hora de início:	14:08	Hora de encerramento:	16:58
Matrícula do Discente:	12012CCP005				
Nome do Discente:	Johnata Ferreira Santos				
Título do Trabalho:	Algoritmos evolutivos multiobjetivo baseados em tabelas para escalonamento de tarefas em ambientes multiprocessados				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Luiz Gustavo Almeida Martins - FACOM/UFU, Danilo Sipoli Sanches - UTFPR, Murillo Guimarães Carneiro - FACOM/UFU (Coorientador) e Paulo Henrique Ribeiro Gabriel - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Danilo Sipoli Sanches - Cornélio Procópio/PR; Luiz Gustavo Almeida Martins, Paulo Henrique Ribeiro Gabriel e Murillo Guimarães Carneiro - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Paulo Henrique Ribeiro Gabriel, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Luiz Gustavo Almeida Martins**, **Professor(a) do Magistério Superior**, em 10/03/2023, às 08:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Paulo Henrique Ribeiro Gabriel**, **Professor(a) do Magistério Superior**, em 10/03/2023, às 09:12, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Murillo Guimarães Carneiro**, **Professor(a) do Magistério Superior**, em 21/03/2023, às 15:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Danilo Sipoli Sanches**, **Usuário Externo**, em 21/03/2023, às 18:35, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4289153** e o código CRC **29767F9E**.

Dedico este trabalho primeiramente a Deus por ter me dado forças para terminá-lo.
Dedico a minha mãe Maria Aparecida Diniz e ao meu irmão Leandro Ferreira Santos por estarem presentes quando eu precisava.
Dedico a minha esposa Ana Paula Lopes Ferreira pelo incentivo e apoio, pela paciência, por ter cuidado de nossa filhinha Chloe Ferreira Lopes sozinha quando eu precisava de tempo para me dedicar ao trabalho. A ela, muito obrigado.
Dedico a meu pai, já falecido, mas que onde quer que esteja tenho certeza que está orgulhoso por esse caminho que trilhei.

Agradecimentos

Primeiramente agradeço a Deus por ter me dado forças para finalizar esse trabalho. Não foi fácil. Conciliar o Mestrado com um trabalho em tempo integral foi extremamente difícil, mas com certeza aproveitei toda a jornada. Agradeço a minha mãe por todos os ensinamentos que me deu, ao meu pai por me orientar sobre o valor do trabalho; antes talvez eu não compreendia tanto, mas agora, mais experiente, entendo tudo que queriam me passar. Agradeço ao meu irmão pela amizade. Agradeço a minha esposa pela dedicação que ela tem por nossa família. Agradeço minha filhinha Chloe por iluminar minha vida. Agradeço ao meu orientador, Dr. Paulo Henrique Ribeiro Gabriel, pelo apoio total no trabalho, pela paciência, pelas várias horas de conversas, pelos esboços criados e desconsiderados, mas que serviram de insumo para o desenho da obra final. Agradeço ao meu co-orientador, Dr. Murillo Guimarães Carneiro, pelo apoio e dicas certeiras para conclusão do trabalho. Por fim, para não ser ingrato com ninguém, agradeço a todas as pessoas que contribuíram direta ou indiretamente para conclusão deste trabalho.

O orientador deste trabalho agradece o apoio financeiro da Fundação de Amparo à Pesquisa do Estado de Minas Gerais – FAPEMIG (Processo APQ-03081-18).

"A nova onda de inteligência artificial não nos traz propriamente a inteligência, mas um componente crítico dela: a previsão."
(Ajay K. Agrawal)

Resumo

O escalonamento de tarefas é uma atividade crucial para assegurar a eficiência dos sistemas computacionais. Este problema é definido como uma série de tarefas que são executadas por vários processadores, sujeitos a diversos critérios de otimização. Alguns problemas envolvem múltiplos critérios, que podem ser conflitantes, tornando a resolução mais complexa. Existem várias soluções baseadas em heurísticas e meta-heurísticas, com algoritmos evolutivos sendo uma das ferramentas mais comuns. Neste trabalho, foram implementados três algoritmos evolutivos multiobjetivo: o NSGA-II (*Non-Dominated Sorting Genetic Algorithm II*), o AEMMT (Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas) e o AEMMD (Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias), sendo estes dois últimos representantes da nova classe de algoritmos proposta para resolver problemas com muitos objetivos. Todos os algoritmos foram adaptados para o problema apresentado e tiveram seus desempenhos avaliados comparativamente sobre um conjunto de problemas com diferentes números de tarefas, processadores e objetivos. Com base nos resultados experimentais, é possível constatar que o AEMMT produziu os melhores resultados, seguido pelo AEMMD e, por fim, pelo NSGA-II.

Palavras-chave: escalonamento de tarefas, sistemas multiprocessados, algoritmos evolutivos, otimização multiobjetivo, problemas com múltiplos objetivos.

Abstract

Task scheduling is an important activity to ensure the efficiency of computer systems. This problem is defined as a series of tasks executed by several processors, subject to different optimization criteria. Some problems involve multiple measures, which may conflict, making resolution more complex. There are several solutions based on heuristics and meta-heuristics, with evolutionary algorithms being one of the most common tools. In this work, we implemented three multiobjective evolutionary algorithms: NSGA-II (Non-Dominated Sorting Genetic Algorithm II), MEAMT (Multiobjective Evolutionary Algorithm with Many Tables), and MEAMD (Multiobjective Evolutionary Algorithm with Multiple Dominances), these last two being representatives of the new class of algorithms proposed to solve problems with many objectives. We adjust all these algorithms for the presented problem and had their performance evaluated comparatively on a set of problems with different numbers of tasks, processors and objectives. Based on the experimental results, it is possible to verify that MEAMT produced the best results, followed by MEAMD and, finally, by NSGA-II.

Keywords: task scheduling, multiprocessor systems, evolutionary algorithms, multiobjective optimization, many-objectives optimization.

Lista de ilustrações

Lista de tabelas

Lista de siglas

AE Algoritmo Evolutivo

AEMMD Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias

AEMMT Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas

AEMMT1 AEMMT com Média Simples

AEMMT2 AEMMT com Média Harmônica

AEMO Algoritmo Evolutivo Multiobjetivo

AG Algoritmo Genético

AGMO Algoritmo Genético Multiobjetivo

C *Communication Cost* (Custo de Comunicação)

CP *Critical Path* (Caminho Crítico)

DAG *Directed Acyclic Graph* (Grafo Acíclico Dirigido)

E Conjunto de Arestas

ER *Error Rate* (Taxa de Erro)

F *Flow Time* (Soma dos Tempos)

GD *General Distance* (Distância Geral)

L *Load Balance* (Balanceamento de Carga)

M *Makespan*

MEAMT *Multi-Objective Evolutionary Algorithm with Many Tables*

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

N Conjunto de Vértices

NSGA-II *Non-Dominated Sorting Genetic Algorithm II* (Algoritmo Genético de Ordenação Não Dominada II)

OBJ Objetivo

P Conjunto de Processadores

PA Pareto Aproximado

PS *Pareto Subset* (Subconjunto de Pareto)

SGA *Standard Genetic Algorithm* (Algoritmo Genético Padrão)

SPEA-II *Strength Pareto Evolutionary Algorithm II* (Algoritmo Evolutivo de Força de Pareto)

STG *Standard Task Graph* (Grafo de Tarefas Padrão)

T Conjunto de Tarefas

VEGA *Vector Evaluated Genetic Algorithm* (Algoritmo Genético Avaliado por Vetor)

W *Waiting Time* (Tempo de Espera)

Sumário

Capítulo 1

Introdução

A adoção de sistemas computacionais distribuídos, como grades e nuvens, e de processadores multinúcleo tem crescido ao longo dos anos (SADASHIV; KUMAR, 2011; SIRHAN; SERHAN, 2018; SECARA, 2020). Simultaneamente, diferentes áreas do conhecimento, como biologia, física e matemática aplicada, têm gerado grande demanda pela execução eficiente de aplicações de alto desempenho e altamente paralelizáveis (LUDÄSCHER; BOWERS; MCPHILLIPS, 2009; LIEW et al., 2017).

No entanto, para tornar sistemas computacionais úteis, tanto para a academia quanto para a indústria, ainda existem problemas a serem resolvidos. Um dos principais é o requisito de desempenho do sistema (XHAFA; ABRAHAM, 2008a; XHAFA; ABRAHAM, 2008b). Alcançar a computação de alto desempenho requer técnicas para atribuir *threads* e processos de forma eficiente aos recursos disponíveis em um ambiente de grande escala, altamente heterogêneo e dinâmico (XHAFA; ABRAHAM, 2008a).

A atividade de atribuir ou designar tarefas sobre recursos que as executem geralmente é aplicada em diferentes áreas de planejamento (XHAFA; ABRAHAM, 2008b). Dessa forma, existem muitas versões de problemas baseados nessa premissa. Por exemplo, em uma fábrica de softwares, a atividade de distribuir a elaboração dos módulos entre os desenvolvedores, considerando a ordem das atividades, o tempo de conclusão de cada tarefa e o custo final do projeto podem ser propriedades relevantes ao gestor. Da mesma forma, ao dividirem o uso de um processador, diversos processos em um sistema operacional devem obedecer uma ordem de execução estabelecida, de forma a proporcionar a execução de todos os processos.

Esta sequência de execução tem sua complexidade aumentada quando os processos encontram um cenário com múltiplos processadores, seja por unidades processadoras multinúcleos ou por arranjos de computadores interligados em rede. A atividade de organizar, designar e avaliar a sequência de execução das tarefas sob um conjunto de recursos é conhecida como escalonamento de tarefas (CASAVANT; KUHL, 1988; DONG; AKL, 2006).

O escalonamento de tarefas em sistemas multiprocessados é um clássico problema $\mathcal{NP}$-difícil (AFRATI; PAPADIMITRIOU; PAPAGEORGIOU, 1988; GAREY; JOHN-

SON, 1979) que consiste em atribuir tarefas sobre recursos computacionais, de maneira a otimizar uma métrica de desempenho. Além disso, esse tipo de problema pode apresentar uma extensa variação de cenários baseadas em restrições atribuídas às tarefas ou aos recursos. Arranjos de computadores homogêneos (isto é, sem variações de hardware) ou heterogêneos são exemplos de diferentes cenários. Outro exemplo é o custo de comunicação, que contabiliza o esforço necessário de comunicação das tarefas via rede ao longo de sua execução (DONG; AKL, 2006).

Além dessas restrições, o problema também apresenta diferentes métricas que medem o desempenho do escalonador em satisfazer os objetivos (SILVA, 2020). Eventualmente, as métricas empregadas possuem objetivos conflitantes. Por exemplo, reduzir o custo de comunicação entre duas tarefas implica reduzir o número de processadores utilizados, ou seja, pode prejudicar o balanceamento de carga entre os diferentes recursos computacionais. Uma consequência desse conflito é o aumento da complexidade do problema, gerando assim novas classes de problemas baseadas na premissa do escalonamento.

Por ser um problema constantemente abordado na literatura, existem diversas soluções inteligentes implementadas sobre heurísticas e meta-heurísticas (TOPCUOGLU; HARIRI; WU, 2002; KUMAR et al., 2019; SILVA; GABRIEL, 2020). Sobre essas últimas, os algoritmos evolutivos (AEs) são ferramentas comumente utilizadas (SILVA, 2020; SILVA; GABRIEL, 2020). Os AEs representam uma classe de técnicas bio-inspiradas que podem ser adaptadas na resolução de vários tipos de problemas (EIBEN; SMITH, 2015). Ao se inspirar no paradigma evolucionista, esses algoritmos trabalham com a evolução de várias soluções simultaneamente, garantindo uma vantagem significativa se comparadas as técnicas sequenciais. Dessa forma, podem explorar a computação paralela e portanto, apresentarem melhor desempenho. Além disso, os operadores genéticos são importantes características de AEs que combinam ou modificam soluções, expandindo a capacidade de exploração do espaço de soluções (EIBEN; SMITH, 2015; GABRIEL; DELBEM, 2008).

Algoritmos evolutivos também têm sido utilizados, com sucesso, para lidar com problemas que apresentam múltiplos objetivos conflitantes (DEB, 2001). Nesse sentido, AEs multiobjetivo (AEMO) levam explicitamente em conta o conceito de Fronteira de Pareto, buscando um equilíbrio entre os objetivos. Na fronteira de Pareto não existe somente uma solução para o problema, mas sim um conjunto de soluções. AEMOs para problemas de escalonamento, geralmente, lidam com dois ou três objetivos (SILVA, 2020; SILVA; GABRIEL, 2020). No entanto, a convergência de AEMOs precisa ser melhorada quando envolve mais de três critérios de otimização (ISHIBUCHI et al., 2011; LAFETÁ et al., 2018). Devido a essa limitação, algumas pesquisas propuseram métodos para amostrar o espaço objetivo e mapear as regiões promissoras adequadamente (LI; PAN, 2015). Um dos métodos que adota essa abordagem é o Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas (AEMMT), que decompõe o problema original em vários problemas multiobjetivo mais direto usando subpopulações (ou tabelas). Esse AE emprega um método para

combinar objetivos, permitindo uma melhor exploração do espaço de busca (BRASIL; DELBEM; SILVA, 2013; LAFETÁ et al., 2018). Mais recentemente, uma extensão desse algoritmo, denominada Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias (AEMMD), alcançou resultados significativos em problemas de otimização combinatória discretos. Esse algoritmo que mantém uma subpopulação de soluções não-dominadas, buscando uma melhor exploração da busca espaço (LAFETÁ et al., 2018).

Diante do exposto, este trabalho tem como objetivo explorar ambos algoritmos no contexto de escalonamento de tarefas multiprocessado. São consideradas cinco funções objetivo bastante conhecidas da literatura, mas pouco exploradas em conjunto: *makespan*, balanceamento de carga, *flowtime*, custo de comunicação e tempo de espera.

A fim de atingir esse objetivo, inicialmente, implementou-se o algoritmo genético mono-objetivo proposto por Omara e Arafa (2010). Nesse algoritmo, foram também aplicadas as métricas não utilizadas no trabalho original. O AG de Omara e Arafa (2010) traz um modelo de cromossomo e operadores de reprodução que foram adaptados aos algoritmos de múltiplos objetivos estudados neste trabalho, ou seja, AEMMT e AEMMD. Como base de comparação, utilizou-se o NSGA-II (DEB et al., 2002), algoritmo evolutivo multiobjetivo bastante usado na literatura. Como resultado, observou-se o desempenho superior dos algoritmo AEMMT e AEMMD mesmo em cenários com apenas dois objetivos.

É importante ressaltar que a possibilidade de trabalhar com diferentes objetivos é uma contribuição relevante para a área de pesquisa, visto que poucos trabalhos focam em um estudo sistemático das diferentes métricas de desempenho (KUMAR et al., 2019). Em suma, seguem as principais contribuições desta pesquisa:

- ❏ Investigação de um problema de escalonamento multiobjetivo com cinco objetivos, número pouco explorado na literatura que se concentra na otimização de até três objetivos;
- ❏ Desenvolvimento de AEs multiobjetivo para o problema do escalonamento de tarefas em que a representação, a codificação e os operadores genéticos foram criados ou adaptados de um AE mono-objetivo;
- ❏ Projeto de duas função de cálculo de aptidão robusta com base na média simples e na média harmônica dos vários objetivos;
- ❏ Investigação de problemas de otimização em larga escala para escalonamento de tarefas com até 300 tarefas ao invés das 50 ou menos, valores geralmente considerados por trabalhos relacionados.

Os demais capítulos desta dissertação estão organizados como descrito a seguir. O Capítulo 2 apresenta os principais conceitos de algoritmos evolutivos, suas características

e formas de execução. Também são introduzidas definições de otimização multiobjetivo, como a fronteira de Pareto, e são descritos os AEs implementados neste tabalho (NSGA-II, AEMMT e AEMMD). O Capítulo 3 é responsável por descrever o problema estudado neste trabalho, ou seja, o escalonamento de tarefas em sistemas multiprocessados. São apresentadas as principais definições e as funções objetivo consideradas.

O Capítulo 4 é dedicado ao desenvolvimento deste trabalho. São detalhadas todas as particularidades do AG mono-objetivo utilizado como base para implementação dos AEs multiobjetivos: representação, função de aptidão, método de seleção, recombinação, mutação e reinserção. Também são apresentadas as médias usadas na avaliação e a forma adotada para normalizar os resultados. No Capítulo 5, são apresentados os experimentos realizados e os seus resultados. Nesse capítulo, são descritos os ambientes, métodos para avaliação e parâmetros de configuração das execuções dos AEs e a análise dos resultados.

Finalmente, o Capítulo 6 apresenta as conclusões desta monografia e indica possíveis direções de pesquisa. Em seguida, são listadas as referências bibliográficas e apresentado o Apêndice A, onde são mostrados resultados adicionais que não foram incluídos no Capítulo 5.

Capítulo 2

Algoritmos Evolutivos para Otimização Multiobjetivo

Problemas de otimização com diversos objetivos têm despertado grande interesse na área de Pesquisa Operacional. Nesses problemas, a qualidade da solução é definida com base na sua adequação em relação a diversos objetivos (ou critérios), muitos dos quais conflitantes (DEB et al., 2002; EIBEN; SMITH, 2015). Desse modo, busca-se analisar um "compromisso" (em inglês, *trade-off*) entre os diferentes objetivos.

Historicamente, métodos de otimização multiobjetivo buscam reduzir esses problemas a outros com apenas um objetivo, simplificando, assim, o problema. Uma classe de algoritmos bastante utilizada nesse contexto é o método de pesos, que multiplica cada função objetivo por um valor escalar e realiza um processo de otimização da função de peso, ou seja, da soma ponderada das funções objetivo. Esses métodos, no entanto, são dependentes da escolha adequada dos pesos, implicando um conhecimento prévio dos intervalos correspondentes aos pesos mais adequados. Por essa razão, métodos que tentam encontrar soluções que apresentam um compromisso com os vários objetivos sem a utilização de pesos passaram a ser explorados (DEB et al., 2002; EIBEN; SMITH, 2015). Entre esses métodos, destacam-se os algoritmos evolutivos (AEs).

Neste capítulo, inicialmente, são apresentados os AEs (seção 2.1), seu funcionamento e seus principais operadores. Em seguida (seção 2.2), discute-se conceitos de otimização multiobjetivo. Finalmente, (seção 2.3), são abordados os algoritmos evolutivos multiobjetivos bem como o detalhamento dos dois algoritmos evolutivos baseados em tabelas e que foram estudados mais a fundo neste trabalho.

2.1 Algoritmos Evolutivos

Algoritmos evolutivos (AEs) são métodos de busca e otimização genéricos, baseados em população, que utilizam mecanismos inspirados em Biologia, como seleção natural e sobrevivência do indivíduo mais adaptado (EIBEN; SMITH, 2015). Simulam em compu-

tador o processo de evolução natural, executando uma eficiente e sistemática busca por valores ótimos no espaço de possíveis soluções de um problema de otimização (GABRIEL; DELBEM, 2008). Nesse contexto, segundo De Jong (2006), três abordagens de AEs foram desenvolvidas de forma independente: a programação evolutiva (PE), as estratégias evolutivas (EEs) e os algoritmos genéticos (AGs). Dentre esses, sem dúvidas os algoritmos genéticos (AGs) são os mais conhecidos, principalmente devido à sua utilização em diferentes áreas do conhecimento (GABRIEL; DELBEM, 2008; EIBEN; SMITH, 2015).

Algoritmos genéticos são técnicas de busca baseadas na simplificação de alguns processos biológicos evolutivos. A analogia se deve ao mecanismo para geração de novos estados, onde a seleção natural é representada através da reprodução sexuada entre duas ou mais soluções. O surgimento da técnica está associada aos trabalhos de Holland (1975) e Goldberg (1989), responsáveis por popularizar os AGs nas décadas de 1970 e 1980.

AGs operam em uma população de soluções e não em uma única solução. Atuam implementando metaheurísticas tais como seleção (do inglês, *selection*), recombinação (do inglês, *crossover*) e mutação (do inglês, *mutation*), para evoluir em melhores soluções (GOLDBERG, 1989). Em resumo, os AGs pertencem a uma classe de algoritmos probabilísticos que são implementados através de buscas paralelas e adaptativas baseadas na teoria evolutiva de sobrevivência dos mais aptos (PACHECO, 1999). Esse tipo de algoritmo se difere das outras técnicas de busca e otimização em alguns princípios básicos (VON ZUBEN, 2000):

1. Operam em um espaço de soluções codificadas e não diretamente no espaço de busca;

2. Operam em um conjunto de pontos (soluções) e não a partir de um ponto isolado;

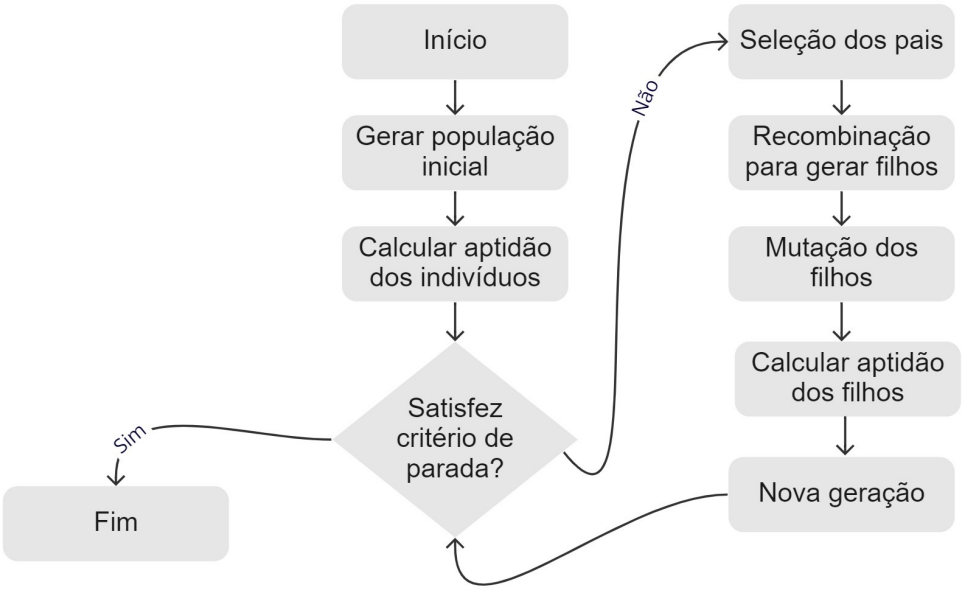
3. Necessitam de informação somente sobre o valor de uma função objetivo para cada membro da população e nenhuma outra informação extra;

4. Usam regras de transição probabilísticas e não determinísticas.

A Figura 1 apresenta as principais fases em um AG típico. No início, o algoritmo cria um conjunto de indivíduos (população) que representam possíveis soluções do problema. Para diferenciar e classificar os indivíduos, o algoritmo utiliza uma função de aptidão que atribui a cada indivíduo uma nota. Este valor será utilizado como parâmetro na seleção de indivíduos aptos para a geração de novas soluções.

Posteriormente, o algoritmo entra em um laço de n iterações (gerações) voltadas à construção de novas populações. Em cada iteração, o algoritmo gera um novo conjunto de possíveis soluções, classificando-as com a função de aptidão e escolhendo os indivíduos que migrarão para a próxima geração de forma a otimizar os resultados. Por exemplo, é possível substituir toda população antiga pela nova população ou selecionar os melhores indivíduos que permanecerão na nova geração, definindo uma estratégia elitista.

Figura 1 – Fluxograma de um algoritmo genético típico.



Fonte: Adaptado de Kang et al. (2011).

A etapa de recombinação, utiliza duas ou mais soluções, com base em suas aptidões, que terão o material genético intercalado gerando um ou mais indivíduos (descendentes). Após a geração do descendente e baseando-se em uma pequena probabilidade aleatória de ocorrência, é possível aplicar uma mudança aleatória nos dados do indivíduo, tarefa essa conhecida como mutação. Por fim, após n gerações ou até alcançar uma condição de parada, o algoritmo retornará o indivíduo que apresenta a melhor nota de aptidão (não necessariamente a solução ótima do problema).

2.1.1 Representação e Avaliação do Indivíduo

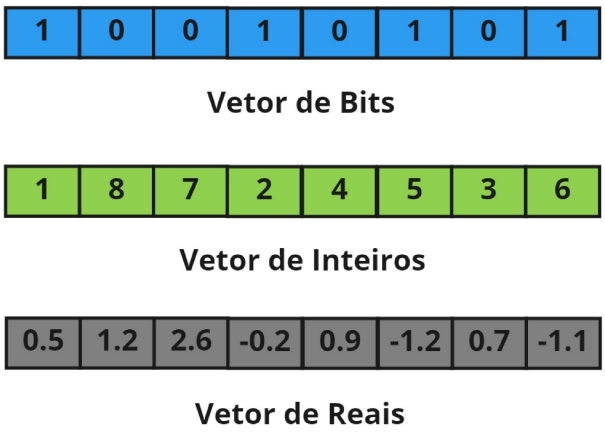
Uma das etapas mais importantes no projeto de um algoritmo genético é a representação do indivíduo (cromossomo). Esse cromossomo possui um código genético, que nada mais é que a representação de uma possível solução para o problema, ou seja, um ponto do espaço de busca do problema a ser resolvido. Essa modelagem pode ser realizada de várias formas, o que pode facilitar ou não a implementação do algoritmo (GABRIEL; DELBEM, 2008). O cromossomo é, geralmente, representado por um vetor ou um conjunto de vetores[1], conforme ilustrado na Figura 2. Seus elementos podem ser:

- Binários: representação mais utilizada, sendo a primeira desenvolvida por Holland (1962);

[1] Existem outras representações, como árvores e listas encadeadas dinâmicas, mas essas não serão discutidas nesta dissertação.

- Inteiros: muito utilizado em problemas de permutação;
- Reais: bastante utilizado em aplicações de engenharia.

Figura 2 – Exemplos de representação de cromossomos.



Fonte: autoria própria.

A qualidade de um indivíduo é a medida de desempenho utilizada para indicar a proximidade que o indivíduo tem de ser a solução para o problema (PACHECO, 1999; VON ZUBEN, 2000; GABRIEL; DELBEM, 2008). Para isso, emprega-se uma função de aptidão (do inglês, *fitness*). Todos os indivíduos da população são avaliados conforme essa função de aptidão, a qual por sua vez, é projetada a partir da função objetivo do problema de otimização. Por exemplo, em um problema de minimização, quanto menor for o valor da função objetivo, maior será a aptidão do indivíduo que gerou essa solução.

2.1.2 Métodos de Seleção

Os métodos de seleção são utilizados para escolher quais indivíduos irão se reproduzir. Os indivíduos selecionados são chamados de pais e os novos indivíduos gerados após reprodução são chamados de filhos ou descendentes. Existem diversos métodos de seleção empregados na literatura, cada um com as suas particularidades. É importante escolher um método de seleção adequado para o problema por conta da pressão seletiva. A pressão seletiva é a probabilidade do melhor indivíduo ser selecionado em comparação com a média.

É importante analisar a pressão seletiva obtida pelo método de seleção escolhido, pois ela poderá influenciar diretamente no resultado do algoritmo. Se for aplicado muita pressão, o AG corre o risco de ter uma convergência prematura; caso contrário, o AG pode se tornar fortemente aleatório e com pouca direção de busca. Resumindo, quanto maior a pressão seletiva, menor será o número de gerações necessárias para encontrar a solução para o problema. Em contrapartida, também será maior o risco de convergência prematura.

Diversos algoritmos de seleção foram propostos ao longo dos anos (EIBEN; SMITH, 2015): roleta, torneio, *ranking*, truncamento, amostragem estocástica, entre outros.

2.1.2.1 Método da Roleta

O método da roleta (do inglês, *roulette wheel selection*) é um dos métodos de seleção mais utilizados (GOLDBERG, 1989). A premissa desse método é selecionar um conjunto de reprodutores de modo que os indivíduos com maior aptidão tenham uma maior probabilidade de serem selecionados (LACERDA; CARVALHO; LUDERMIR, 2002). Seja f_i a aptidão do i-ésimo indivíduo de uma população de N indivíduos; a probabilidade $\mathcal{P}_i$ de se selecionar esse indivíduo é dada por:

$$\mathcal{P}_i = \frac{f_i}{\sum_i^N f_i} \tag{1}$$

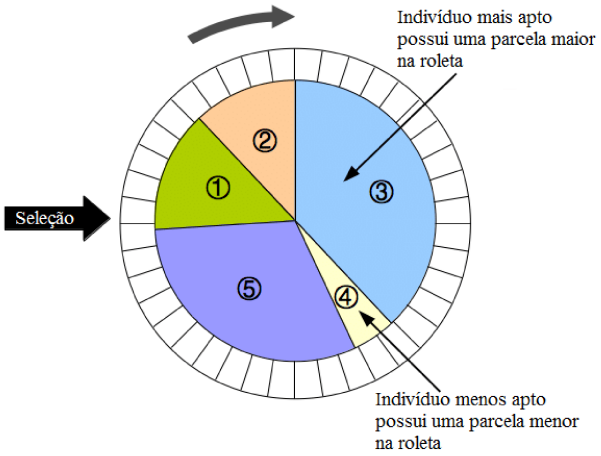
A Figura 3 apresenta a abstração do método da roleta. É importante observar que a função de aptidão pode deixar o método da roleta ser executado de uma forma não desejada. Quando os piores indivíduos tem probabilidades muitas inferiores aos melhores indivíduos, o método estará renunciando bastante os piores indivíduos o que não é algo ideal para um AG. Assim, deve-se avaliar se essa situação ocorrerá para a função de aptidão escolhida.

2.1.2.2 Método do Torneio

Outro método bastante conhecido da literatura é o método de seleção do torneio (do inglês, *tournament selection*). Nele, k indivíduos são selecionados da população de forma aleatória. A variável k, que indica o número de indivíduos selecionados, é comumente chamada de *tour*. No segundo passo, os indivíduos do grupo pré-selecionado competem entre si para determinar quem será escolhido para reprodução. Essa competição analisará a aptidão de cada integrante do grupo e aquele com o valor mais alto vencerá o torneio e será eleito.

- **Torneio simples**: a seleção dos indivíduos é realizada sem considerar a aptidão relativa;
- **Torneio estocástico**: a seleção dos indivíduos é realizada utilizando o método da roleta.

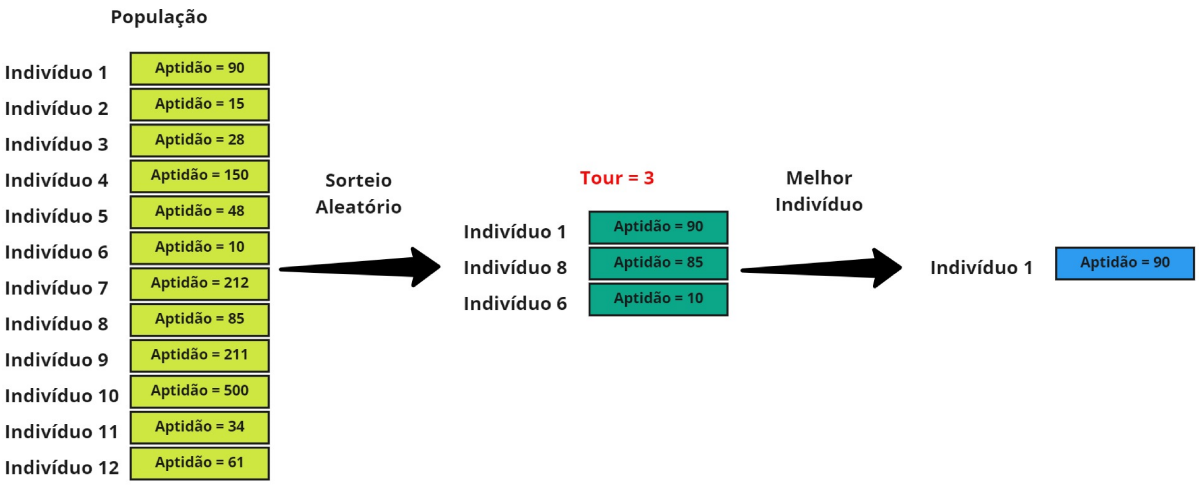
Figura 3 – Método de seleção da roleta.



Fonte: Adaptado de Wall (1996).

A Figura 4 apresenta um exemplo da utilização do método do torneio com $tour = 3$. Em resumo, três indivíduos são selecionados da população inicial, competem entre si e aquele com a maior aptidão é eleito. No exemplo da figura, o indivíduo com aptidão igual a 90 foi o vencedor. No torneio simples, todos os indivíduos possuem a mesma

Figura 4 – Ilustração do método de seleção por torneio com $tour = 3$.

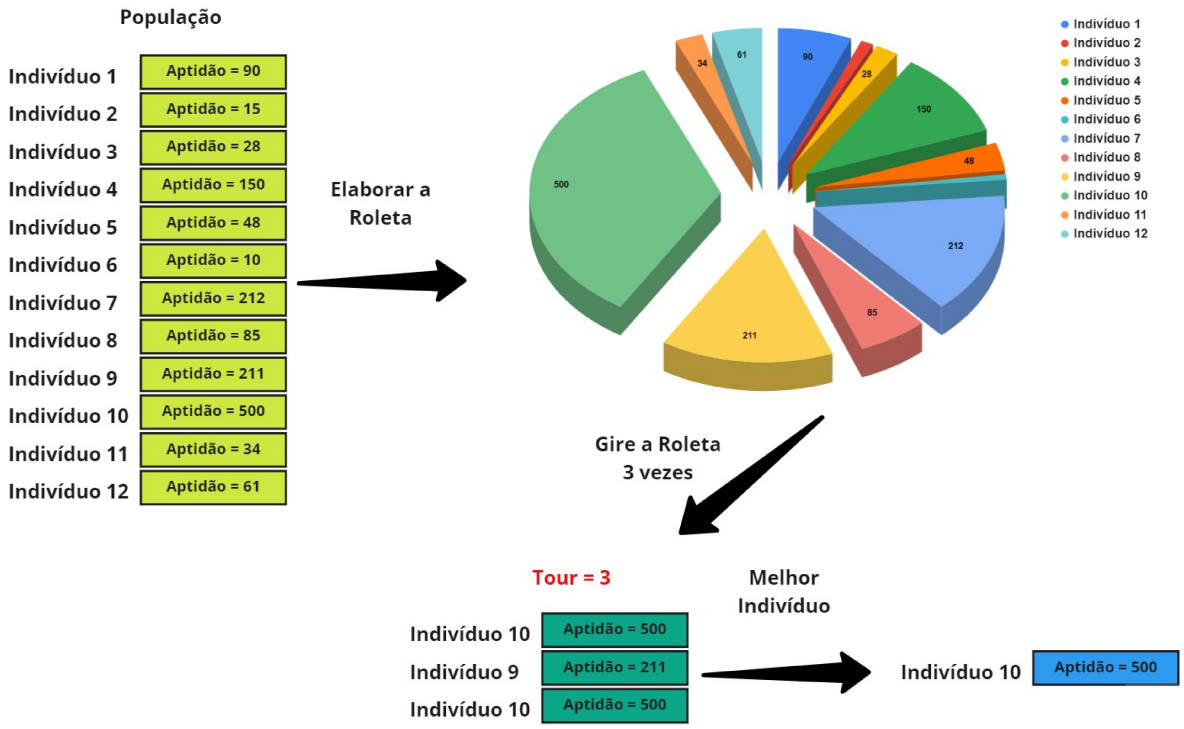


Fonte: Adaptado de Razali e Geraghty (2011).

probabilidade de serem selecionados. A grande diferença para o torneio estocástico é que esse método também irá considerar a função de aptidão do indivíduo para determinar a seleção, que nesse caso é feita através do método da roleta. A Figura 5 exemplifica a escolha do indivíduo através do método de seleção por torneio estocástico.

De forma geral, o método do torneio possui algumas vantagens e uma delas é a facili-

Figura 5 – Exemplo de seleção pelo método do torneio estocástico com $tour = 3$.



Fonte: autoria própria.

dade em alternar entre problemas de maximização e minimização. A diferença será apenas na hora de selecionar o melhor indivíduo: no problema de maximização o indivíduo com a maior função de aptidão será selecionado, enquanto que no problema de minimização será selecionado o indivíduo com a menor aptidão. Outra vantagem interessante é a facilidade em aumentar ou diminuir a pressão seletiva, simplesmente alterando o valor do *tour*.

2.1.3 Operadores de Reprodução

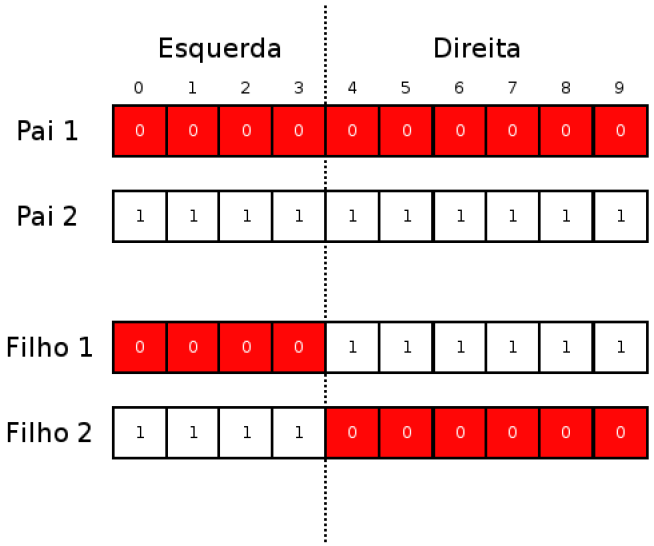
Nos AEs, os indivíduos de uma população são formados por uma codificação que representa uma solução. A codificação é baseada em uma abstração do problema e cada símbolo empregado caracteriza a solução candidata no espaço de busca. Portanto, é necessário definir mecanismos que explorem o espaço de busca atrás de outras soluções candidatas. Os mecanismos adotados são os operadores de recombinação e mutação, baseados nos conceitos biológicos de reprodução sexuada e assexuada, respectivamente.

2.1.3.1 Recombinação

Conforme destacado no nome, a recombinação (do inglês, *crossover*) é um operador que mistura o material genético de um par de genótipos (pais) em um, dois ou mais genótipos descendentes (filhos). A recombinação é um operador sempre estocástico: as

escolhas de quais partes de cada pai serão combinadas e como isso será feito dependem de fatores aleatórios. A combinação de um número maior de pais também é possível sendo denominada como operadores de recombinação com alta aridade, todavia, sem um correspondente biológico (EIBEN; SMITH, 2015).

Figura 6 – Exemplo de recombinação de um ponto.



Fonte: autoria própria.

A Figura 6 ilustra uma recombinação com uma única divisão. Ao recombinar os genes através de um ponto de corte, é possível criar duas novas soluções, ambas construídas com informações cruzadas de ambos os pais.

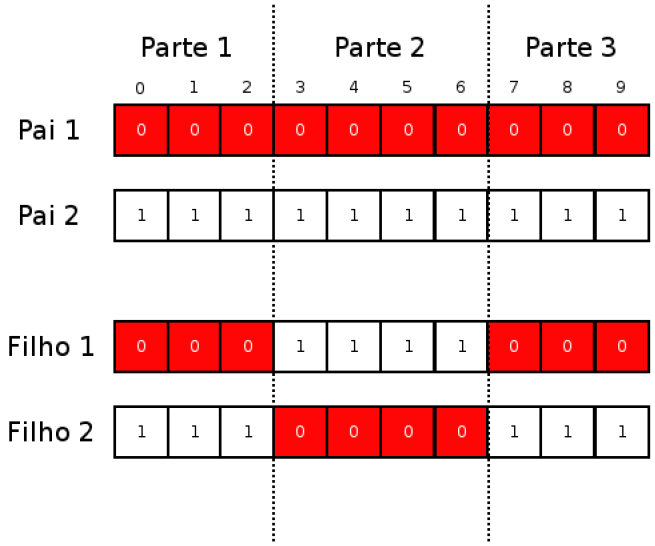
Adicionalmente, a Figura 7 apresenta uma recombinação que implementa n pontos de corte. Os pontos de corte são definidos arbitrariamente em ambos os modelos. Em resumo, os descendentes apresentados em cada recombinação representam soluções distintas apesar de possuírem os mesmos pais.

2.1.3.2 Mutação

A mutação é um operador unário que ao ser aplicado em um indivíduo gera um mutante modificado ou um descendente. De maneira análoga ao operador de recombinação, esse tipo de operador é estocástico, isto é, sua saída (o filho) dependerá de uma série de escolhas aleatórias, causando uma mudança não enviesada (EIBEN; SMITH, 2015).

A Figura 8 apresenta um exemplo de mutação binária. Nesse exemplo, um locus sorteado aleatoriamente tem seu respectivo gene modificado, de maneira semelhante ao modelo introduzido por Holland (1975). Em contrapartida, a Figura 9 contempla uma mutação permutada. Neste cenário, o indivíduo é formado por símbolos sem repetição.

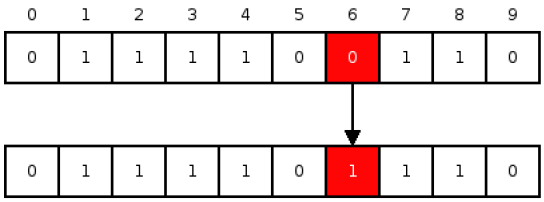
Figura 7 – Exemplo de recombinação de n pontos.



Fonte: autoria própria.

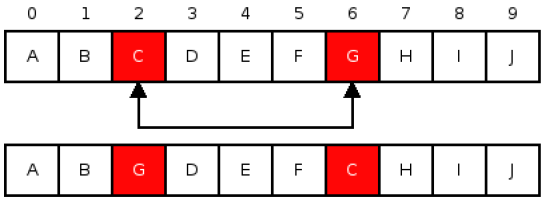
Logo, o procedimento adotado também efetua uma pequena modificação nos genes do indivíduo, todavia, sem produzir uma codificação inválida.

Figura 8 – Mutação aplicada em indivíduo com codificação binária .



Fonte: autoria própria.

Figura 9 – Mutação aplicada em indivíduo com codificação de permutação.



Fonte: autoria própria.

2.1.4 Reinserção

Após todos os filhos terem sido gerados, por meio dos operadores de reprodução, é necessário decidir quais deles farão parte de uma nova população para iniciar uma nova geração do AG. Essa etapa é chamada de reinserção e, assim como as etapas anteriores, possui vários métodos de execução:

Reinserção pura: toda a população antiga é substituída pelos filhos gerados. Pode não ser considerada uma boa escolha, já que muito material genético é perdido, inclusive bons indivíduos que podem não ser gerados novamente, piorando assim o resultado do AG.

Reinserção uniforme: nesse método, os melhores indivíduos da população total (pais e filhos) serão selecionados. A seleção desses melhores indivíduos é realizada através de qualquer um dos métodos de seleção citados neste trabalho (roleta, torneio, etc.) e consiste no método mais fiel à abstração que os AGs fazem da natureza, pois o fato do indivíduo ter a melhor aptidão não é garantia da sua sobrevivência. Esse método, geralmente, não permite repetição: o indivíduo só poderá ser selecionado uma única vez para a próxima geração.

Reinserção ordenada: a população total (pais e filhos) é ordenada e os n melhores indivíduos serão selecionados, sendo n o tamanho da população definida para o problema. Um ponto de atenção desse método é que ele provoca a perda de diversidade, uma vez que seleciona sempre os melhores indivíduos, descartando os piores. Essa particularidade pode causar uma convergência prematura.

Reinserção elitista: um percentual da população antiga (pais) é mantido para a próxima geração. É um método bastante utilizado na literatura, pois garante a troca constante do material genético dos indivíduos, uma vez que mescla a seleção entre pais e filhos, sem descartar os melhores indivíduos da população anterior.

2.2 Otimização Multiobjetivo

Segundo Barbosa, Ribeiro e Arantes (2010), métodos de otimização multiobjetivo têm duas metas principais: *i*) minimizar a *distância* entre a frente não dominada e a frente de Pareto ótimo; e *ii*) encontrar um conjunto de soluções diversificadas. Em contrapartida, os AEs buscam solucionar dois problemas: *i*) a avaliação e a seleção das soluções de forma a permitir uma busca eficiente para o conjunto de Pareto ótimo; e *ii*) uma forma de manter a diversidade da população evitando a convergência prematura. Assim, essa classe de algoritmos tem se mostrado adequada para lidar com diversos problemas do mundo real, uma vez que é bastante comum a necessidade de otimizar mais de uma função objetivo.

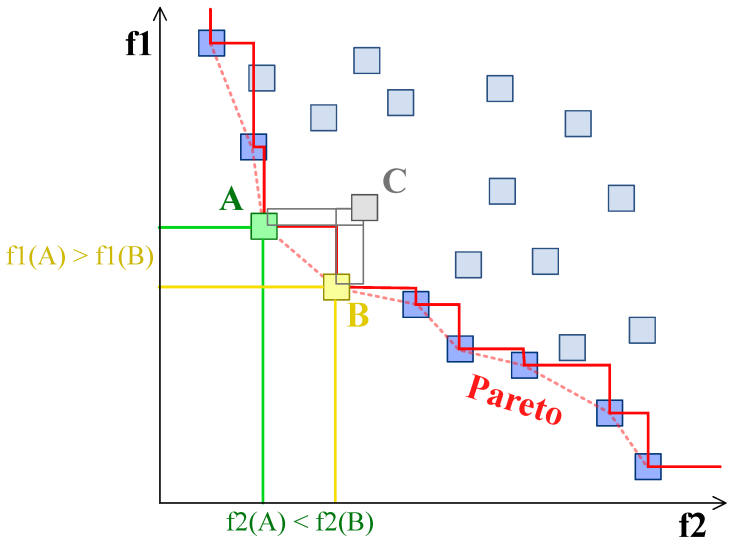
Nesse caso, não existe somente uma solução para o problema, mas sim um conjunto de soluções, denominado conjunto de Pareto-ótimo ou fronteira de Pareto (DEB et al., 2002).

O conceito de Pareto-ótimo constitui a origem da busca na otimização multiobjetivo. Pela definição, um vetor X pertence ao Pareto-ótimo, se não existe um outro vetor viável X^* que possa melhorar algum objetivo, sem causar uma piora em pelo menos um dos demais objetivos. Em outras palavras, um vetor solução X pertence ao conjunto de soluções Pareto-ótimo se não existe nenhum vetor solução X^* que domine X.

O conjunto das soluções não-dominadas em X é chamado de conjunto Pareto-ótimo. A imagem de um determinado conjunto Pareto-ótimo no espaço dos valores dos objetivos é chamada de fronteira de Pareto (KONAK; COIT; SMITH, 2006). A dominância de Pareto afirma que uma solução X é melhor que uma solução Y, ou que X domina Y, se, e somente se:

1. X é melhor que Y em pelo menos um dos objetivos avaliados;
2. X não é pior que Y em nenhum dos objetivos avaliados.

Figura 10 – Dominância e Fronteira de Pareto.



Fonte: Pareto... (2023)

A Figura 10 representa o conceito de dominância e fronteira de Pareto. Pegando como base os objetivos $f1$ e $f2$, ambos de minimização, tem-se:

- a solução C é dominada pelas soluções A e B, pois ela possui ambos os objetivos maiores (tanto $f1$ quanto $f2$);
- $f1(A) > f1(B)$, portanto a solução A não domina a solução B;

- $f2(A) < f2(B)$, portanto a solução B não domina a solução A;
- A e B não são dominadas por nenhuma outra solução, portanto fazem parte da Fronteira de Pareto.

2.3 Algoritmo Evolutivo Multiobjetivo

Como dito anteriormente, AGs são técnicas de busca baseadas na simplificação de alguns processos biológicos evolutivos. Seu objetivo é encontrar uma solução otimizada que resolva o problema proposto. Essa solução é classificada de acordo com uma função de aptidão (também chamada de função objetivo). É nesse ponto que ocorre a diferenciação em relação aos algoritmos evolutivos multiobjetivos (AEMOs). A principal diferença entre um algoritmo tradicional e sua variação multiobjetivo está na forma de cálculo da aptidão dos indivíduos da população. No caso dos algoritmos multiobjetivos, geralmente o conceito de dominância é usado de diferentes formas (BUENO; OLIVEIRA, 2010).

Um dos primeiros AEs empregados nesse contexto é o VEGA (do inglês, *Vector Evaluated Genetic Algorithm*), desenvolvido por Schaffer (1985). Esse algoritmo é um AG modificado que avalia cada objetivo separadamente. Uma das suas limitações é que as soluções da fronteira obtidas, em geral, possuem baixa diversidade. Posteriormente, Goldberg (1989) criou um procedimento que ordena as soluções com base no conceito de *dominância* (visto na Seção 2.2), fornecendo um valor de aptidão para uma solução proporcional ao número de soluções que esta domina, ou seja, não perde em nenhum objetivo e é melhor em pelo menos um. Com isso, soluções com maior número de dominados possuem maior aptidão.

Os dois algoritmos mais utilizados e bem conhecidos, com bom desempenho para problemas multiobjetivo (até 3 objetivos) são o *Non-Dominated Sorting Genetic Algorithm II* (NSGA-II) (DEB, 2001) e o *Strength Pareto Evolutionary Algorithm II* (SPEA-II) (ZITZLER; LAUMANNS; THIELE, 2001), ambos algoritmos evolutivos e custosos. O primeiro utiliza o ranking por não dominância e o *crowding distance* (ordenação de todos os indivíduos por cada objetivo), enquanto que o segundo calcula, para cada indivíduo, a distância para todos os demais.

A seguir, são apresentados mais detalhes do NSGA-II, algoritmo multiobjetivo implementado e utilizado neste trabalho. Esse algoritmo é um dos mais citados na literatura e tem apresentado bom desempenho para problemas com dois e três objetivos, sendo, portanto, adequado para as comparações planejadas aqui.

2.3.1 O *Non-Dominated Sorting Genetic Algorithm II*

O NSGA-II é um método de otimização multiobjetivo baseado em algoritmos genéticos para a geração do conjunto de soluções não dominadas (DEB, 2001; DEB et al., 2002). O

funcionamento desse AG é similar a um AG padrão com diferença no cálculo da aptidão. Esse cálculo é feito baseado em fronteiras (*ranks*) e no cálculo de distâncias (*crowding distance*).

Os *ranks* são as fronteiras em que a população é dividida. Eles são baseados no conceito de dominância entre as soluções, ou seja, na primeira fronteira estão presentes os indivíduos não-dominados, na segunda fronteira os indivíduos dominados somente por aqueles da primeira fronteira, na terceira fronteira os indivíduos dominados somente por aqueles da primeira e segunda fronteiras, e assim sucessivamente até finalizar a divisão de toda a população.

Como a divisão do indivíduo em fronteira depende exclusivamente de seu grau de dominância, vários indivíduos podem pertencer a uma mesma fronteira. Para discriminá-los é realizado o cálculo de distâncias, o que confere melhor avaliação às soluções mais espalhadas da mesma fronteira. O objetivo é garantir melhor diversidade na população.

O primeiro passo de execução do NSGA-II é gerar uma população $\mathcal{P}$ de tamanho n. Na sequência, usando os métodos de um AG típico: seleção (utiliza torneio simples com $tour = 2$), recombinação e mutação. Com isso, gera-se uma população de filhos (descendentes) do mesmo tamanho de $\mathcal{P}$. É feito uma união entre os conjuntos de pais e filhos gerando assim uma nova população de tamanho $2n$. A próxima etapa é o que diferencia esse AG: é feita a classificação dos indivíduos com base nas fronteiras de dominância e também é calculado as distâncias em cada fronteira (ela será utilizada como critério de desempate).

A nova população é ordenada com base nessa classificação e nesse cálculo de distância. Os descendentes finais a serem selecionados serão aqueles com o melhor grau de dominância. Se o limite do tamanho da população n for superado, será necessário eliminar os indivíduos com menor *crowding distance* da última frente selecionada. Se o critério de parada for atingido, o AG finaliza sua execução e o retorno do algoritmo será o conjunto dos indivíduos não-dominados da primeira fronteira. Caso contrário, a população definida após o corte será reinserida como população na próxima geração do AG e o processo será repetido até que o critério seja alcançado. O Algoritmo 11 descreve o fluxo de execução do NSGA-II.

2.3.2 Otimização de Muitos Objetivos

Os AEMOs são bastante utilizados de forma eficiente quando deseja-se trabalhar com até três objetivos. Contudo, para problemas com quatro ou mais objetivos, o número de soluções para aproximar o conjunto Pareto-ótimo aumenta exponencialmente e o processo de decisão pode ficar muito difícil (MENDES, 2014).

Problemas de otimização multiobjetivo com mais de três objetivos são comumente referidos como problemas de otimização de muitos objetivos (do inglês, *many-objetive optimization problems*). Normalmente, esta classe de problemas traz novos e complexos

Figura 11 – Pseudocódigo do algoritmo NSGA-II.

```
1  Pop ← INICIALIZA população com soluções candidatas aleatórias;
2  calcularRank(Pop);
3  calcularCrowdingDistance(Pop);
4  repita
5  |   Pais ← selecao(Pop);
6  |   Filhos ← crossover(Pais);
7  |   mutacao(Filhos);
8  |   Pop' ← Pop ⋃ Filhos;
9  |   calcularRank(Pop');
10 |   calcularCrowdingDistance(Pop');
11 |   Pop ← reinsercao(Pop');
12 até CONDIÇÃO DE PARADA satisfeita;
13 retorna População final de soluções
```

desafios aos métodos de otimização atuais, principalmente mantendo o equilíbrio correto entre convergência e diversidade. Durante os últimos anos, várias abordagens foram propostas para resolver problemas de muitos objetivos.

Muitos dessas abordagens focam no conceito de decomposição, ou seja, transforma um problema multiobjetivo em muitos problemas de otimização de objetivo único (ISHIBUCHI et al., 2011). Neste trabalho, focou-se no estudo de dois algoritmos que utilizam o conceito de subpopulações para decompor e resolver problemas de otimização. Esses dois algoritmos são descritos a seguir.

2.3.3 Multi-Objective Evolutionary Algorithm with Many Tables

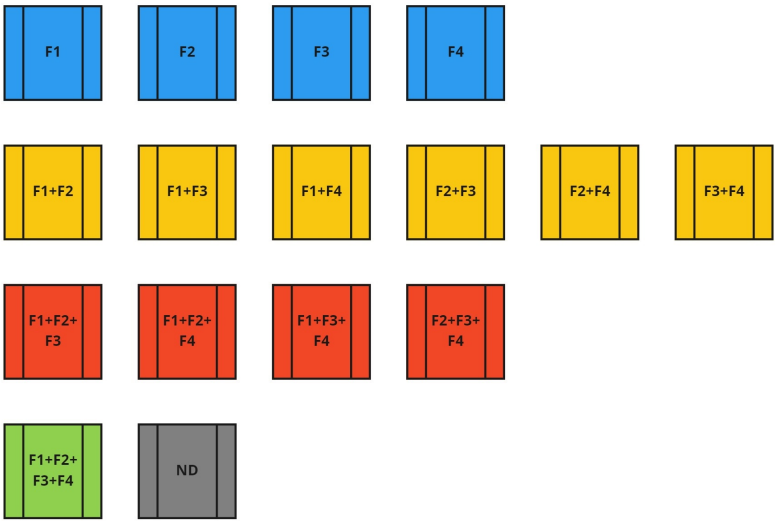
O *Multi-Objective Evolutionary Algorithm with Many Tables* (MEAMT) é um AEMO proposto inicialmente para predição de estrutura de proteínas (BRASIL, 2012; BRASIL; DELBEM; SILVA, 2013). Trata-se de uma adaptação do *Multi-Objective Evolutionary Algorithm in Tables* (MEAT), proposto para problemas de dois objetivos (SANTOS et al., 2010; GABRIEL; MELO; DELBEM, 2012). Esse algoritmo foi projetado para lidar eficientemente com vários objetivos através da combinação dos mesmos, realizando uma amostragem mais adequada do espaço das funções objetivos (BRASIL; DELBEM; SILVA, 2013). O algoritmo foi proposto inicialmente com o nome em português Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas (AEMMT) e será referenciado como tal neste trabalho.

O AEMMT trabalha decompondo o problema em subproblemas menores. Ele manipula um esquema de tabelas que representam uma combinação diferente dos objetivos utilizados. Suponha que o problema possua 4 objetivos ($F1$, $F2$, $F3$ e $F4$): serão criadas assim 15 tabelas a partir dessa combinação e mais uma tabela extra para guardar os indi-

víduos não-dominados, conforme pode ser visto na Figura 12. Nesse exemplo, as tabelas são dispostas da seguinte maneira:

- **Tabelas azuis:** representam os objetivos um a um;
- **Tabelas amarelas:** representam a combinação dos objetivos dois a dois;
- **Tabelas vermelhas:** representam a combinação dos objetivos três a três;
- **Tabela verde:** representa a combinação dos objetivos quatro a quatro;
- **Tabela cinza:** representa os indivíduos não-dominados.

Figura 12 – Subpopulações do AEMMT.



Fonte: Adaptado de Brasil, Delbem e Silva (2013).

O primeiro passo do algoritmo é gerar uma população inicial a fim de preencher todas as tabelas com indivíduos distintos uns dos outros. As tabelas possuem um tamanho fixo, logo o tamanho da população inicial vai depender desse tamanho de tabela. Suponha que as tabelas possuam um tamanho fixo de 30 indivíduos, considerando as 16 tabelas para os quatro objetivos do problema a população inicial seria de $16 \times 30 = 480$ indivíduos. Dessas soluções, seriam extraídas as melhores para o preenchimento das tabelas de acordo com a combinação dos objetivos que cada uma representa. O indivíduo dessa população só entra em uma tabela se ele for melhor que a pior solução presente na população dessa tabela, caso ela tenha atingido seu tamanho máximo. Para verificar se esse indivíduo é melhor que o pior existente na tabela é utilizado a média aritmética (função que transforma os múltiplos objetivos em um valor escalar).

A tabela dos não-dominados tem seu comportamento um pouco diferente. Ela irá guardar um novo indivíduo sempre que ele não for dominado por nenhum outro indivíduo da tabela. Uma particularidade dessa tabela é que seu tamanho pode ser diferente das demais. Foi utilizado neste trabalho o tamanho 100. Da mesma forma como as demais, sempre que seu tamanho é atingido, o pior indivíduo em relação a média aritmética é removido da tabela.

O número de gerações em que o AE será executado após o preenchimento inicial é uma configuração da execução. A cada geração são escolhidas duas tabelas através de um torneio duplo considerando suas pontuações. Vale ressaltar que a tabela dos não-dominados também participa do sorteio. Uma tabela tem sua pontuação inicial igual a zero e esse valor é incrementado sempre que ela fornecer um indivíduo que sobreviverá à geração. Para evitar que as mesmas tabelas sejam sempre escolhidas devido suas altas pontuações, esses valores são zerados a cada 100 gerações.

Após as escolhas das tabelas, será sorteado aleatoriamente um indivíduo de cada uma delas para realização do *crossover* e geração de um novo descendente. Esse filho gerado passará pelo mesmo processo que os indivíduos da população inicial: ele será comparado com os piores indivíduos de cada uma das tabelas para definir se ele tomará ou não seu lugar naquela tabela. Após a execução de todas as gerações, o AG tem sua execução finalizada e o resultado é o conjunto dos indivíduos não-dominados. Um detalhe importante sobre o AEMMT é que cada geração produz apenas um descendente, portanto ele precisa de um número maior de gerações para alcançar o mesmo número de soluções avaliadas que outros AGs. O pseudocódigo do AEMMT é apresentado no Algoritmo 13.

Figura 13 – Pseudocódigo do algoritmo AEMMT.

```
1  Tabelas ← criarTabelas(qtdObjetivos);
2  Pop_0 ← INICIALIZA população com soluções candidatas aleatórias;
3  popularTabelas(tabelas, Pop_0);
4  repita
5  |  resetarScoreTabelas(Tabelas);
6  |  Tabela1 ← torneioDuplo(tabelas);
7  |  Tabela2 ← torneioDuplo(tabelas);
8  |  Filho ← crossover(Tabela1, Tabela2);
9  |  mutacao(Filho);
10 |  popularTabelas(tabelas, Filho);
11 |  adicionarScoreTabela(Tabela1);
12 |  adicionarScoreTabela(Tabela2);
13 até CONDIÇÃO DE PARADA satisfeita;
14 retorna Indivíduos da tabela de não-dominados
```

2.3.4 Multi-Objective Evolutionary Algorithm based on Non-dominated Decomposed Sets

O *Multi-Objective Evolutionary Algorithm based on Non-dominated Decomposed Sets* (MEANDS) foi proposto como uma evolução do AEMMT. O novo algoritmo mantém as características de *crossover*, mutação e seleção, porém acrescenta duas modificações ao modelo: a formação e a pontuação das tabelas (LAFETÁ et al., 2018). O algoritmo foi proposto inicialmente com o nome em português Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias (AEMMD) e será referenciado como tal neste trabalho.

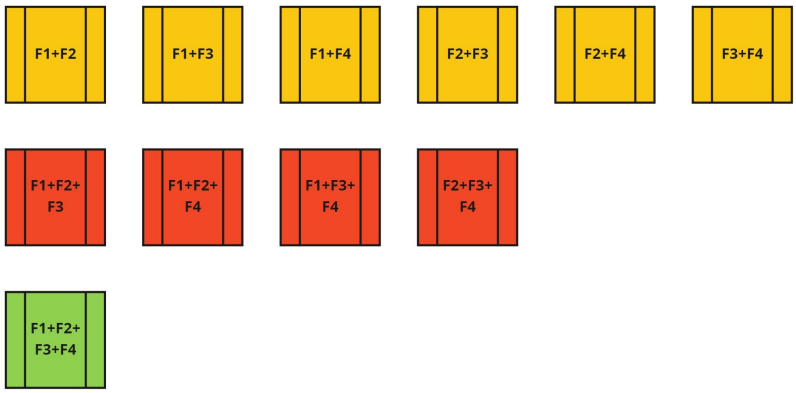
No AEMMD, ao invés de utilizar a média aritmética em relação aos objetivos da tabela para definir se um indivíduo entrará ou não naquela tabela, usa-se o conceito de dominância. Ou seja, um novo indivíduo só entrará na tabela se ele não for dominado por nenhum outro indivíduo já presente, considerando apenas os objetivos que ela representa. Esse modelo faz com que as tabelas criadas para execução do AG sejam diferentes daquelas presentes no AEMMT. As tabelas relativas aos objetivos individuais foram eliminadas uma vez que não faz sentido o conceito de dominância em apenas um objetivo. A tabela de não-dominância específica presente no AEMMT também foi eliminada, uma vez que todas as tabelas do AEMMD possuem o conceito de não-dominância. Suponha que o problema possua 4 objetivos ($F1$, $F2$, $F3$ e $F4$): serão criadas assim 11 tabelas a partir da combinação dos objetivos (2 a 2, 3 a 3 e 4 a 4), conforme pode ser visto na Figura 14. Nesse exemplo, as tabelas são dispostas da seguinte forma (todas elas considerando indivíduos não-dominados):

- **Tabelas amarelas:** representam a combinação dos objetivos dois a dois;
- **Tabelas vermelhas:** representam a combinação dos objetivos três a três;
- **Tabela verde:** representa a combinação dos objetivos quatro a quatro.

A outra diferença desse AG fica por conta da pontuação das tabelas que é realizada por convergência e não por contribuição. No AEMMT, a tabela recebedora do ponto é aquela que forneceu o indivíduo para realização do *crossover* e produziu um descendente sobrevivente para a próxima geração. Já no AEMMD, a recebedora do ponto é aquela tabela que recebeu o filho, ou seja, sempre que um indivíduo é inserido em uma tabela, sua pontuação é incrementada. Além disso, as tabelas do AEMMD não possuem tamanho fixo e suas pontuações não são reiniciadas em nenhum momento.

O primeiro passo do algoritmo é gerar uma população inicial a fim de preencher todas as tabelas com indivíduos distintos uns dos outros. Como as tabelas não possuem tamanho fixo, a quantidade de indivíduos da população inicial é gerada de forma aleatória (pode ser uma configuração do AG). Todos os indivíduos da população são avaliados em relação a dominância para o preenchimento das tabelas. Um indivíduo só entra em uma tabela

Figura 14 – Subpopulações do AEMMD.



Fonte: Adaptado de Brasil, Delbem e Silva (2013) e Lafetá et al. (2018).

se ele não for dominado por nenhum outro indivíduo presente na tabela. Uma vez que ele entrou, todos os indivíduos que são dominados por ele são removidos.

Assim como no AEMMT, a cada geração são escolhidas duas tabelas através de um torneio duplo considerando suas pontuações, nesse modelo pontuação por convergência e não por contribuição. Após as escolhas das tabelas, será sorteado aleatoriamente um indivíduo de cada uma delas para realização do *crossover* e geração de um novo descendente. Esse filho gerado passará pelo mesmo processo que os indivíduos da população inicial: ele será comparado com todos os indivíduos de cada uma das tabelas e só entrará nelas se não for dominado por nenhum deles. Após a execução de todas as gerações, o AG tem sua execução finalizada e o resultado é o conjunto da tabela que considera todos os indivíduos no cálculo. Assim como o AEMMT, cada geração produz apenas um descendente, portanto, o AEMMD também precisa de um número maior de gerações para alcançar o mesmo número de soluções avaliadas em outros AGs. O algoritmo Figura 15 apresenta, em pseudocódigo, o funcionamento do AEMMD.

Figura 15 – Pseudocódigo do algoritmo AEMMD.

```
1  Tabelas ← criarTabelas(qtdObjetivos);
2  Pop_0 ← INICIALIZA população com soluções candidatas aleatórias;
3  popularTabelas(tabelas, Pop_0);
4  repita
5  |  Tabela1 ← torneioDuplo(tabelas);
6  |  Tabela2 ← torneioDuplo(tabelas);
7  |  Filho ← crossover(Tabela1, Tabela2);
8  |  mutacao(Filho);
9  |  popularTabelas(tabelas, Filho);
10 até CONDIÇÃO DE PARADA satisfeita;
11 retorna Indivíduos da tabela que considera a combinação de todos os objetivos
```

Capítulo 3

Escalonamento de Tarefas em Sistemas Multiprocessados

Este capítulo apresenta conceitos necessários para compreensão das técnicas estudadas e utilizadas no desenvolvimento deste trabalho. A seção 3.1 será responsável por introduzir, de forma resumida, os conceitos básicos essenciais para a compreensão do estudo realizado. Serão apresentados conceitos gerais sobre escalonamento de tarefas e algoritmos evolutivos. A seção 3.2 será responsável por apresentar os trabalhos correlatos.

3.1 Definições

Problemas de escalonamento são geralmente associados a aplicações como planejamento de produção e transporte, programação de horários, roteirização de veículos, escalação de equipe, planejamento de eventos, etc., que requerem determinados recursos (trabalhadores, ferramentas, energia, CPUs, memória, largura de banda, etc.) de forma que certos critérios de otimização e/ou viabilidade sejam atendidos (DROZDOWSKI, 2009).

No contexto deste trabalho, o escalonamento de tarefas se traduz na atribuição eficiente de aplicações paralelas sobre os recursos disponíveis em um sistema distribuído (CASAVANT; KUHL, 1988; DONG; AKL, 2006). Nesse sentido, Chapin e Weissman (2004) estabelecem os seguintes componentes do problema:

- ❏ Um conjunto de tarefas que devem ser executadas;
- ❏ Um conjunto de recursos que executam as tarefas;
- ❏ Um conjunto de restrições que devem ser obedecidas;
- ❏ Um conjunto de objetivos utilizados para medir o desempenho do escalonador.

As restrições e os objetivos são definidos na formulação do problema. As restrições são usadas para definir a viabilidade de encontrar uma solução, enquanto que os objetivos

definem a qualidade da solução. É obrigatório que as restrições sejam atendidas para o projeto de um algoritmo coerente com o problema. Já os objetivos devem ser alcançados com o intuito de formar melhores soluções. As definições de restrições e objetivos podem ser associadas as tarefas, recursos, métricas de desempenho ou pela combinação desses elementos. Na modelagem do problema, é comum definir restrições e objetivos como equivalentes, todavia, essas propriedades devem ser tratadas de formas diferentes na resolução do problema.

Existem variados tipos de restrições. Tarefas que devem ser executadas antes de outras tarefas, definindo uma hierarquia de execução, são exemplos de restrições de precedência. Por exemplo, uma impressora só pode começar a imprimir documentos após a finalização ou autorização do software que a comanda. Restrições temporais definem quando recursos ou tarefas poderão ser executadas. Por exemplo, rotinas de backup só poderão ser executadas fora de expediente e durante a madrugada, ou a impressora só pode imprimir documentos no período da tarde.

A qualidade das soluções pode ser medida através de métricas fundamentadas nos objetivos. O objetivo mais comum é a minimização do tempo de execução das tarefas (*makespan*). Outros exemplos de objetivos são: eficiência no uso de recursos, minimização do custo, prioridade entre tarefas, minimização do atraso na comunicação, dentre outros. A política de escalonamento para um sistema multiprocessado geralmente incorpora uma combinação de alguns desses critérios (CHAPIN; WEISSMAN, 2004). Alguns problemas são, ainda, construídos com base na combinação de objetivos, muitas vezes conflitantes. A possibilidade de que esses objetivos combinados possam ser conflitantes aumenta a medida que novos objetivos são acrescentados ao problema. Por exemplo, um algoritmo que deseja reduzir o tempo de execução do projeto deve observar que o custo final do projeto tende a aumentar.

Neste trabalho, considera-se o problema de escalonamento de tarefas com as características descritas a seguir.

Seja $P = \{p_1, \ldots, p_m\}$ um conjunto de m processadores. Esses processadores podem ser homogêneos (ou seja, idênticos, com a mesma velocidade de processamento) ou heterogêneos (diferentes velocidades de processamento devido as variações de hardware). Os processadores também possuem memória própria e são fortemente conectados em rede, ou seja, não apresentam restrições de conexão (GOLUB; KASAPOVIC, 2002).

Sobre esses processadores, será atribuído um conjunto de n tarefas, $T = \{t_1, \ldots, t_n\}$. As tarefas não são preemptivas, ou seja, ao serem associadas a um processador, devem ser executadas sem interrupções até o final (GOLUB; KASAPOVIC, 2002). A principal restrição empregada aqui é a *ordem de precedência* das tarefas. Dessa maneira, o mapeamento das tarefas e suas precedências pode ser representado através de um grafo acíclico dirigido (DAG, do inglês *Directed Acyclic Graph*). Dessa forma, um DAG $G(T, E)$ representa um programa paralelo com suas tarefas, onde T é um conjunto de vértices e E é um

conjunto de arestas. Cada vértice $t_i \in T$ representa uma tarefa e consiste em um conjunto de instruções que devem ser executadas sequencialmente. As arestas representam a ordem de precedência entre as tarefas, de forma que o par ordenado (t_i,t_j) indica que a tarefa t_j é filha de t_i, ou seja, t_j só poderá ter sua execução iniciada após a conclusão de, no mínimo, t_i (OMARA; ARAFA, 2010). Tarefas que não possuem pais são nomeadas de tarefas (ou vértices) de entrada, enquanto que tarefas que não possuem filhos são denominadas tarefas (ou vértices) de saída.

Pesos podem ser associados aos vértices, representando o número de ciclos de computação (carga de trabalho) necessários para cada tarefa. Arestas também podem ter pesos, indicando o custo de comunicação entre pares de tarefas (ROBERT, 2011). Esse custo define o tempo que a tarefa sucessora deverá aguardar para ser iniciada após o término da tarefa predecessora. Dessa forma, o custo de comunicação da aresta (t_i, t_j) é associado quando as tarefas são designadas a diferentes processadores. Em contrapartida, quando as tarefas são atribuídas ao mesmo processador, o custo de comunicação é nulo (OMARA; ARAFA, 2010).

Na versão do problema estudada aqui, não são consideradas restrições como prazo de execução (*deadlines*), ou seja, todas as tarefas podem ser executadas sem exceder um limite máximo de tempo. Finalmente, também não há tempo de liberação (*release time*); isso significa que todas as tarefas estão disponíveis no mesmo instante. A Figura 16 apresenta um DAG com custo de comunicação onde cada vértice apresenta a identificação da tarefa juntamente ao seu respectivo custo computacional. Já na Figura 17, tem-se um exemplo de atribuição, considerando dois processadores.

Conforme mencionado anteriormente, a qualidade de uma solução de escalonamento pode ser medida de diferentes maneiras. Em uma revisão sistemática da literatura, Silva (2020) listou diversas métricas, sendo que as mais comuns são:

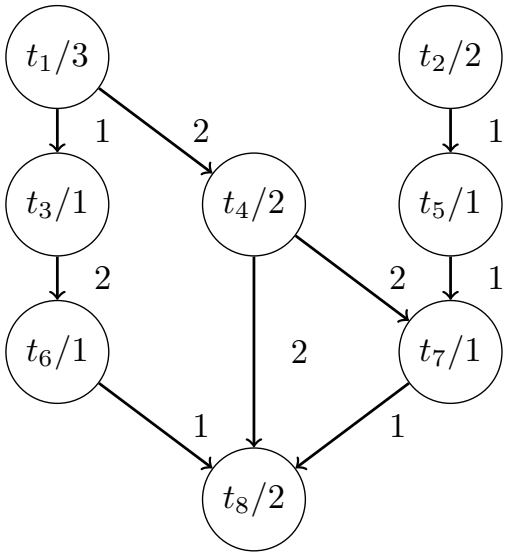
Makespan (M): Também conhecida como tempo total de conclusão (do inglês, *total completion time*), é uma das métricas mais comuns, presente em grande parte dos trabalhos relacionados. Pode ser obtida computando-se o tempo de finalização da última tarefa executada ou da finalização do último processador em execução. A Equação (2) apresenta o cálculo do *makespan*, sendo que $ftp(p_j)$ representa o tempo de finalização de um processador $p_j \in P$.

$$M = \max(ftp(p_j)), j = 1, \ldots, m \tag{2}$$

A fim de ilustrar essa métrica, considere a Figura 17. Nela, tem-se os seguinte valores: $ftp(p_1) = 8$ e $ftp(p_2) = 12$, o que resulta em um *makespan* $M = 12$.

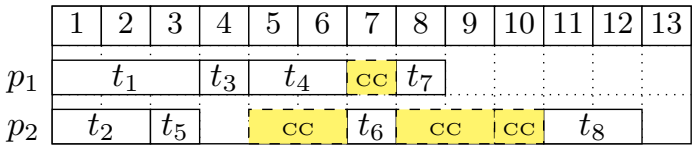
Balanceamento de carga (L): Outra forma de avaliar uma solução de escalonamento é observar se há uma distribuição adequada de tarefas nos processadores do ambiente. Essa distribuição pode ser avaliada pelo balançamento de carga (do inglês, *load*

Figura 16 – Exemplo de DAG com pesos, representando uma aplicação composta por oito tarefas.



Fonte: Adaptado de Silva e Gabriel (2020).

Figura 17 – Exemplo de escalonamento do DAG da Figura 16 em um ambiente composto por dois processadores. Nessa figura, *cc* representa o custo de comunicação.



Fonte: Adaptado de Silva e Gabriel (2020).

balancing), mostrado na Equação (3), onde avg é a média dos tempos de finalização dos processadores do conjunto (OMARA; ARAFA, 2010).

$$L = \frac{\max(ftp(p_j))}{avg}, j = 1, \ldots, m \tag{3}$$

Novamente em relação à Figura 17, o valor médio dos tempos de finalização é $avg = \frac{12+8}{2} = 10$; logo, o valor do balanceamento de carga é $L = \frac{12}{10} = 1{,}2$.

Flowtime (F): Assim como o *makespan*, essa métrica também está associada ao tempo. Kaur, Chhabra e Singh (2010) definem o *flowtime* como a soma dos tempos de finalização de todas as tarefas. A Equação (4) apresenta o cálculo do *flowtime*,

sendo que $ftt(t_i)$ é o tempo de finalização de uma tarefa $t_i \in T$.

$$F = \sum_{i=1}^{n} ftt(t_i) \tag{4}$$

No exemplo da Figura 17, somando todos os tempos de conclusão, tem-se o valor $F = 45$.

Custo de comunicação (C): Além de ser proeminente em boa parte dos cenários abordados, o custo de comunicação também pode ser utilizado na otimização. Dessa forma, uma minimização do custo de comunicação significa a redução de espera de tarefas que necessitam de dados de suas tarefas predecessoras.

Na Figura 17, há custos de comunicação entre as tarefas t_5 e t_7 (1 unidade de tempo), t_3 e t_6 (2 unidades) e t_4 e t_8 (2 unidades) e, finalmente, entre t_7 e t_8 (1 unidade de tempo). Logo, o custo de comunicação total é igual a $C = 6$.

Tempo de espera (W): Embora não seja descrita diretamente no trabalho de Silva (2020), o tempo de espera (do inglês, *waiting time*) é outra medida que tem se mostrado relevante. Basicamente, descreve o tempo que uma tarefa precisa aguardar para ser executada após ser designada a um processador, sendo, portanto, influenciada diretamente por fatores como o custo de comunicação e a carga de trabalho do processador. Trata-se de uma medida de qualidade de serviço, computada a partir da diferença entre o tempo de inicio da execução de uma tarefa t_i e o maior tempo final de execução de sua(s) tarefa(s) predecessora(s). Geralmente, considera-se o tempo de espera total, ou seja, a soma dos tempos de espera individuais calculados para cada uma das tarefas.

Nesse sentido, no exemplo anterior (Figura 17), considerando apenas o processador p_1, as tarefas t_3 e t_4 podem iniciar sua execução logo após a finalização de t_1 (predecessora de ambas). Porém, como ambas são atribuídas ao mesmo processador e t_3 é executada antes, t_4 deve aguardar sua finalização, ou seja, o tempo de espera de t_4 é igual a 1 unidade de tempo. Já a tarefa t_6 foi designada ao processador p_2, o que implicou um custo de comunicação; logo, seu tempo de espera, após a finalização de t_3 é de 2 unidade de tempo.

A complexidade do problema de escalonamento de tarefas pode ser analisada através de uma relação entre processadores, tempo de processamento das tarefas e restrições de precedência (GOLUB; KASAPOVIC, 2002). Nesse sentido, Afrati, Papadimitriou e Papageorgiou (1988) demostram que esse problema é $\mathcal{NP}$-Difícil para dois ou mais processadores, independente da topologia do DAG. Além disso, Omara e Arafa (2010) argumentam que esse é um dos problemas mais desafiadores da computação paralela. Algoritmos ruins podem falhar em explorar o verdadeiro potencial de um sistema paralelo,

limitando os ganhos do paralelismo devido ao custo excessivo de comunicação e subutilização dos recursos (DHINGRA; GUPTA; BISWAS, 2014). Por essa razão, esse problema tem sido tratado por diferentes abordagens algorítmicas.

3.2 Abordagens de Escalonamento de Tarefas

A literatura na área oferece várias abordagens para lidar com o problema de escalonamento de tarefas. Tais abordagens, geralmente, são classificadas em dois paradigmas: abordagens determinísticas e não-determinísticas (SACHDEVA; PANWAR, 2015). A abordagem determinística trabalha com algoritmos que, a partir de uma entrada específica, apresentam a mesma solução. Alguns exemplos da abordagem determinística são: escalonamento por lista (ARABNEJAD; BARBOSA, 2014), escalonamento por agrupamento (GERASOULIS; YANG, 1992) e escalonamento por duplicação de tarefa (JIN; SCHIAVONE; TURGUT, 2007).

Já a abordagem não-determinística contempla algoritmos com a capacidade de gerar diferentes saídas dependendo das escolhas efetuadas durante a execução. Essa abordagem é constantemente utilizada em problemas combinatórios de um modo geral, onde a busca por soluções ótimas pode enfrentar longas explorações no espaço de soluções. Técnicas como algoritmos evolutivos (OMARA; ARAFA, 2010), otimização por colônia de formigas (CHEN; CHENG, 2005), colônia artificial de abelhas (LI; PAN, 2015), otimização por enxame de partículas (SIVANANDAM; VISALAKSHI; BHUVANESWARI, 2007), recozimento simulado (NANDA; DEGROOT; STENGER, 1992), busca tabu (PORTO; RIBEIRO, 1995) e algoritmo de busca cuco (MARICHELVAM; PRABAHARAN; YANG, 2014) são exemplos de abordagens não-determinísticas.

Entre as abordagens não-determinísticas, os algoritmos evolutivos (AEs) estão entre as mais empregadas (SILVA, 2020; SILVA; GABRIEL, 2020). Em um trabalho bastante referenciado, Hou, Hong e Ansari (1994) projetaram um AG, definindo conceitos que seriam adotados posteriormente, como a codificação dos indivíduos e o cálculo do *makespan* a partir dessa codificação. Wall (1996) também trouxe significativas contribuições para a área, inserindo o problema de escalonamento dentro de uma taxonomia bastante conhecida. Outros trabalhos (OMARA; ARAFA, 2010; SINGH; SINGH, 2012) focaram em estudar novas representações para os indivíduos e em heurísticas para melhorar o processo de busca.

Mais recentemente, muitos trabalhos têm focado no uso comercial de sistemas distribuídos, uma questão crescente devido ao aumento na geração de *workflows* científicos e no uso de ambientes de nuvens computacionais (LIEW et al., 2017; KUMAR et al., 2019). Assim, tem-se tido um maior foco na busca por soluções que utilizam dois ou mais objetivos. Muitos desses trabalhos consideram uma soma ponderada desses objetivos (DOĞAN; ÖZGÜNER, 2005; CHITRA; VENKATESH; RAJARAM, 2011). Entretanto, abordagens

evolutivas considerando fronteira de Pareto também têm sido empregadas (GUZEK et al., 2014; AHMAD; SHEIKH, 2019; SHEIKH; AHMAD; ARSHAD, 2019; WANGSOM; LAVANGNANANDA; BOUVRY, 2019), tendo como foco, porém, cenários com apenas dois ou três objetivos.

Capítulo 4

Desenvolvimento

O objetivo deste trabalho é desenvolver e comparar algoritmos genéticos multiobjetivo para o problema do escalonamento de tarefas considerando, simultaneamente, diversas métricas de desempenho. Esse problema envolve o mapeamento de um grafo acíclico dirigido (DAG), que representa uma coleção de tarefas computacionais e suas relações de precedência, sobre um ambiente de processamento paralelo. Para o desenvolvimento dos algoritmos, é necessário obedecer algumas regras que fazem parte da representação do problema:

- O DAG é composto por n nós (tarefas) $\{t_1, t_2, \ldots, t_n\}$;
- A tarefa pode ser executada apenas em um processador, sem preempção;
- O custo computacional de uma tarefa é denotado pelo peso desse nó;
- O grafo contém arestas dirigidas que representam a restrição de precedência (um nó filho não pode ser executado antes de seu nó pai ser finalizado);
- O peso da aresta é o custo de comunicação que só ocorre se os extremos dessas arestas forem executados em processadores diferentes.

Neste capítulo, são descritas as decisões de projeto tomadas no desenvolvimento desses algoritmos. Há diversas implementações de AGs presentes na literatura, principalmente na otimização de um único objetivo. Assim, várias das decisões tomadas aqui baseiam-se em trabalhos relacionados.

4.1 Representação do Indivíduo

A representação de um indivíduo, ou seja, a codificação do cromossomo, é uma das etapas mais críticas no desenvolvimento de um AG. Essa representação pode facilitar muito a implementação do AG, além de ter impacto direto em seu desempenho.

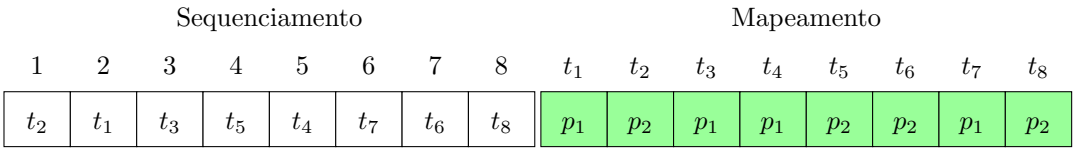
Neste trabalho, foi empregada a representação *Matching Scheduling Encoding* (MSE), proposta originalmente por Wang et al. (1997) e amplamente utilizada na literatura (SILVA, 2020). Especificamente, seguiu-se aqui o trabalho de Omara e Arafa (2010), que descreve em detalhes essa representação. Nesta implementação, o indivíduo é dividido em dois segmentos:

1. **Sequenciamento**: do inglês, *scheduling*, representa a ordem em que as tarefas serão processadas;

2. **Mapeamento**: do inglês, *mapping*, traz o índice dos processadores aos quais as tarefas são atribuídas.

A Figura 18 representa um indivíduo codificado na MSE. O DAG representado é o mesmo mostrado na Figura 16 (Seção 3.1), ou seja, um conjunto de oito tarefas $T = \{t_1, \ldots, t_8\}$. Considera-se, também, um ambiente computacional com apenas dois processadores, ou seja, $P = \{p_1, p_2\}$. Analisando os dois segmentos, observa-se que o processador p_1 executará as tarefas t_1, t_3, t_4 e t_7 (mapeamento), de acordo com a ordem definida no sequenciamento. Já p_2 executará, em ordem, as tarefas t_2, t_5 e t_8.

Figura 18 – Representação da codificação do indivíduo.



Fonte: Adaptado de Silva (2020).

Para obtenção do sequenciamento, utiliza-se a *ordenação topológica* do DAG. É importante destacar que um mesmo DAG pode ter mais de uma ordenação válida; assim, é possível gerar diferentes sequenciamentos a partir de um mesmo grafo direcionado. Já o mapeamento pode ser gerado de maneira aleatória. Com isso, a MSE garante que as soluções geradas na população inicial são sempre factíveis, ou seja, todos os indivíduos reprentam um escalonamento válido.

4.2 Operadores de Reprodução

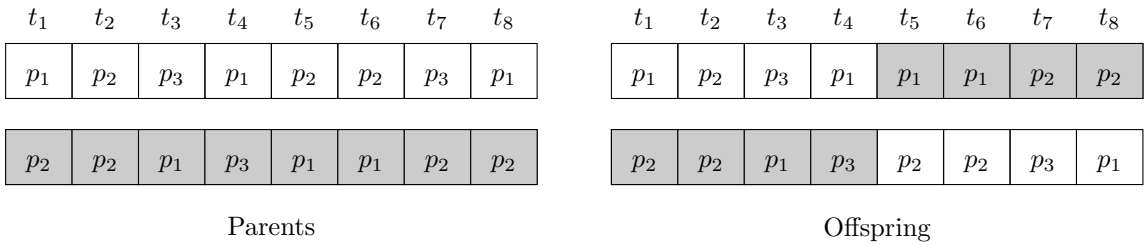
Uma vez definido a representação dos indivíduo, pode-se projetar os operadores de reprodução (recombinação e mutação). Novamente, tomou-se como base o trabalho de Omara e Arafa (2010). Os operadores implementados são descritos a seguir.

4.2.1 Recombinação

Omara e Arafa (2010) propuseram dois operadores de recombinação, denominados *crossover map* e *order crossover*. Esses operadores não são aplicados simultaneamente; em vez disso, após a seleção dos pais, sorteia-se um valor aleatório $rn \in [0, 1]$. Se $rn < 0.5$ então será executado o primeiro operador; caso contrário, executa-se o segundo.

A Figura 19 ilustra o modo de execução do operador *crossover map*. Inicialmente, é sorteado um ponto de corte *pc* de forma aleatória considerando o número total de tarefas do DAG. Nesse exemplo, foi utilizado $pc = 4$. Esse operador impacta diretamente no vetor segmento de mapeamento do cromossomo, alterando o índice dos processadores. Os valores à esquerda do ponto de corte são mantidos para ambos os pais e os valores à direita são trocados para geração de dois novos descendentes. A recombinação *crossover map* gera dois novos descendentes. Além do mais, ambos os filhos tem seus segmentos de sequenciamento mantidos sem alteração, ou seja, o Filho 1 tem o vetor de sequenciamento copiado do Pai 1 e o Filho 2 tem copiado do Pai 2.

Figura 19 – Exemplo de *crossover* denominado *Crossover Map*.



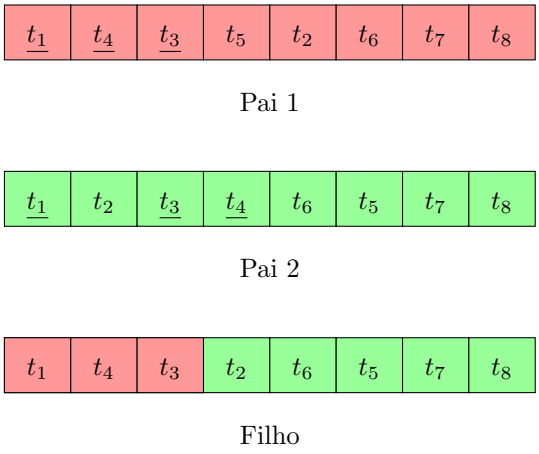
Fonte: Adaptado de Silva (2020).

O outro operador (*order crossover*) é ilustrado na Figura 20. Esse operador é aplicado apenas sobre o vetor de sequenciamento, alterando a ordem de processamento das tarefas. Neste caso, também ocorre o sorteio de um ponto de corte de forma aleatória, considerando o tamanho total de tarefas do DAG. No exemplo, é utilizado $pc = 3$. O filho gerado é composto pelos valores à esquerda (em rosa) do ponto de corte do Pai 1 e pelos valores restantes (em verde) do Pai 2 na ordem em que eles aparecem com exceção dos já copiados anteriormente do primeiro pai. Essa recombinação gera apenas um descendente que tem o vetor de mapeamento todo copiado do Pai 1.

4.2.2 Mutação

A forma como a mutação é realizada é bastante simples e ocorre apenas no vetor de mapeamento. Assim como o *crossover map*, o intuito desse operador de reprodução

Figura 20 – Exemplo de *crossover* denominado *Order Crossover.*

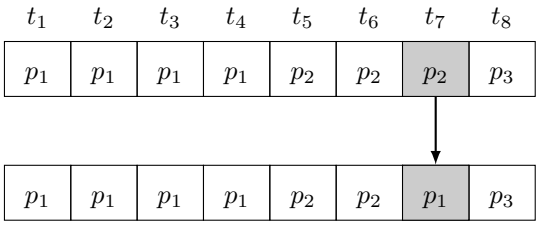


Fonte: Adaptado de Omara e Arafa (2010).

é alterar o índice do processador em que uma tarefa será executada, dessa forma, será gerada diversidade na população. Nesse operador são sorteados dois valores aleatórios: o primeiro é o índice da tarefa que terá seu processador alterado e o segundo é o índice do novo processador ao qual será designada a tarefa escolhida pelo primeiro sorteio.

A Figura 21 apresenta um exemplo de execução da mutação. O primeiro valor sorteado foi o número 7 que representa a tarefa t_7. De acordo com o vetor mapeamento, a mesma seria executada no processador p_2. Com o segundo valor sorteado sendo o número 1, a tarefa t_7 passa a ser executada pelo processador p_1, produzindo, assim, um indivíduo diferente do anterior. O vetor de sequenciamento não é alterado.

Figura 21 – Exemplo de mutação aplicada ao indivíduo.



Fonte: Adaptado de Silva (2020).

4.3 Reinserção

Ainda em relação ao trabalho de Omara e Arafa (2010), as autoras optaram pelo método de reinserção elitista, ou seja, uma parte da população antiga é mantida para a próxima geração. Nesse AG não existe um percentual exato, pois, como existem dois operadores de recombinação, a população final não terá um tamanho fixo (lembrando que um dos operadores produz dois filhos, enquanto o outro produz apenas um). Em todo caso, todos os filhos são mantidos para a próxima geração e o percentual restante é obtido através dos melhores pais.

Os AEMOs propostos também possuem sua própria estratégia de reinserção e elas foram descritas na seção 2.3. Recapitulando: o NSGA-II concatena as populações de pais e filhos e antes de iniciar a próxima geração, faz a ordenação dos indivíduos com base em seus *ranks* e *crowding distance* e elimina os piores indivíduos.

Já a proposta do AEMMT e do AEMMD é bem diferente. Eles não fazem a reinserção da população para uma nova geração. Os indivíduos incluídos nas tabelas do modelo ao fim da geração continuam presentes nas tabelas para a próxima geração.

4.4 Funções de Aptidão

A função de aptidão (*fitness*) é responsável por avaliar a qualidade do indivíduo enquanto solução para o problema. No trabalho mencionado anteriormente (OMARA; ARAFA, 2010), que é utilizado como base para as implementações descritas nesta monografia, o objetivo considerado é a minimização do *makespan*. Porém, comumente, AGs consideram a maximização do valor de *fitness*, assim, a Equação (5) mostra como computar o valor da aptidão (denotado por V_1) a partir do valor de *makespan* (denotado por M).

$$V_1 = \frac{1}{M} \tag{5}$$

Como exemplo, considere a solução codificada na Figura 18. Sabe-se que o *makepsan* dessa solução é $M = 12$ (ver Figura 17), assim, seu valor do *fitness* será $V_1 = \frac{1}{12} \approx 0.83$. É importante destacar que as cinco métricas de desempenho adotadas neste trabalho (ou seja, *makespan*, balanceamento de carga, *flowtime*, custo de comunicação e tempo de espera) são de minimização. Assim, o procedimento mostrado na Equação (5) foi empregado para computar os valores de aptidão relativo a cada métrica.

Quando se trabalha com um único objetivo, fica fácil saber se um indivíduo é melhor que o outro, simplesmente comparando a métrica de desempenho utilizada e verificando aquele com a melhor avaliação. Porém, no caso de AEMOs, é preciso utilizar algum tipo de estratégia mais elaborada. Pode ser possível aplicar uma estratégia de pesos para cada um dos objetivos combinando seus valores em um único objetivo (FRANÇA, 2018). Às vezes, porém, essa ponderação não é viável, fazendo-se necessário uma abordagem

que trabalhe apenas com o conceito de dominância de Pareto. Esse tema foi descrito no Capítulo 2.

Neste trabalho, combinou-se ambas abordagens. Para cada um dos AEMOs implementados, o retorno dado é o conjunto dos indivíduos não-dominados (Pareto-ótimo); porém dentro desse conjunto, foram realizadas as avaliações dos objetivos combinados, elegendo assim o melhor indivíduo e qual o AEMO que o produziu. Nesse sentido, foram utilizadas duas estratégias: a média simples e média harmônica, ambas aplicadas ao *fitness* dos indivíduos da população final. Embora a média simples seja usada em outros trabalhos (FRANÇA, 2018), optou-se por empregar, também, a média harmônica pois os objetivos considerados aqui apresentam magnitudes distintas. Nesse sentido, essa média tende fortemente para elementos menores de um conjunto, mitigando significativamente o impacto de *outliers*.

Para realizar o cálculo das médias, é importante fazer a normalização dos dados, uma vez que os valores das métricas utilizadas pelos AEMOs possuem uma grande diferença de intervalos se comparados uns aos outros. Para isso, é preciso converter os valores das métricas para um valor no intervalo [0,1]. Por esse motivo, é necessário obter os valores dos extremos (limitantes inferior e superior) para cada uma das métricas. Neste trabalho, optou-se por implementar o algoritmo genético mono-objetivo proposto por Omara e Arafa (2010).

Esse algoritmo, denominado SGA (do inglês, *standard genetic algorithm*) foi executado extensivamente para cada um dos cinco objetivos considerados neste trabalho[1]. Depois de encontrar os valores extremos, foi calculado o valor *norm* de um *fitness* f conforme mostrado na Equação (6), onde $V_i^{\max}$ e $V_i^{\min}$ são, respectivamente, o maior e o menor valor de *fitness* para os objetivos $i = \{1 \ldots,5\}$.

$$norm = \frac{f - V_i^{\min}}{V_i^{\max} - V_i^{\min}} \tag{6}$$

A fim de ilustrar a normalização, considere que o pior (ou seja, o maior) valor de makespan encontrado foi $M = 43$, enquanto o melhor foi $M = 16$. Os valores *fitness* são, portanto, $V_1^{\max} = 0{,}0625$ e $V_1^{\min} \approx 0{,}0233$, respectivamente. Suponha, então, que um novo indivíduo é gerado, com o makepan de 26 e *fitness* $f \approx 0{,}0385$. Nesse caso, o valor normalizado será:

$$norm = \frac{0{,}0385 - 0{,}0233}{0{,}0625 - 0{,}0233} \approx 0{,}3878.$$

Após a normalização de todos os valores de todos os objetivos, é possível computar as médias empregadas neste trabalho. A média simples (ou média aritmética) de um conjunto numérico x_1,$x_2, \ldots,x_n$ é a soma dos valores do conjunto dividida pelo número total de elementos no conjunto, conforme exposto pela Equação (7).

[1] É importante lembrar que o SGA originalmente considera apenas o *makespan*; neste trabalho, implementou-se todas as funções objetivo e manteve-se a estrutura do AG, substituindo apenas a função.

$$\bar{x}_H = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) \tag{7}$$

Já a média harmônica é obtida pela razão entre o número total de elementos do conjunto e a soma do inverso dos elementos do conjunto, conforme realçado pela Equação (8).

$$\bar{x} = n\left(\sum_{i=1}^{n} \frac{1}{x_i}\right)^{-1} \tag{8}$$

Capítulo 5

Resultados e Discussão

Neste capítulo são apresentadas as principais investigações realizadas no trabalho, envolvendo os ambientes multiobjetivos propostos para resolver o problema do escalonamento de tarefas com processamento paralelo. Os resultados obtidos através dos experimentos foram analisados de acordo com as métricas investigadas nos AEMOs implementados.

5.1 Ambiente de Experimentos

Neste trabalho, foi desenvolvido um programa na linguagem Java 8 para simular a execução do algoritmo multiobjetivo (NSGA-II) e dos algoritmos para múltiplos objetivos (AEMMT e AEMMD). As tarefas são representadas através de um DAG, bem como suas restrições de precedência, o custo computacional e o custo de comunicação. Os experimentos foram executados em um computador com processador Intel(R) Core(TM) i7-4790 CPU 3.60 GHz e 32.0 GB de memória principal. O código-fonte, as instâncias utilizadas e todos os resultados estão disponíveis no seguinte endereço: <https://github.com/johnataferreira/agscheduling>.

A população inicial de cada algoritmo é gerada de forma aleatória por meio de uma distribuição uniforme. Para isso, o programa utiliza o método estático `Random` da classe `Math` da própria linguagem Java. Esse método gera um valor aleatório no intervalo $[0, 1)$. Como dito, a ordem das tarefas é gerada de forma aleatória, porém respeitando a ordem de precedência existente e representada no DAG. Já o processador é sorteado de acordo com a quantidade de processadores totais utilizados no problema.

Para gerar resultado com valores médios, foi utilizado uma variação do método `Random` da classe `Math`, uma que recebe uma semente (*seed*) como parâmetro. Quando uma semente é usada por este método, os valores aleatórios gerados são sempre os mesmos, assim é possível reproduzir o mesmo resultado sempre que necessário. Para cada configuração do algoritmo, foram utilizadas 10 sementes distintas. As sementes também foram geradas de forma aleatória, no caso, dentro do intervalo entre o menor e o maior inteiro do

Java. Assim, obteve-se as seguintes sementes: −1995383100, −1989726638, −1949695351, −813009771, −609065737, 64083307, 866349162, 918118122, 983153476, 1793184929.

O primeiro critério para a avaliação dos algoritmos é a quantidade de objetivos otimizada por ele. Para os três AEMOs implementados, utilizou-se a seguinte quantidade de objetivos: 5, 4, 3 e 2. Busca-se, com isso, analisar os resultados com essas combinações. O segundo critério é a quantidade de tarefas representadas no DAG. Neste trabalho foram considerados três DAGs, com 50, 100 e 300 tarefas, respectivamente. Esses grafos fazem parte do repositório *Standard Task Graph* (STG), uma coleção *benchmark* para avaliação de algoritmos de escalonamento de tarefas[1]. Escolheu-se, aleatoriamente, os seguintes DAGs:

- `rand0000.stg` para 50 tarefas;
- `rand0055.stg` para 100 tarefas;
- `rand0105.stg` para 300 tarefas.

Os DAGs presentes no STG não possuem custo de comunicação. Como essa era uma das métricas a ser trabalhada, foi necessário implementar um algoritmo capaz de gerar esses valores e atribuir às arestas do DAG. A estratégia para geração do custo de comunicação foi ler todos os vértices e identificar a maior e a menor carga de trabalho dentre esses vértices. Com esses limites definidos, foi sorteado (aleatoriamente e com distribuição uniforme) um valor entre o mínimo e o máximo e atribuído a cada uma das arestas.

Por fim, há uma última variação na forma de execução dos AEMOs: a quantidade de processadores que executam as tarefas mapeadas. O algoritmo foi executado para cada DAG respeitando a seguinte quantidade de processadores: 2, 4, 8 e 16.

5.2 Método de Avaliação

Os experimentos realizados foram executados considerando as variações citadas na seção anterior, ou seja: **quantidade de objetivos** × **quantidade de tarefas no DAG** × **quantidade de processadores**. O objetivo dessa análise é encontrar o melhor AEMO para cada uma dessas variações.

Inicialmente foi feita uma análise em torno dos objetivos de forma individual. Busca-se saber como se deu a distribuição particular desses objetivos dentro dos cenários possíveis. Para realizar essa análise, foi necessário executar os algoritmos extensivamente até encontrar um valor máximo e mínimo por objetivo. Através desse resultado, montou-se uma

[1] Disponível em: <https://www.kasahara.cs.waseda.ac.jp/schedule/index.html>. Acesso em 6 fev. 2023.

tabela com dez intervalos de valores dentro do mínimo e máximo atingidos. Essa tabela será base para análise dos objetivos obtidos e será apresentada na Seção 5.2.

Por fim, após a execução de todos os AEMOs do trabalho para cada uma dessas combinações, foi feita uma análise para observar qual algoritmo produziu o melhor indivíduo, ou seja, a melhor solução para o problema do escalonamento de tarefas. Foram utilizadas as duas medidas descritas na seção 4.4, ou seja, a média simples e a média harmônica.

Neste capítulo serão apresentados todos os resultados (tabelas e gráficos) produzidos para o DAG de 50 tarefas. Por simplicidade, apenas a análise dos melhores indivíduos é descrita para os DAGs de 100 e 300 tarefas. Os demais resultados dessas instâncias estão no apêndice.

Como dito anteriormente, a divisão dos intervalos foi realizada em dez partes iguais. O valor de *Makespan* será utilizado pra exemplificar essa divisão. O melhor resultado para essa métrica foi 133,0 e o pior foi 287,0. A diferença entre esses valores é 154,0. Como foram utilizados 10 intervalos, fez-se necessário dividir esse valor por 10 resultando em 15,4. Assim, cada intervalo trabalhado possui uma parcela própria com esse valor.

Como esse objetivo é uma métrica de minimização, quanto menor seu valor, melhor o indivíduo. Desse modo, o menor valor possível fica no intervalo 90–100 que representa a parcela dos melhores resultados. Nesse exemplo, como o melhor resultado foi 133,0 e o valor que compõe o intervalo é 15,4, o respectivo intervalo possui os limites 133,0 e 148,4 $(133{,}0 + 15{,}4)$. O mesmo cálculo é realizado para os demais intervalos e também para os demais objetivos. Vale ressaltar que o intervalo de valores obtidos não varia em relação a quantidade de objetivos trabalhada, uma vez que o intuito dessa análise é encontrar o melhor e o pior valor para a métrica.

A Tabela 1 traz os valores de intervalo para todos os objetivos, para o DAG de 50 tarefas. Seguindo a notação estabelecida no Capítulo 3, m representa o número de processadores. Os valores em negrito representam o melhor e o pior valor da métrica, considerando uma determinada quantidade de processadores definida.

Observa-se, na tabela, que à medida em que o número de processadores aumenta o melhor valor de *Makespan* (M) diminui, exceto no cenário com 16 processadores que gerou um valor um pouco maior que a execução com 8 processadores. Isso mostra que o aumento de processadores, nesse caso, trouxe de fato uma redução no tempo de processamento, reduzindo a carga de cada processador.

Por outro lado, algo diferente ocorre com o balanceamento de carga (*Load balancing*, L), que aumentou junto com o aumento no número de processadores. Isso indica que a redução no tempo de processamento gerou um desequilíbrio na distribuição da carga entre os processadores.

Já o resultado obtido para o *Flowtime* (F) foi mais parecido com o *Makespan*. Além disso, outro comportamento percebido através de análise da Tabela 1, é que a diferença entre o melhor e o pior valor de F foi diminuindo com o aumento do número de proces-

Tabela 1 – Intervalo de valores obtidos para cada objetivo no DAG de 50 tarefas.

Obj.	**Inter.**	$m = 2$		$m = 4$		$m = 8$		$m = 16$	
		Inf.	**Sup.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**
M	90–100	**133,0**	148,4	**85,0**	100,7	**74,0**	88,7	**76,0**	85,5
	80–90	148,4	163,8	100,7	116,4	88,7	103,4	85,5	95,0
	70–80	163,8	179,2	116,4	132,1	103,4	118,1	95,0	104,5
	60–70	179,2	194,6	132,1	147,8	118,1	132,8	104,5	114,0
	50–60	194,6	210,0	147,8	163,5	132,8	147,5	114,0	123,5
	40–50	210,0	225,4	163,5	179,2	147,5	162,2	123,5	133,0
	30–40	225,4	240,8	179,2	194,9	162,2	176,9	133,0	142,5
	20–30	240,8	256,2	194,9	210,6	176,9	191,6	142,5	152,0
	10–20	256,2	271,6	210,6	226,3	191,6	206,3	152,0	161,5
	0–10	271,6	**287,0**	226,3	**242,0**	206,3	**221,0**	161,5	**171,0**
L	90–100	**1,00000**	1,03233	**1,00146**	1,04827	**1,04092**	1,10721	**1,18876**	1,28219
	80–90	1,03233	1,06465	1,04827	1,09508	1,10721	1,17350	1,28219	1,37563
	70–80	1,06465	1,09698	1,09508	1,14189	1,17350	1,23979	1,37563	1,46907
	60–70	1,09698	1,12931	1,14189	1,18871	1,23979	1,30608	1,46907	1,56251
	50–60	1,12931	1,16163	1,18871	1,23552	1,30608	1,37238	1,56251	1,65595
	40–50	1,16163	1,19396	1,23552	1,28233	1,37238	1,43867	1,65595	1,74939
	30–40	1,19396	1,22628	1,28233	1,32915	1,43867	1,50496	1,74939	1,84283
	20–30	1,22628	1,25861	1,32915	1,37596	1,50496	1,57125	1,84283	1,93626
	10–20	1,25861	1,29094	1,37596	1,42277	1,57125	1,63754	1,93626	2,02970
	0–10	1,29094	**1,32326**	1,42277	**1,46959**	1,63754	**1,70383**	2,02970	**2,12314**
F	90–100	**3263,0**	3681,1	**2071,0**	2475,3	**1679,0**	2066,8	**1672,0**	1918,0
	80–90	3681,1	4099,2	2475,3	2879,6	2066,8	2454,6	1918,0	2164,0
	70–80	4099,2	4517,3	2879,6	3283,9	2454,6	2842,4	2164,0	2410,0
	60–70	4517,3	4935,4	3283,9	3688,2	2842,4	3230,2	2410,0	2656,0
	50–60	4935,4	5353,5	3688,2	4092,5	3230,2	3618,0	2656,0	2902,0
	40–50	5353,5	5771,6	4092,5	4496,8	3618,0	4005,8	2902,0	3148,0
	30–40	5771,6	6189,7	4496,8	4901,1	4005,8	4393,6	3148,0	3394,0
	20–30	6189,7	6607,8	4901,1	5305,4	4393,6	4781,4	3394,0	3640,0
	10–20	6607,8	7025,9	5305,4	5709,7	4781,4	5169,2	3640,0	3886,0
	0–10	7025,9	**7444,0**	5709,7	**6114,0**	5169,2	**5557,0**	3886,0	**4132,0**
C	90–100	**127,0**	191,2	**276,0**	345,6	**396,0**	470,0	**467,0**	542,5
	80–90	191,2	255,4	345,6	415,2	470,0	544,0	542,5	618,0
	70–80	255,4	319,6	415,2	484,8	544,0	618,0	618,0	693,5
	60–70	319,6	383,8	484,8	554,4	618,0	692,0	693,5	769,0
	50–60	383,8	448,0	554,4	624,0	692,0	766,0	769,0	844,5
	40–50	448,0	512,2	624,0	693,6	766,0	840,0	844,5	920,0
	30–40	512,2	576,4	693,6	763,2	840,0	914,0	920,0	995,5
	20–30	576,4	640,6	763,2	832,8	914,0	988,0	995,5	1071,0
	10–20	640,6	704,8	832,8	902,4	988,0	1062,0	1071,0	1146,5
	0–10	704,8	**769,0**	902,4	**972,0**	1062,0	**1136,0**	1146,5	**1222,0**
W	90–100	**366,0**	521,4	**262,0**	383,6	**219,0**	314,1	**201,0**	269,7
	80–90	521,4	676,8	383,6	505,2	314,1	409,2	269,7	338,4
	70–80	676,8	832,2	505,2	626,8	409,2	504,3	338,4	407,1
	60–70	832,2	987,6	626,8	748,4	504,3	599,4	407,1	475,8
	50–60	987,6	1143,0	748,4	870,0	599,4	694,5	475,8	544,5
	40–50	1143,0	1298,4	870,0	991,6	694,5	789,6	544,5	613,2
	30–40	1298,4	1453,8	991,6	1113,2	789,6	884,7	613,2	681,9
	20–30	1453,8	1609,2	1113,2	1234,8	884,7	979,8	681,9	750,6
	10–20	1609,2	1764,6	1234,8	1356,4	979,8	1074,9	750,6	819,3
	0–10	1764,6	**1920,0**	1356,4	**1478,0**	1074,9	**1170,0**	819,3	**888,0**

sadores.

O custo de comunicação (*Communication cost*, C), por sua vez, trouxe bastante similaridade com os resultados de L: quanto maior o número de processadores, maior (portanto, pior) o valor de C. Isso faz sentido, uma vez que o custo de comunicação incorre apenas se a tarefa for executada em um processador diferente da sua predecessora ou sucessora.

Por fim, os resultados de tempo de espera (*Waiting time*, W) contrariam os valores

de C. Ou seja, quanto mais processadores disponíveis para executar as tarefas do DAG, menor será o tempo em que eles ficarão aguardando para serem processadas. Ao que tudo indica, o tempo de comunicação acaba sendo compensado pelo tempo de execução dentro de cada processador, reduzindo a espera.

5.3 Experimentos e Avaliação dos Resultados

Foram realizados diversos experimentos visando uma análise comparativa entre o AG multiobjetivo NSGA-II e os AGs *many-objectives* AEMMT (este com duas variações: um que utiliza a média simples e outra a média harmônica) e AEMMD. O número de objetivos varia entre 2 e 5, os DAGs que mapeiam o problema do escalonamento mapeiam 50, 100 e 300 tarefas que são executadas em 2, 4, 8 e 16 processadores. Ao todo foram estabelecidos 48 cenários experimentais para cada algoritmo.

As execuções realizadas adotaram os parâmetros de configuração listados na Tabela 2. Cada AEMO tem suas próprias características de execução, portanto a configuração utilizada faz jus a essa especificidade.

Tabela 2 – Parâmetros utilizados na execução dos AEMOs.

Parâmetro	**AEMMT**	**AEMMD**	**NSGA-II**
Tamanho da população	480	480	200
Número de gerações	15000	15000	500
Avaliações de *Fitness*	15480	15480	100000
Método de seleção	Torneio Simples ($tour = 2$)	Torneio Simples ($tour = 2$)	Torneio Simples ($tour = 2$)
Taxa de crossover	-	-	100%
Taxa de mutação	a cada 500 gerações	a cada 500 gerações	2%
Tamanho das tabelas	30	-	-
Tamanho da tabela de dominância	100	-	-
Método para reinserção	-	-	Reinserção Ordenada

Com relação ao tamanho da população, o AEMMT possui um cálculo com base no tamanho de suas tabelas de indivíduos. Considera-se, como tamanho da população, o produto do tamanho da tabela pela quantidade de tabelas. No caso, são 16 tabelas de tamanho 30, cada, totalizando 480 indivíduos. O AEMMD não estabelece um limite de tamanho em suas tabelas; assim, obtou-se por utilizar, também, 480 indivíduos.

Como os AGs *many-objectives* produzem apenas um descendente por geração, é necessário um número alto de gerações para alcançar o a mesma quantidade de soluções avaliadas que outros AGs, por isso a grande diferença deles para com o NSGA-II. O inverso ocorre com as avaliações de *fitness*. O NSGA-II possui um valor muito maior, uma vez que a cada geração produz 200 novos indivíduos através do operador de *crossover*. Isso ocorre por conta do tamanho da população e a taxa de *crossover*. Se adotasse uma taxa menor, a diferença também seria reduzida.

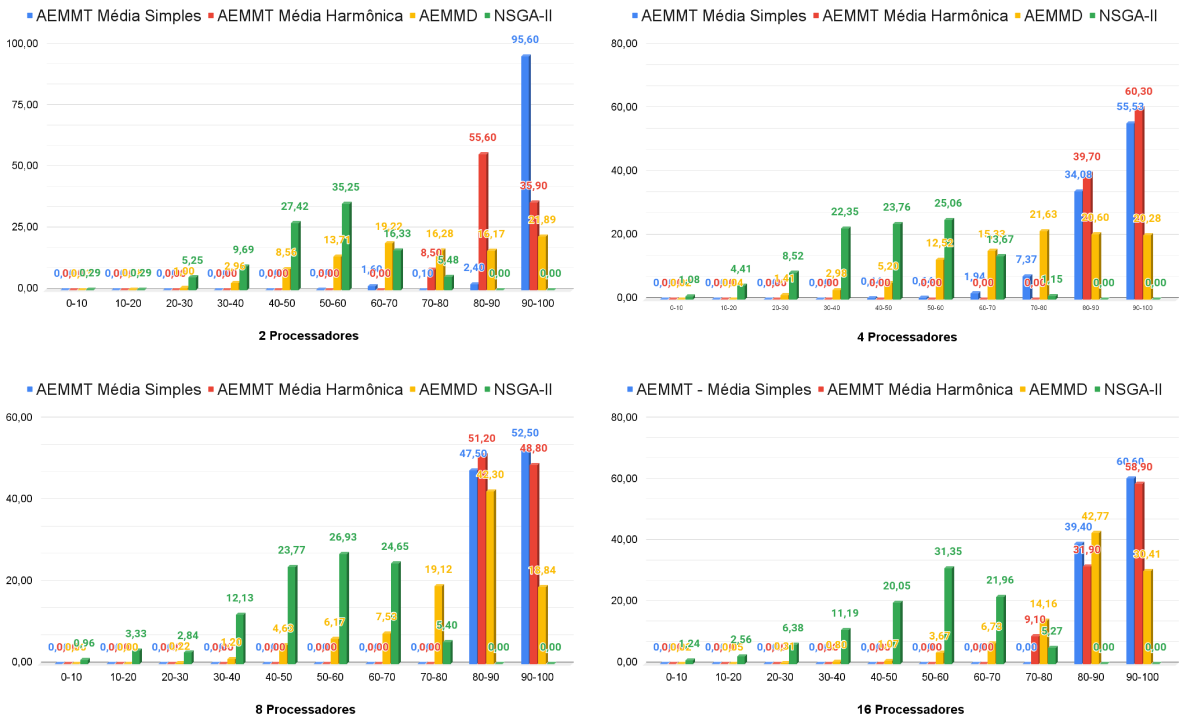
O método de seleção utilizado por esses algoritmos é o mesmo (i.e., torneio simples), mas com uma diferença: o torneio realizado no AEMMT e no AEMMD ocorre nas tabelas de indivíduos enquanto que para o NSGA-II ocorre na própria população. A taxa de *crossover* não incorre para o AEMMT e para o AEMMD, pois ocorre apenas um *crossover* por geração. Por fim, o método de reinserção padrão do NSGA-II é a reinserção ordenada, ou seja, os melhores indivíduos são mantidos para a próxima geração. Já o AEMMT e o AEMMD não possuem especificamente um método de reinserção: todos os indivíduos presentes nas tabelas, ao fim de uma geração, são mantidos para a próxima.

5.3.1 Percentuais por AEMO nos intervalos dos objetivos

Na Seção 5.2 foram apresentados os intervalos obtidos para cada um dos objetivos investigados neste trabalho. Agora, são apresentados os percentuais atingidos por AEMO para cada um desses objetivos, nos respectivos intervalos mapeados. A apresentação é feita por meio de gráficos, o que facilitará a comparação dos AEMOs, permitindo identificar os melhores resultados. Os intervalos cujos percentuais obtidos pelos algoritmos alcançaram valores menores ou iguais a 0.1 foram omitidos dos gráficos.

A Figura 22 apresenta uma comparação dos resultados para o *Makespan* em um DAG de 50 tarefas em uma execução com 5 objetivos. Os melhores resultados ficaram por conta do AEMMT com média simples, seguido pelo AEMMT com média harmônica, o AEMMD

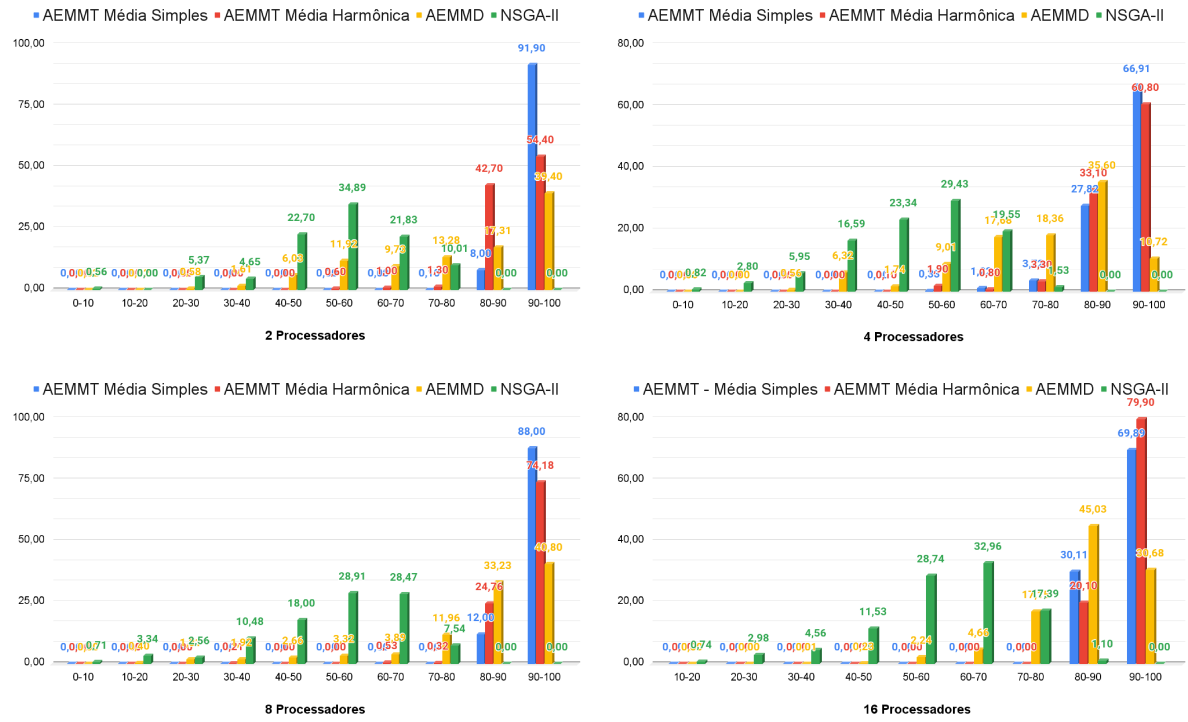
Figura 22 – Percentuais obtidos por AEMO nos intervalos de valores do *Makespan* em um DAG de 50 tarefas com execução de 5 objetivos.



e, por último, o NSGA-II. Um dado interessante é que, com 2 processadores, o AEMMT ficou muito acima dos demais, mas no restante das execuções ele ficou disputando posições com o AEMMT com média harmônica.

Na execução com 4 objetivos, os resultados foram similares ao caso anterior, conforme observado na Figura 23. O AEMMT com média simples é superior em cenários com 2 e 8 processadores e o AEMMT com média harmônica nos demais. Ambos seguidos pelo AEMMD e depois pelo NSGA-II. Houve apenas uma inversão entre os dois AEMMTs nas execuções com 4 e 16 processadores.

Figura 23 – Percentuais obtidos por AEMO nos intervalos de valores do *Makespan* em um DAG de 50 tarefas com execução de 4 objetivos



Na terceira execução, com 3 objetivos, o AEMMT com média harmônica se sobressaiu em todas as execuções seguido pelo AEMMT com média simples. O AEMMD passou a gerar menos indivíduos no intervalo de 90 a 100. A Figura 24 ilustra essa piora. O NSGA-II, por sua vez continua mantendo uma similaridade entre as gerações.

A última execução trouxe o pior resultado. No cenário com 2 processadores, todas as execuções trouxeram poucos indivíduos no melhor intervalo. Pode-se observar, na Figura 25, que o AEMMT com média harmônica se sobressaiu nos cenários com 4, 8 e 16 processadores, já o AEMMT com média simples foi melhor na execução com 2, com uma pequena diferença. O NSGA-II continua sem produzir indivíduos nas melhores posições. O AEMMD também não produziu bons indivíduos nessa execução, sendo que nenhum foi gerado no melhor intervalo e poucos aparecem no intervalo de 80 a 90.

Figura 24 – Percentuais obtidos por AEMO nos intervalos de valores do *Makespan* em um DAG de 50 tarefas com execução de 3 objetivos

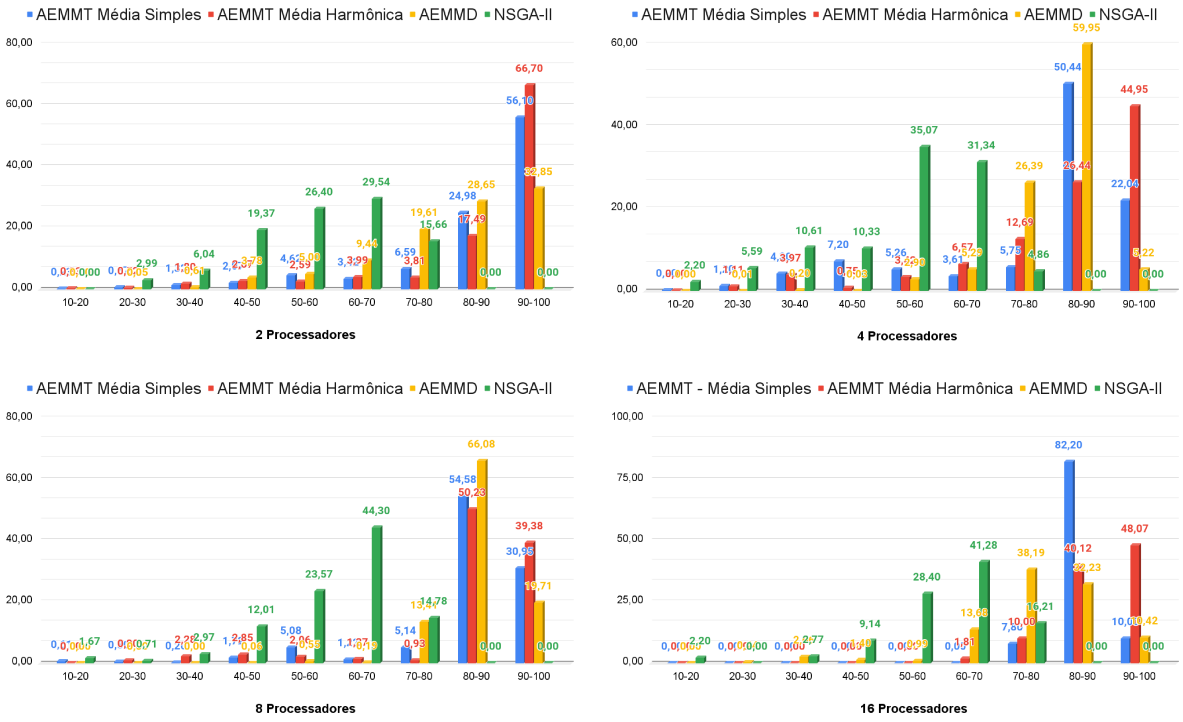
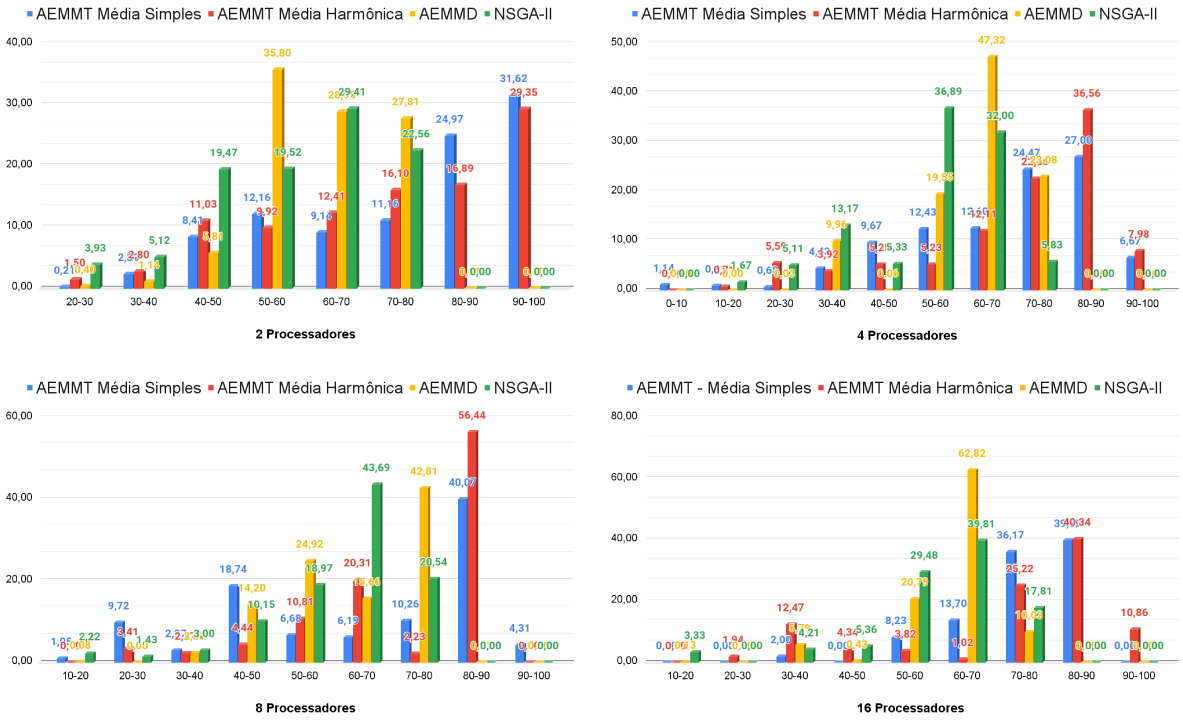
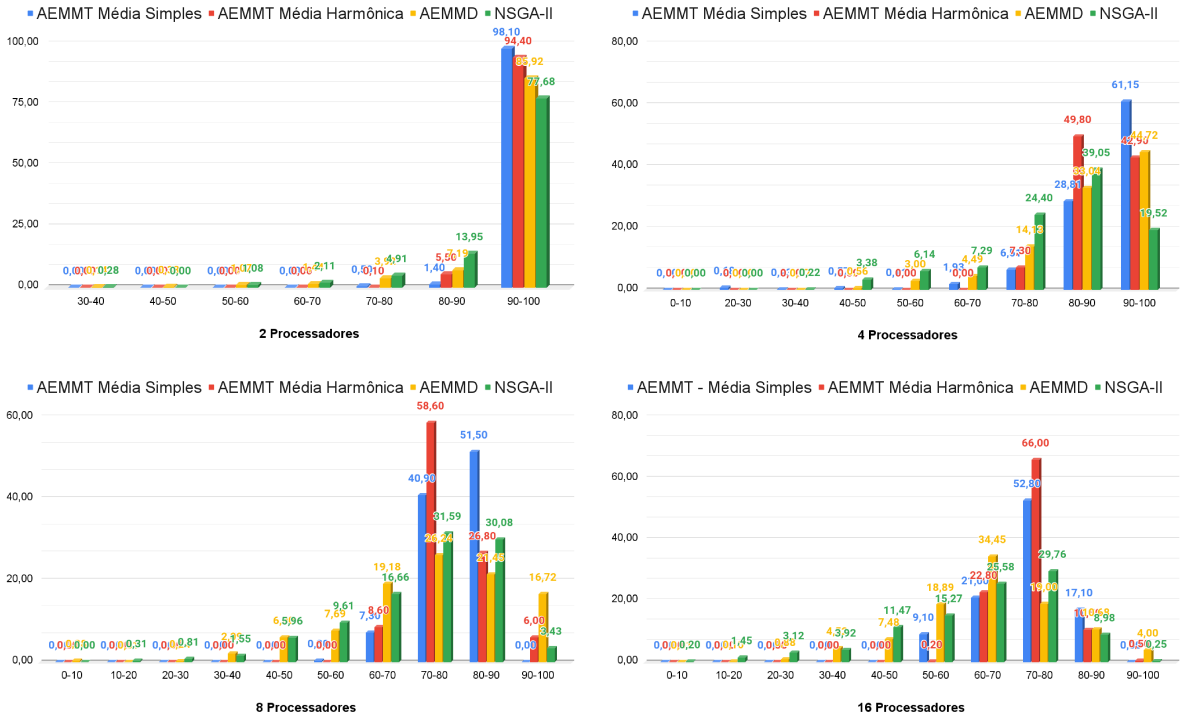


Figura 25 – Percentuais obtidos por AEMO nos intervalos de valores do *Makespan* em um DAG de 50 tarefas com execução de 2 objetivos



Analisando o *Load balancing* em um DAG de 50 tarefas, com execução de 5 objetivos (Figura 26), pode-se perceber que a grande maioria dos indivíduos ficou no intervalo 90–100 para 2 e 4 processadores com o AEMMT de média simples se sobressaindo. Para as execuções com 8 e 16 processadores o AEMMD obteve os melhores resultados. O AEMMT com media harmônica e o NSGA-II geraram muitos bons indivíduos apenas nas execuções com 2 e 4 processadores.

Figura 26 – Percentuais obtidos por AEMO nos intervalos de valores do *Load balancing* em um DAG de 50 tarefas com execução de 5 objetivos



A Figura 27 representa a execução com 4 objetivos e é possível visualizar que as execuções com 2 e 4 processadores foram bastante similares se comparadas a execução anterior com uma ligeira inversão de valores entre o AEMMT com média harmônica e AEMMD na execução de 4 processadores. O NSGA-II foi o AG que gerou os resultados mais similares em relação à execução com 5 objetivos.

Da mesma forma, as execuções com 3 e 2 objetivos também tiveram seus resultados similares. A Figura 28 assim como a Figura 29, mostram um resultado de 100% de indivíduos no intervalo 90–100 para 2 processadores tanto para o AEMMT com média simples quanto para o AEMMD. O AEMMT se manteve assim também para a execução com 4 processadores, porém o AEMMD caiu bastante na execução com 3 objetivos. O NSGA-II também foi bem nas execuções com 2 e 4 processadores, seus melhores resultados até o momento.

Figura 27 – Percentuais obtidos por AEMO nos intervalos de valores do *Load balancing* em um DAG de 50 tarefas com execução de 4 objetivos

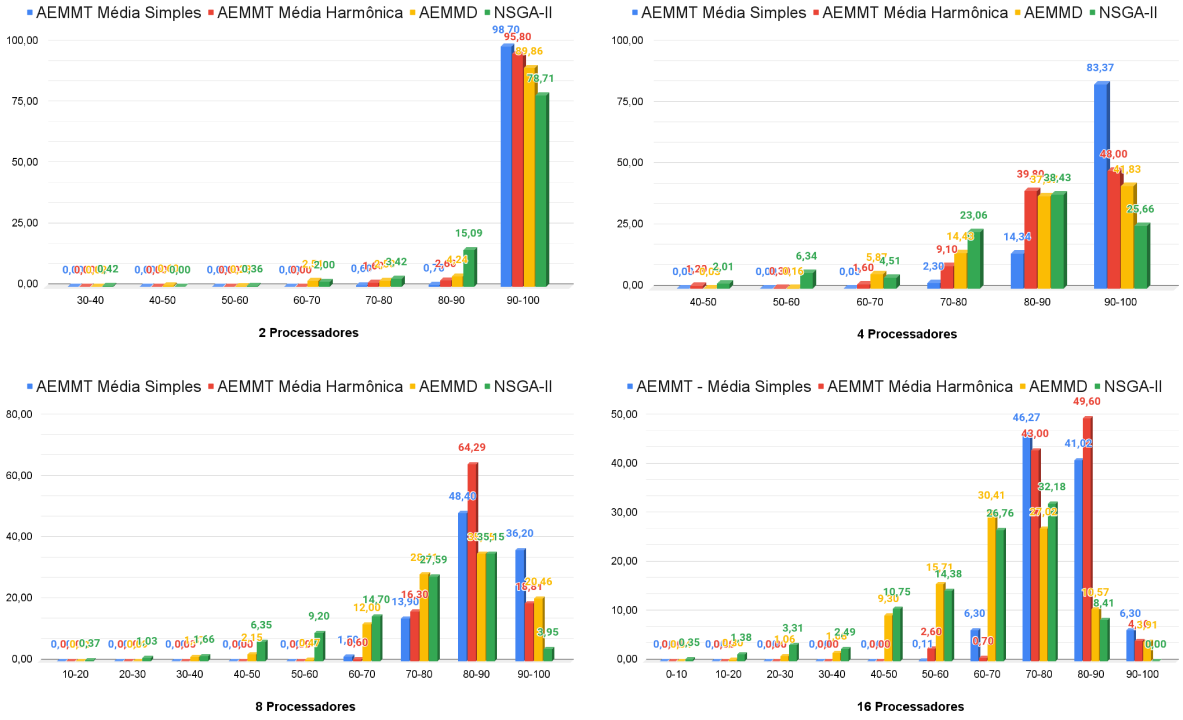


Figura 28 – Percentuais obtidos por AEMO nos intervalos de valores do *Load balancing* em um DAG de 50 tarefas com execução de 3 objetivos

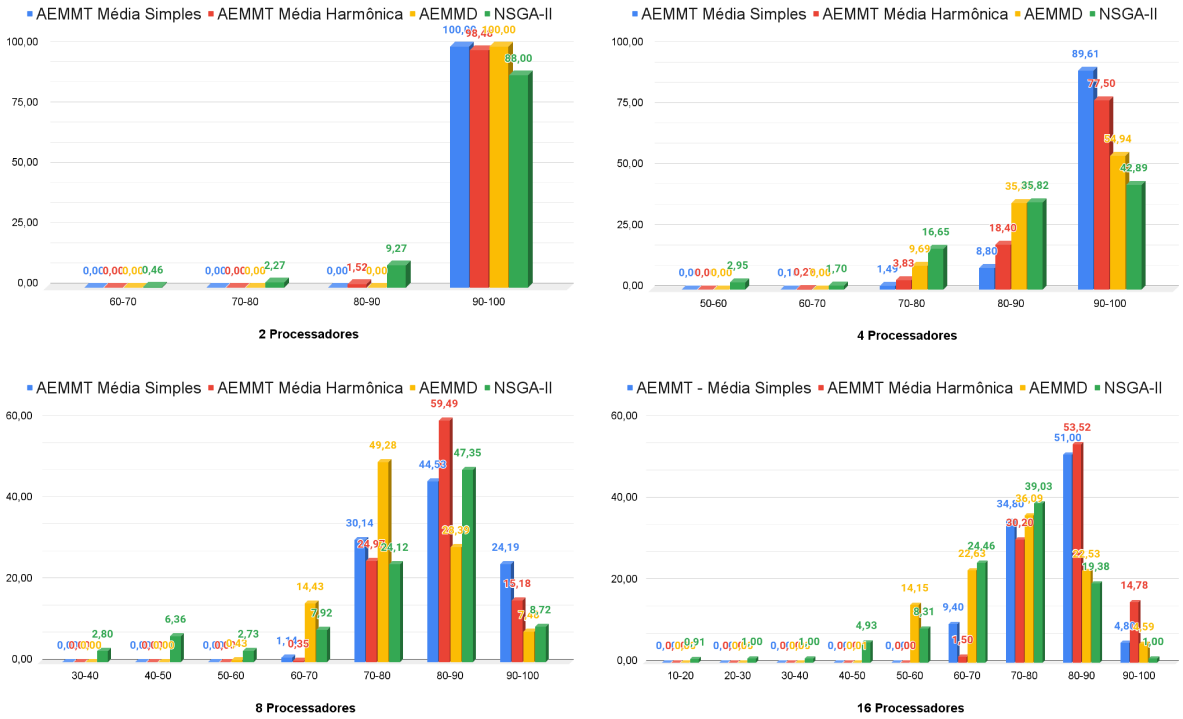
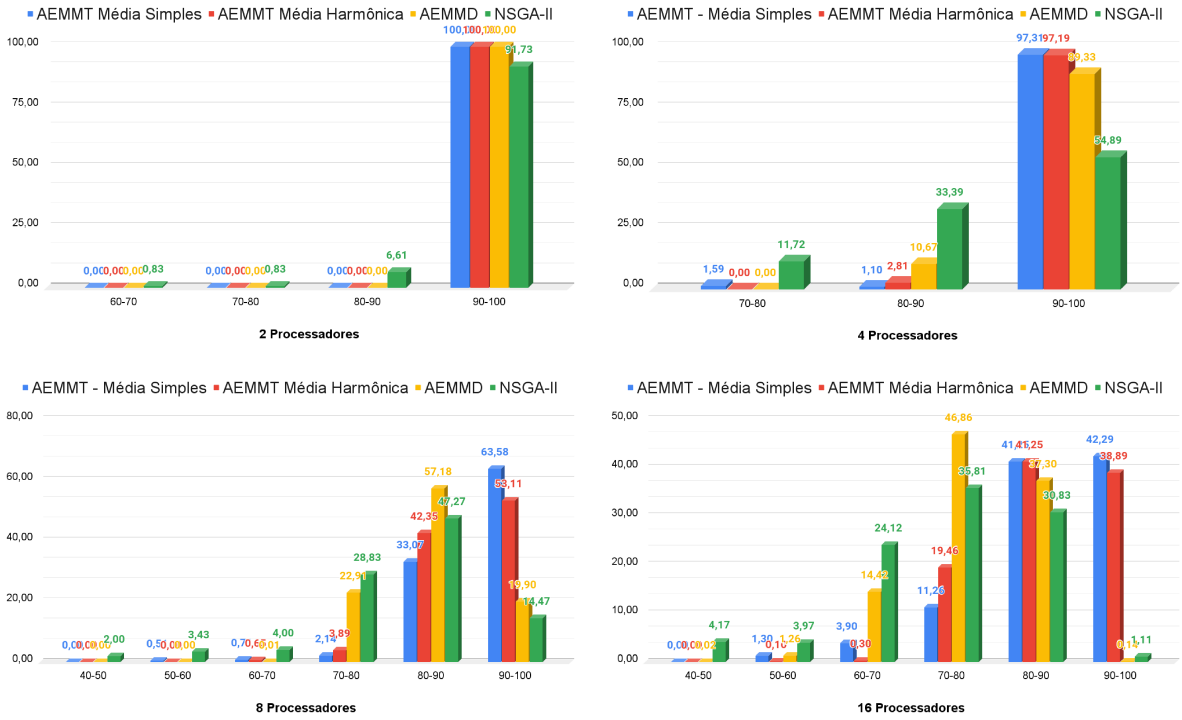


Figura 29 – Percentuais obtidos por AEMO nos intervalos de valores do *Load balancing* em um DAG de 50 tarefas com execução de 2 objetivos



As próximas métricas não foram utilizadas em todas as execuções. A análise do *Flowtime* por exemplo foi realizada apenas nas execuções com 5, 4 e 3 objetivos. Para 2 objetivos, como já analisado anteriormente, foram utilizados apenas o *Makespan* e o *Load balancing*. À medida em que os objetivos aumentam, a quantidade de execuções diminui. O *Communication cost* por sua vez foi utilizado nas execuções com 5 e 4 objetivos e o *Waiting time*, como última métrica, será utilizado apenas na execução com 5 objetivos.

A Figura 30 exibe um bom comportamento do AEMMT com média simples: em todas as execuções, o maior percentual obtido foi no intervalo 90–100. A variação dos resultados foi pequena de uma execução para outra, caindo um pouco apenas com 4 processadores, onde o AEMMT com média harmônica dominou os resultados. Um ponto a ser observado foi o resultado do AEMMD. Ele não teve bons resultados com 2, 4 e 8 processadores, mas ficou próximo dos melhores algoritmos na execução com 16. O NSGA-II por sua vez não trouxe variações, manteve mais ou menos os mesmos resultados.

A execução com 4 objetivos trouxe um resultado similar quando a análise é feita em cima do AEMMT com média simples. O AG que teve uma variação em relação a execução anterior foi o AEMMT com média harmônica: seus resultados aumentaram no melhor intervalo para 2 processadores e diminuíram para 16. É possível visualizar na Figura 31 que o AEMMD e o NSGA-II também mantiveram resultados similares com a execução anterior.

Figura 30 – Percentuais obtidos por AEMO nos intervalos de valores do *Flowtime* em um DAG de 50 tarefas com execução de 5 objetivos

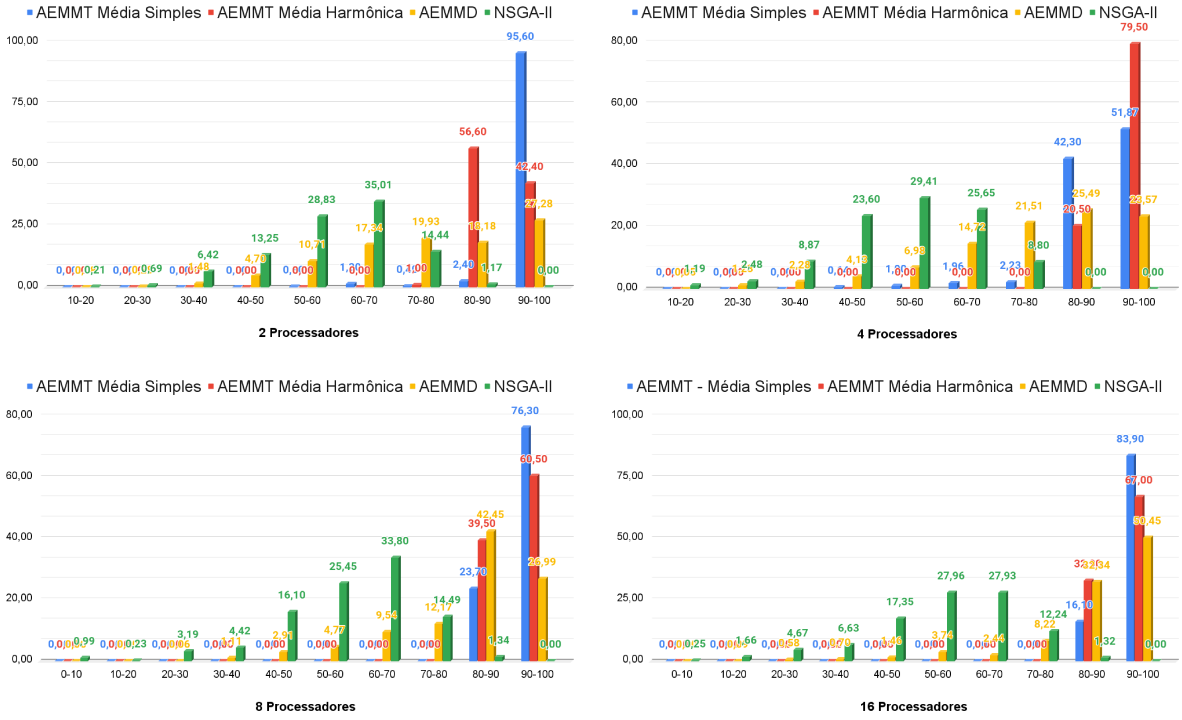
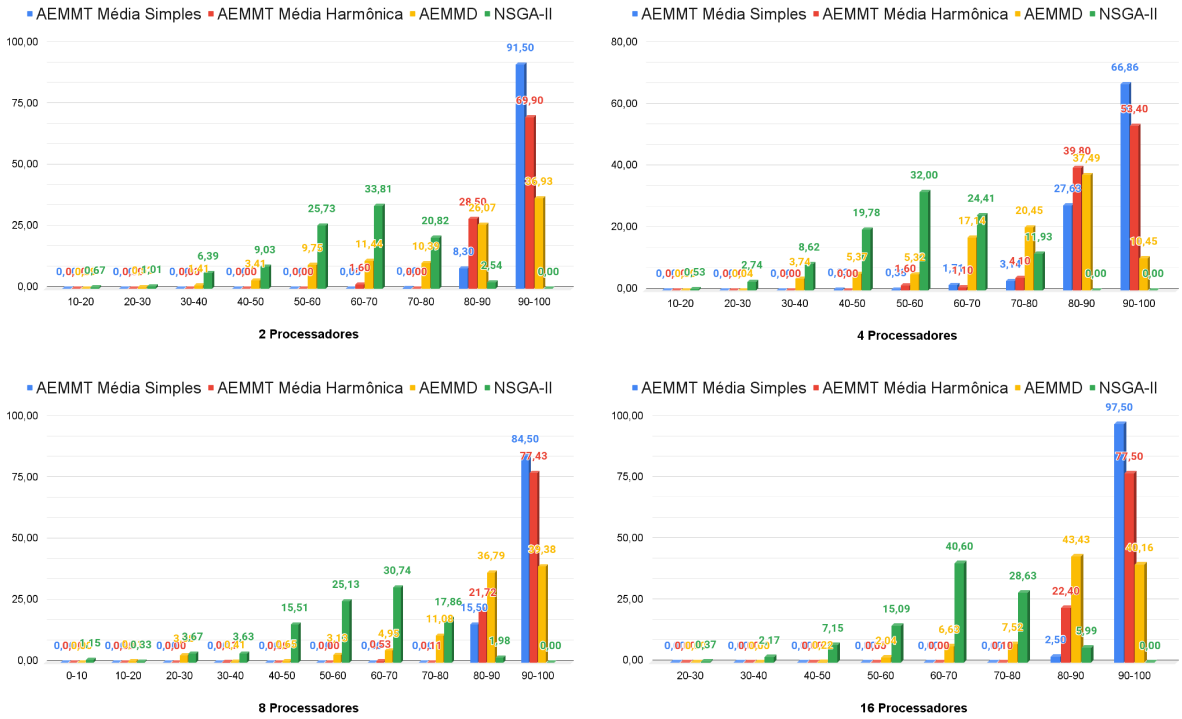
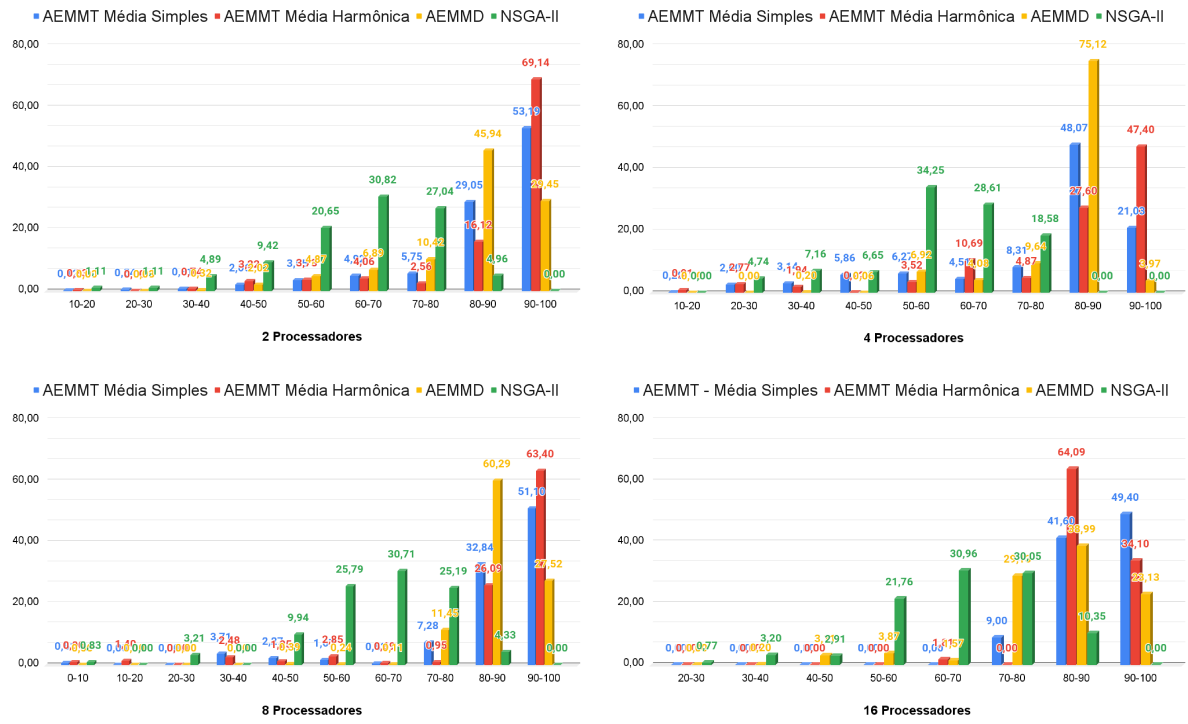


Figura 31 – Percentuais obtidos por AEMO nos intervalos de valores do *Flowtime* em um DAG de 50 tarefas com execução de 4 objetivos



Pela Figura 32, percebe-se que o resultado da execução do DAG de 50 tarefas com 3 objetivos trouxe o AEMMT com média harmônica como melhor algoritmo em quase todas as execuções, exceto na execução com 16 processadores, onde o AEMMT com média simples se sobressaiu. O AEMMD ficou bem próximo do resultado da execução com 4 objetivos e o NSGA-II também não mostrou melhora, não produzindo indivíduos no melhor intervalo.

Figura 32 – Percentuais obtidos por AEMO nos intervalos de valores do *Flowtime* em um DAG de 50 tarefas com execução de 3 objetivos



Os resultados com o *Communication cost* ficaram concentrados em poucos intervalos. Pela Figura 33 (execução com 5 objetivos), é possível ver que o AEMMD gerou melhores resultados em quase todas as execuções, com exceção da execução com 8 processadores, onde foi vencido pelo AEMMT com média simples. Além disso, na execução com 2 processadores, o AEMMD praticamente empatou com o AEMMT com média harmônica. O NSGA-II, novamente, não produziu indivíduos no melhor intervalo. A execução com 4 objetivos também trouxe resultados concentrados em poucos intervalos. Dessa vez o AEMMD foi melhor apenas com 4 processadores. Com 2 e 8 o AEMMT com média harmônica obteve os melhores resultados e com 16 o AEMMT com média simples venceu. Dessa vez o resultado ficou distribuído em vários algoritmos como pode ser visto na Figura 34. O NSGA-II conseguiu trazer indivíduos apenas no intervalo de 80 a 90.

Figura 33 – Percentuais obtidos por AEMO nos intervalos de valores do *Communication cost* em um DAG de 50 tarefas com execução de 5 objetivos

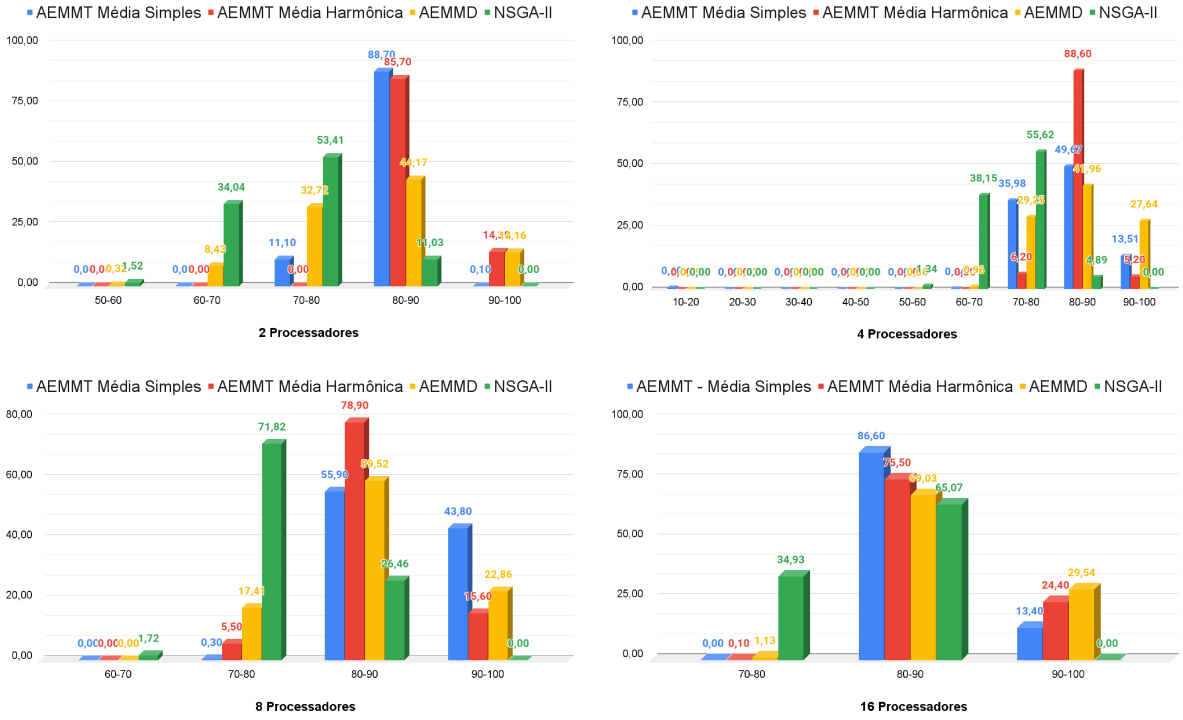
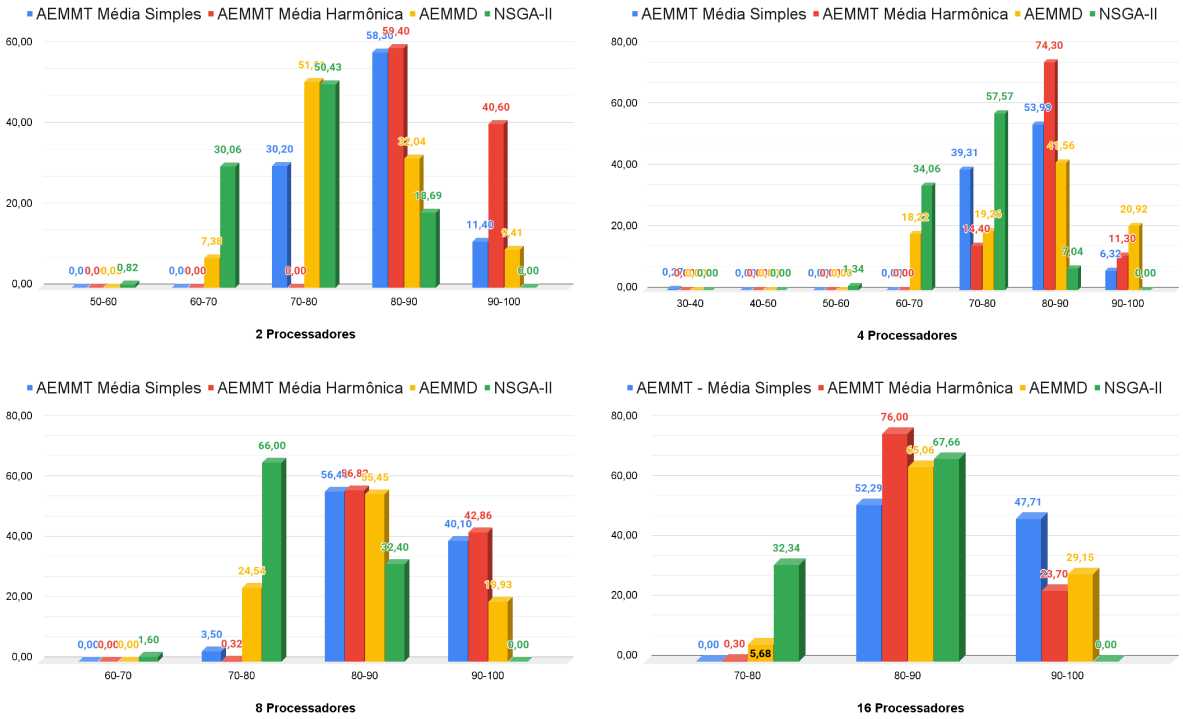
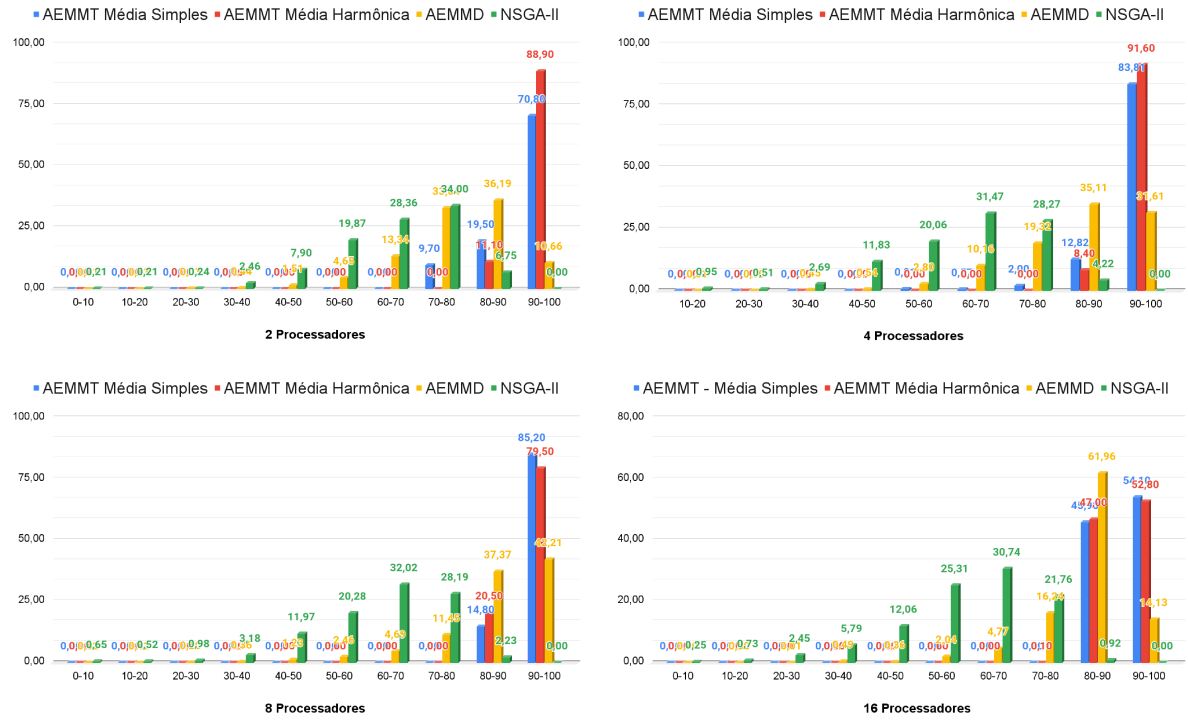


Figura 34 – Percentuais obtidos por AEMO nos intervalos de valores do *Communication cost* em um DAG de 50 tarefas com execução de 4 objetivos



A métrica do *Waiting time* foi utilizada apenas na execução do DAG com 5 objetivos, logo a comparação dos resultados foi feita pela quantidade de processadores utilizados. No geral, a disputa dos melhores algoritmos para essa métrica ficou entre o AEMMT com média simples e o AEMMT com média harmônica, este se sobressaindo com 8 e 16 processadores e aquele com 2 e 4. O AEMMD trouxe resultados no melhor intervalo, mas ainda assim, ficou bastante longe dos vencedores. Já o NSGA-II manteve um padrão em todas as execuções, gerando valores bem próximos uns dos outros para cada intervalo.

Figura 35 – Percentuais obtidos por AEMO nos intervalos de valores do *Waiting time* em um DAG de 50 tarefas com execução de 5 objetivos



5.3.2 Melhores valores de objetivos encontrados pelos AEMOs

Nesta seção, são apresentados os melhores valores encontrados por objetivo para cada um dos AEMOs implementados. Apresenta-se, também, a média dos melhores valores por objetivo e o desvio padrão dessa média. Esse resultado tem como foco a execução dos algoritmos considerando 5 objetivos, pois essa configuração é o foco deste trabalho.

Por convenção, a sigla AEMMT1 representará o AEMMT com média simples; já AEMMT2 é o AEMMT com média harmônica. A Tabela 3 apresenta os resultados para cada um dos algoritmos trabalhados. Essa tabela exibe os melhores valores obtidos em negrito. A execução das configurações com 2, 4, 8 e 16 processadores foram agrupadas para facilitar a análise.

Tabela 3 – Melhores objetivos encontrados por AEMO em um DAG de 50 tarefas na execução com 5 objetivos.

m	Obj.	AEMMT1		AEMMT2		AEMMD		NSGA-II	
		melhor	média	melhor	média	melhor	média	melhor	média
2	M	**133,000**	134,600±1,955	**133,000**	142,900±5,587	**133,000**	135,200±2,251	167,000	173,500±5,603
	L	**1,00000**	1,00036±0,00114	**1,00000**	1,00136±0,00325	**1,00000**	1,00000±0,00000	**1,00000**	1,00022±0,00069
	F	**3258,000**	3350,900±70,693	3300,000	3537,700±132,766	3299,000	3371,800±77,740	3917,000	4122,900±139,155
	C	191,000	212,000±18,791	146,000	190,100±20,388	**127,000**	160,000±24,545	208,000	226,200±9,647
	W	382,000	450,000±73,224	**374,000**	436,100±30,377	385,000	439,300±31,833	561,000	621,500±32,153
4	M	**82,000**	95,400±8,746	88,000	95,700±4,111	85,000	94,900±6,226	127,000	137,600±6,736
	L	1,00386	1,01254±0,00808	1,01064	1,03198±0,01526	**1,00146**	1,00698±0,00443	1,01155	1,02356±0,00692
	F	**2022,000**	2318,500±191,513	2107,000	2290,800±123,105	2136,000	2267,500±104,138	3000,000	3128,200±117,795
	C	316,000	366,600±39,822	327,000	365,900±24,062	**305,000**	332,000±16,425	353,000	404,300±19,883
	W	**236,000**	298,700±33,247	267,000	318,800±31,365	262,000	305,500±27,273	437,000	496,300±32,121
8	M	80,000	87,300	4,398±80,000	89,000±5,981	**75,000**	87,000±6,864	108,000	114,000±4,372
	L	1,11515	1,14781±0,02160	1,07533	1,14477±0,03766	**1,04092**	1,06321±0,01075	1,07648	1,10041±0,01619
	F	1803,000	1926,800±95,211	1837,000	1974,100±107,410	**1718,000**	1963,300±140,580	2330,000	2478,100±74,536
	C	433,000	463,400±21,930	436,000	472,700±21,562	**418,000**	448,300±16,647	479,000	505,700±13,073
	W	230,000	264,700±20,955	**225,000**	258,400±24,254	227,000	266,300±24,143	389,000	406,600±11,325
16	M	76,000	83,300±4,473	**71,000**	82,400±6,096	78,000	82,700±3,057	96,000	101,100±3,814
	L	1,33333	1,39128±0,05424	1,21035	1,35920±0,06677	**1,18876**	1,22430±0,01854	1,26796	1,33138±0,03055
	F	**1657,000**	1798,200±74,093	1670,000	1806,300±90,222	1767,000	1799,300±24,734	2071,000	2205,000±89,800
	C	512,000	544,900±18,460	506,000	534,500±20,018	**489,000**	506,000±11,324	546,000	561,900±10,898
	W	233,000	254,300±19,247	238,000	252,200±15,069	**221,000**	249,900±14,433	317,000	345,300±17,218

Considerando os 5 objetivos trabalhados com as 4 variações no número de processadores, tem-se 20 cenários distintos. O AEMMD atingiu o melhor resultado em 60% dos cenários, seguido pelo AEMMT1 com 35%, na sequência AEMMT2 com 25% e por último, com o pior resultado, o NSGA-II em apenas 5% (ou seja, um único cenário e todos os demais algoritmos atingiram o melhor resultado nesse cenário). Na execução com 2 processadores houve um empate entre os três AGs de múltiplos objetivos: AEMMT1, AEMMT2 e AEMMD, todos tendo atingido melhores resultados em três das cinco métricas trabalhadas. Com 4 processadores, O AEMMT1 se sobressaiu, seguido pelo AEMMD. Além disso, o AEMMD dominou os resultados com 8 e 16 processadores. Com 8, foi o melhor em três métricas e com 16, em quatro.

5.3.3 Melhores indivíduos encontrados pelos AEMOs

Como dito na Seção 4.4, cada algoritmo tem como retorno para solução do problema a fronteira dos indivíduos não-dominados. Dentro desse conjunto, foi aplicada uma avaliação para identificar a melhor solução. Essa avaliação consiste em normalizar os dados das métricas e utilizá-los através de uma função escalar para calcular a aptidão do indivíduo. Esse cálculo utilizou duas medidas: média simples e média harmônica.

A seguir, são apresentados os resultados produzidos pelos algoritmos em cada um dos DAGs trabalhados. Foi feita uma análise comparativa entre os resultados dos AGs, a fim de identificar alguma relação nos resultados obtidos com o aumento do número de tarefas.

5.3.3.1 Média Simples

A Tabela 4 apresenta os resultados produzidos pelos AGs em cada um dos cenários mapeados para o DAG de 50 tarefas. Considerando o cenário de 5 objetivos o AEMMT1 se sobressaiu em 75% dos casos. Ele foi superado pelo AEMMD apenas no cenário com 8 processadores. Na execução com 4 objetivos o AEMMT1 foi melhor em todos os cenários. Em se tratando de 3 objetivos, houve um empate entre AEMMT1 e AEMMD, este melhor com mais processadores e aquele melhor com menos processadores. Novamente, na execução com 2 processadores, o AEMMT1 foi melhor em todos os cenários, inclusive com valores bem superiores se comparados ao AEMMD e ao NSGA-II, exceto na execução com 8 processadores onde seus valores ficaram próximos aos do AEMMD. O NSGA-II não produziu bons resultados nos cenários com 5 e 4 objetivos. Suas melhores execuções foram nos ambientes com 2 e 3 processadores. Sua melhor execução foi com 2 objetivos e 2 processadores onde seu resultado ficou bem próximo ao do AEMMD.

Para o DAG de 100 tarefas, o resultado geral não foi muito diferente em relação a execução anterior. O AEMMT1 dominou na totalidade as execuções com 4, 3 e 2 objetivos, perdendo para o AEMMD apenas em dois cenários: com 4 e 16 processadores utilizando 5 objetivos, mas mesmo assim foi superado por muito pouco. Todos os AGs na

Tabela 4 – Melhores indivíduos com média simples encontrados por AEMO em um DAG de 50 tarefas.

Obj.	m	**AEMMT1**		**AEMMD**		**NSGA-II**	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**
5	2	**0,8813**	0,8178±0,0323	0,8416	0,7871±0,0366	0,5838	0,5487±0,0237
	4	**0,9016**	0,7865±0,0652	0,8809	0,7549±0,0550	0,5183	0,4686±0,0239
	8	0,7959	0,7563±0,0354	**0,8210**	0,7198±0,0556	0,5356	0,5015±0,0189
	16	**0,8151**	0,7543±0,0427	0,7724	0,7381±0,0222	0,5623	0,5248±0,0276
4	2	**0,8970**	0,8355±0,0288	0,8280	0,8025±0,0172	0,6275	0,5982±0,0247
	4	**0,8959**	0,8138±0,0454	0,7533	0,7020±0,0419	0,5503	0,5093±0,0211
	8	**0,8523**	0,7849±0,0462	0,8128	0,7502±0,0429	0,5750	0,5450±0,0235
	16	**0,8268**	0,7857±0,0285	0,8135	0,7138±0,0500	0,6335	0,5727±0,0314
3	2	**0,9757**	0,9285±0,0436	0,9464	0,8763±0,0716	0,7546	0,6838±0,0383
	4	**0,8629**	0,8074±0,0467	0,8122	0,7331±0,0494	0,6067	0,5428±0,0339
	8	0,8632	0,7652±0,0512	**0,8729**	0,6982±0,0732	0,5858	0,5392±0,0333
	16	0,7747	0,7439±0,0269	**0,7841**	0,6955±0,0587	0,6175	0,5461±0,0366
2	2	**0,9728**	0,9404±0,0266	0,8222	0,7915±0,0323	0,7998	0,7530±0,0309
	4	**0,8827**	0,8049±0,0393	0,7123	0,6723±0,0219	0,6633	0,6158±0,0294
	8	**0,7711**	0,7424±0,0213	0,7303	0,6418±0,0576	0,6231	0,5855±0,0259
	16	**0,8270**	0,7384±0,0549	0,6644	0,6101±0,0381	0,6401	0,5675±0,0313

média geraram melhores indivíduos se comparados à execução com o DAG de 50 tarefas. A Tabela 5 deixa claro que o NSGA-II continuou não gerando bons indivíduos e sua melhor execução foi com 2 objetivos e 2 processadores, com o valor final do indivíduo um pouco menor se comparado ao melhor indivíduo dele com o DAG de 50 tarefas.

Tabela 5 – Melhores indivíduos com média simples encontrados por AEMO em um DAG de 100 tarefas.

Obj.	m	**AEMMT1**		**AEMMD**		**NSGA-II**	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**
5	2	**0,8644**	0,8395±0,0171	0,8446	0,8075±0,0263	0,6348	0,5804±0,0295
	4	0,8641	0,8070±0,0316	**0,8689**	0,7880±0,0392	0,5896	0,5494±0,0234
	8	**0,9065**	0,8335±0,0397	0,8963	0,8270±0,0406	0,5748	0,5528±0,0189
	16	0,8861	0,8633±0,0198	**0,8874**	0,8513±0,0205	0,6654	0,6174±0,0256
4	2	**0,9264**	0,8914±0,0203	0,8925	0,8394±0,0309	0,6486	0,6192±0,0242
	4	**0,9158**	0,8672±0,0476	0,8550	0,8068±0,0357	0,6293	0,5876±0,0207
	8	**0,8759**	0,8526±0,0168	0,8509	0,8200±0,0212	0,6295	0,5996±0,0209
	16	**0,9063**	0,8741±0,0155	0,8841	0,8452±0,0296	0,6969	0,6568±0,0277
3	2	**0,9746**	0,9189±0,0331	0,9410	0,8739±0,0511	0,6977	0,6545±0,0306
	4	**0,8593**	0,8301±0,0320	0,7618	0,6993±0,0524	0,5735	0,5368±0,0260
	8	**0,8400**	0,8001±0,0296	0,7766	0,7092±0,0389	0,5728	0,5378±0,0232
	16	**0,8881**	0,8428±0,0374	0,8430	0,7634±0,0603	0,6444	0,5900±0,0370
2	2	**0,9791**	0,9008±0,0410	0,8159	0,7710±0,0341	0,7448	0,7110±0,0308
	4	**0,8572**	0,8058±0,0332	0,7560	0,6897±0,0333	0,6580	0,6149±0,0229
	8	**0,8421**	0,7844±0,0347	0,6713	0,6297±0,0319	0,6209	0,6004±0,0134
	16	**0,8698**	0,8128±0,0360	0,7913	0,6879±0,0591	0,6897	0,6155±0,0326

Com o aumento de 100 para 300 tarefas, percebe-se novamente a melhora do AEMMT1.

A tabela 6 confirma o domínio desse algoritmo. Dessa vez ele não foi superado por nenhum outro. Produziu melhores indivíduos para todas as combinações entre quantidades de objetivos e quantidades de processadores. Outra detalhe importante percebido nessa execução é que o valor obtido pelo AEMMT1 é em média 12% melhor que o AEMMD e 60% melhor que o NSGA-II. Esse número aumentou bastante nessa execução. Nos DAGs de 50 e 100 tarefas esse valores foram similares entre si: em torno de 7% de aumento do AEMMD para o AEMMT1 e 39% de aumento do NSGA-II para o AEMMT1.

Tabela 6 – Melhores indivíduos com média simples encontrados por AEMO em um DAG de 300 tarefas.

Obj.	m	**AEMMT1**		**AEMMD**		**NSGA-II**	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**
5	2	**0,8908**	0,8483±0,0320	0,8551	0,8184±0,0354	0,5491	0,5315±0,0172
	4	**0,8612**	0,7947±0,0427	0,8187	0,7109±0,0560	0,4986	0,4891±0,0086
	8	**0,8739**	0,8170±0,0531	0,8680	0,8384±0,0249	0,5425	0,5149±0,0167
	16	**0,8729**	0,7792±0,0455	0,8431	0,7683±0,0411	0,5132	0,4953±0,0134
4	2	**0,9734**	0,8834±0,0584	0,8605	0,7947±0,0403	0,5924	0,5539±0,0200
	4	**0,9064**	0,8565±0,0401	0,8219	0,7610±0,0375	0,5406	0,5290±0,0074
	8	**0,8967**	0,8196±0,0505	0,8722	0,7848±0,0454	0,5521	0,5238±0,0117
	16	**0,9117**	0,8389±0,0433	0,8251	0,7840±0,0293	0,5569	0,5392±0,0119
3	2	**0,8931**	0,8501±0,0334	0,8638	0,7342±0,0585	0,5924	0,5372±0,0286
	4	**0,8695**	0,7715±0,0462	0,7288	0,6198±0,0542	0,4851	0,4662±0,0085
	8	**0,8565**	0,7595±0,0556	0,6961	0,5889±0,0709	0,4428	0,4057±0,0218
	16	**0,7979**	0,7526±0,0467	0,6700	0,6035±0,0593	0,4488	0,4254±0,0174
2	2	**0,9155**	0,8563±0,0466	0,7410	0,6684±0,0426	0,6633	0,6297±0,0177
	4	**0,8017**	0,7458±0,0356	0,6592	0,6296±0,0243	0,5863	0,5735±0,0100
	8	**0,7415**	0,6837±0,0388	0,5946	0,5649±0,0212	0,5509	0,5229±0,0160
	16	**0,7782**	0,7327±0,0340	0,6680	0,5995±0,0377	0,5669	0,5282±0,0211

Os gráficos a seguir comprovam a tendência de execução dos AGs a cada variação na quantidade de objetivos. A Figura 36 mostra a evolução no ambiente de 50 tarefas. O AEMMD e o NSGA-II geraram resultados mais similares entre os processadores. Com 2, 4 e 8 o AEMMD manteve o padrão: diminuiu o valor do indivíduo com 4 objetivos, aumentou esse valor com 3 objetivos e diminuiu com 2, saindo desse modelo apenas com 16 processadores. O NSGA-II manteve uma evolução de subida em todas as execuções. Já o AEMMT1 não demonstrou padrão, apenas similaridades entre 4 e 16 processadores.

A Figura 37 apresenta os resultados para o DAG de 100 tarefas. O AEMMT1 seguiu aumentando seu valor na execução com 2 processadores, mas nas demais oscilou bastante. Assim como o AEMMT1, o AEMMD gerou aumento no valor do indivíduo apenas na execução com 2 processadores. Nas demais, houve apenas queda. O NSGA-II manteve o padrão obtido no DAG de 50 tarefas apenas na execução de 2 processadores, nos demais ele aumentava o valor com 4 processadores, tinha uma queda com 3 e aumentava novamente com 2, onde na maioria dos casos é sua melhor execução.

Figura 36 – Tendência do melhor indivíduo por AEMO no DAG de 50 tarefas com redução de objetivos - Média Simples

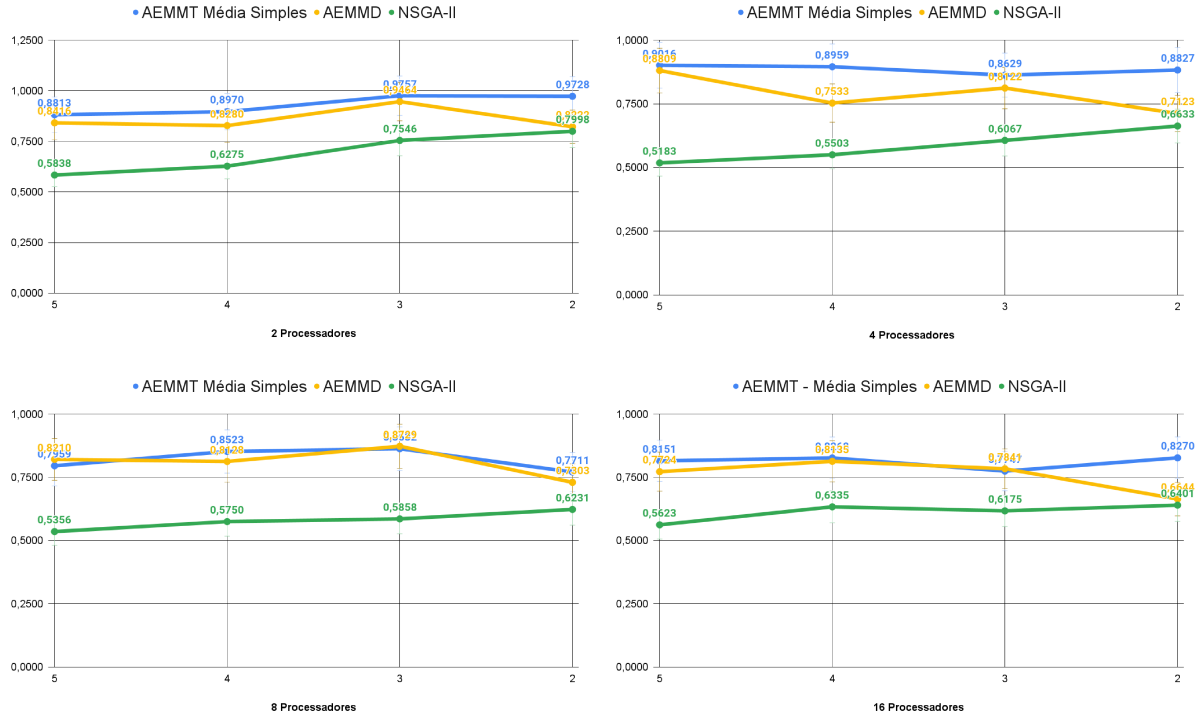
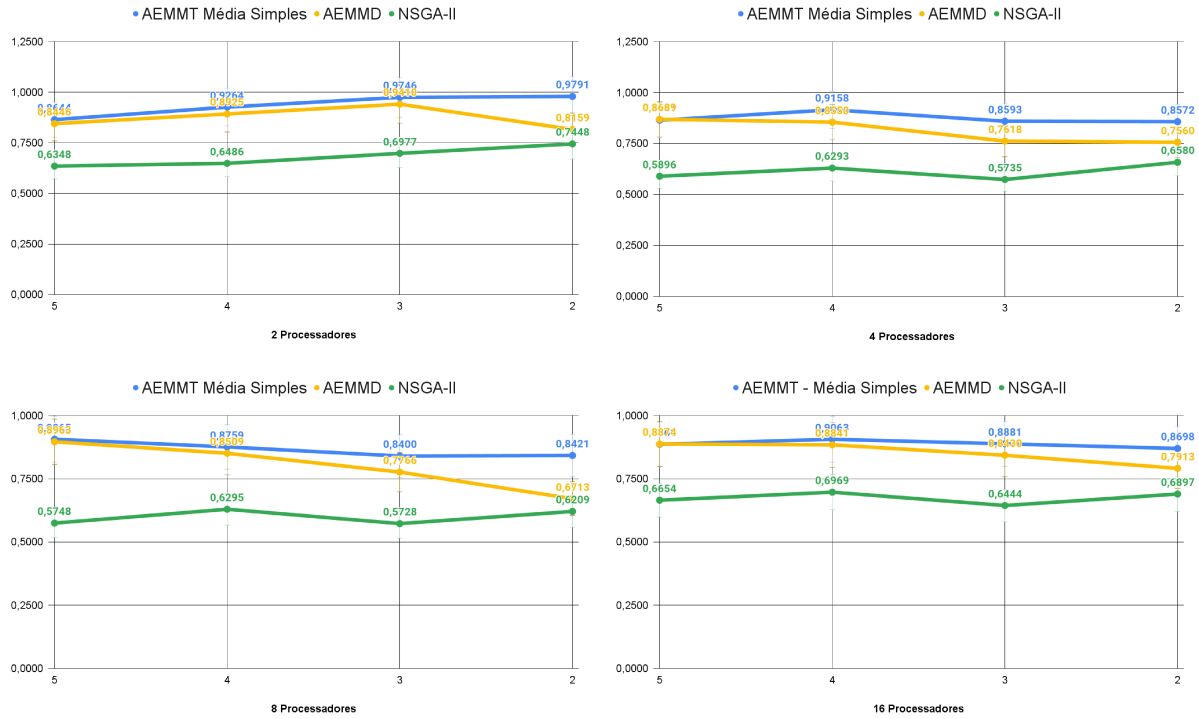
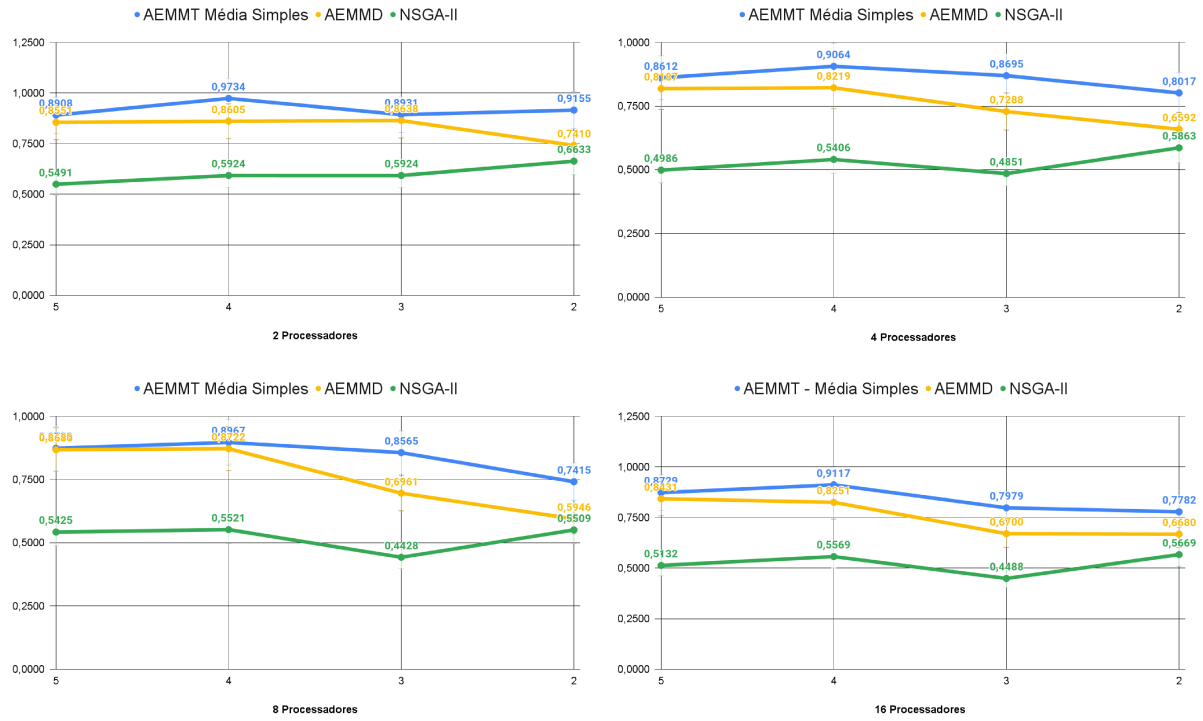


Figura 37 – Tendência do melhor indivíduo por AEMO no DAG de 100 tarefas com redução de objetivos - Média Simples



Na execução do DAG de 300 tarefas o AEMMT1 manteve um padrão nas execuções com 4, 8 e 16 processadores: sempre produzia um indivíduo pior que a execução anterior a partir de 4 objetivos. O AEMMD e o NSGA-II geraram resultados similares em todas as execuções. É possível visualizar através da Figura 38 que as linhas que representam esses objetivos são semelhantes a cada variação no número de processadores.

Figura 38 – Tendência do melhor indivíduo por AEMO no DAG de 300 tarefas com redução de objetivos - Média Simples



5.3.3.2 Média Harmônica

A Tabela 7 apresenta os resultados produzidos pelos AGs em cada um dos cenários mapeados para o DAG de 50 tarefas. Considerando o cenário de 5 objetivos houve empate entre o AEMMT2 e o AEMMD. Ambos foram melhores em duas configurações: o AEMMT2 com 2 e 16 processadores e o AEMMD nos demais. O ambiente com 4 e 2 processadores foi completamente dominado pelo AEMMT2. Esse algoritmo só não foi o melhor em todos os cenários com 3 objetivos, pois o AEMMD se sobressaiu na execução com 8 processadores. No geral o NSGA-II não gerou bons resultados quando comparado aos demais. Suas melhores execuções foram com 2 e 3 objetivos, ambas na configuração de 2 processadores.

Analisando os resultados do cenário de 100 tarefas, percebe-se que o AEMMT2 melhorou em relação a execução anterior. Antes ele tinha perdido em 3 configurações, mas nesse cenário ele perdeu apenas na configuração de 8 processadores com 5 objetivos, mas mesmo

Tabela 7 – Melhores indivíduos com média harmônica encontrados por AEMO em um DAG de 50 tarefas.

Obj.	m	**AEMMT2**		**AEMMD**		**NSGA-II**	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**
5	2	**0,8436**	0,7446±0,0379	0,7764	0,7241±0,0389	0,5838	0,5487±0,0237
	4	0,8055	0,7602±0,0373	**0,8615**	0,7324±0,0564	0,5183	0,4686±0,0239
	8	0,7674	0,7282±0,0354	**0,7950**	0,7098±0,0527	0,5356	0,5015±0,0189
	16	**0,8292**	0,7471±0,0468	0,7473	0,7179±0,0232	0,5623	0,5248±0,0276
4	2	**0,8445**	0,7888±0,0342	0,7638	0,7276±0,0309	0,5805	0,5286±0,0389
	4	**0,8171**	0,7695±0,0331	0,7256	0,6681±0,0477	0,5050	0,4682±0,0232
	8	**0,8382**	0,7817±0,0419	0,8124	0,7442±0,0463	0,5672	0,5244±0,0337
	16	**0,8570**	0,7813±0,0480	0,7880	0,7056±0,0447	0,6279	0,5664±0,0322
3	2	**0,9777**	0,9480±0,0275	0,9457	0,8673±0,0804	0,7225	0,6534±0,0406
	4	**0,9314**	0,8324±0,0748	0,8025	0,7198±0,0514	0,5651	0,4991±0,0395
	8	0,8257	0,7698±0,0485	**0,8688**	0,6904±0,0804	0,5759	0,5149±0,0432
	16	**0,8400**	0,7665±0,0473	0,7742	0,6901±0,0593	0,6139	0,5411±0,0375
2	2	**0,9930**	0,9443±0,0276	0,7890	0,7439±0,0450	0,7556	0,6910±0,0494
	4	**0,8662**	0,7927±0,0473	0,6578	0,6160±0,0355	0,6094	0,5341±0,0445
	8	**0,8096**	0,7376±0,0349	0,6932	0,6137±0,0575	0,5939	0,5484±0,0365
	16	**0,8157**	0,7565±0,0379	0,6574	0,5830±0,0379	0,6397	0,5512±0,0408

assim, teve seu valor bem próximo ao do AEMMD, vencedor desta etapa. Comparando a média dos valores em relação a execução anterior, todos os AGs tiveram um ligeiro aumento. Novamente, o NSGA-II manteve o padrão e produziu seu melhor indivíduo na configuração de 2 objetivos com 2 processadores. A Tabela 8 comprova essa análise.

Tabela 8 – Melhores indivíduos com média harmônica encontrados por AEMO em um DAG de 100 tarefas.

Obj.	m	**AEMMT2**		**AEMMD**		**NSGA-II**	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**
5	2	**0,8827**	0,8272±0,0296	0,8119	0,7709±0,0272	0,6348	0,5804±0,0295
	4	**0,8844**	0,8139±0,0403	0,8606	0,7730±0,0431	0,5896	0,5494±0,0234
	8	0,8706	0,8090±0,0453	**0,8864**	0,8204±0,0411	0,5748	0,5528±0,0189
	16	**0,9034**	0,8662±0,0231	0,8803	0,8473±0,0196	0,6654	0,6174±0,0256
4	2	**0,9086**	0,8542±0,0227	0,8548	0,8017±0,0312	0,6151	0,5823±0,0261
	4	**0,8959**	0,8482±0,0336	0,8502	0,7988±0,0398	0,5892	0,5325±0,0263
	8	**0,8861**	0,8365±0,0267	0,8506	0,8170±0,0238	0,5972	0,5527±0,0338
	16	**0,9028**	0,8696±0,0224	0,8834	0,8433±0,0305	0,6744	0,6309±0,0307
3	2	**0,9727**	0,9258±0,0400	0,9396	0,8683±0,0549	0,6655	0,6097±0,0364
	4	**0,9017**	0,7800±0,0759	0,7464	0,6703±0,0650	0,5422	0,4856±0,0281
	8	**0,8703**	0,8077±0,0469	0,7753	0,7003±0,0444	0,5435	0,5032±0,0358
	16	**0,8785**	0,8486±0,0252	0,8409	0,7614±0,0617	0,6358	0,5804±0,0350
2	2	**0,9690**	0,9102±0,0391	0,7892	0,7098±0,0582	0,7014	0,6455±0,0379
	4	**0,8389**	0,7843±0,0458	0,7475	0,6135±0,0615	0,5748	0,5221±0,0290
	8	**0,8423**	0,7751±0,0400	0,6379	0,5533±0,0492	0,5770	0,5277±0,0381
	16	**0,8440**	0,8149±0,0282	0,7904	0,6644±0,0735	0,6725	0,5961±0,0388

No último ambiente (Tabela 9) ficou claro o domínio do AEMMT2: superou os outros

algoritmos em todos os cenários. Em algumas configurações venceu por pouco o AEMMD como por exemplo na configuração de 3 objetivos com 2 processadores, mas na grande maioria, a diferença foi alta. Na média tem seus valores superiores ao AEMMD em 16% e se comparados ao NSGA-II essa média aumenta para mais de 80%. Esse ambiente fez com que o NSGA-II produzisse os piores resultados de todas as avaliações.

Tabela 9 – Melhores indivíduos com média harmônica encontrados por AEMO em um DAG de 300 tarefas.

Obj.	m	**AEMMT2**		**AEMMD**		**NSGA-II**	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**
5	2	**0,9622**	0,822±0,0574	0,8499	0,7956±0,0472	0,5491	0,5315±0,0172
	4	**0,8021**	0,7375±0,0461	0,7959	0,6806±0,0625	0,4986	0,4891±0,0086
	8	**0,8725**	0,8095±0,0365	0,8587	0,8304±0,0279	0,5425	0,5149±0,0167
	16	**0,8791**	0,7674±0,0559	0,8320	0,7515±0,0445	0,5132	0,4953±0,0134
4	2	**0,8750**	0,8369±0,0292	0,8508	0,7823±0,0417	0,5255	0,4728±0,0334
	4	**0,8792**	0,8229±0,0402	0,8200	0,7396±0,0456	0,4244	0,3778±0,0207
	8	**0,8832**	0,8257±0,0470	0,8638	0,7713±0,0532	0,4416	0,3593±0,0323
	16	**0,8768**	0,8162±0,0309	0,8158	0,7746±0,0309	0,4414	0,4251±0,0121
3	2	**0,8882**	0,8658±0,0156	0,8536	0,6994±0,0757	0,5064	0,4438±0,0393
	4	**0,8565**	0,7392±0,0483	0,7005	0,5527±0,0714	0,3742	0,3266±0,0206
	8	**0,8542**	0,7313±0,0702	0,6807	0,5568±0,0894	0,3775	0,2946±0,0309
	16	**0,8273**	0,7179±0,0597	0,6667	0,5915±0,0674	0,3684	0,3546±0,0112
2	2	**0,9460**	0,8400±0,0619	0,6674	0,5293±0,0805	0,5398	0,4814±0,0329
	4	**0,8166**	0,7174±0,0634	0,5975	0,4708±0,0671	0,3999	0,3632±0,0250
	8	**0,7259**	0,6602±0,0588	0,5306	0,4356±0,0646	0,4365	0,3391±0,0364
	16	**0,7837**	0,7366±0,0353	0,6077	0,5339±0,0639	0,4513	0,4171±0,0209

A tendência do AEMMT2 foi correlata se comparar em pares as execuções de 2 com 4 processadores e de 8 com 16, mas distintas de um par para o outro. O AEMMD gerou um certo padrão em quase todas as execuções, exceto com 16 processadores onde gerou 2 quedas consecutivas. O NSGA-II por sua vez manteve um acréscimo a cada redução na quantidade de objetivos. Isso ocorreu para todas as configurações de processadores, conforme observado na Figura 39.

No DAG com 100 tarefas, o AEMMT2 gerou incremento total apenas na execução com 2 processadores, nos demais houve bastante oscilação. Para o AEMMD houve queda em praticamente toda execução com 4, 8 e 16 processadores. Ele teve um ligeiro aumento na execução com 2 processadores e mesmo assim apenas nas execuções de 4 e 3 objetivos. O NSGA-II trouxe um resultado atípico. Na configuração de 8 e 16 processadores o resultado foi similar: aumento para 4 objetivos, queda para 3 e aumento novamente para 2 objetivos. Mas para as execuções de 2 e 4 processadores, os resultados obtidos foram inversos um do outro, conforme pode ser visto na Figura 40.

Figura 39 – Tendência do melhor indivíduo por AEMO no DAG de 50 tarefas com redução de objetivos - Média Harmônica

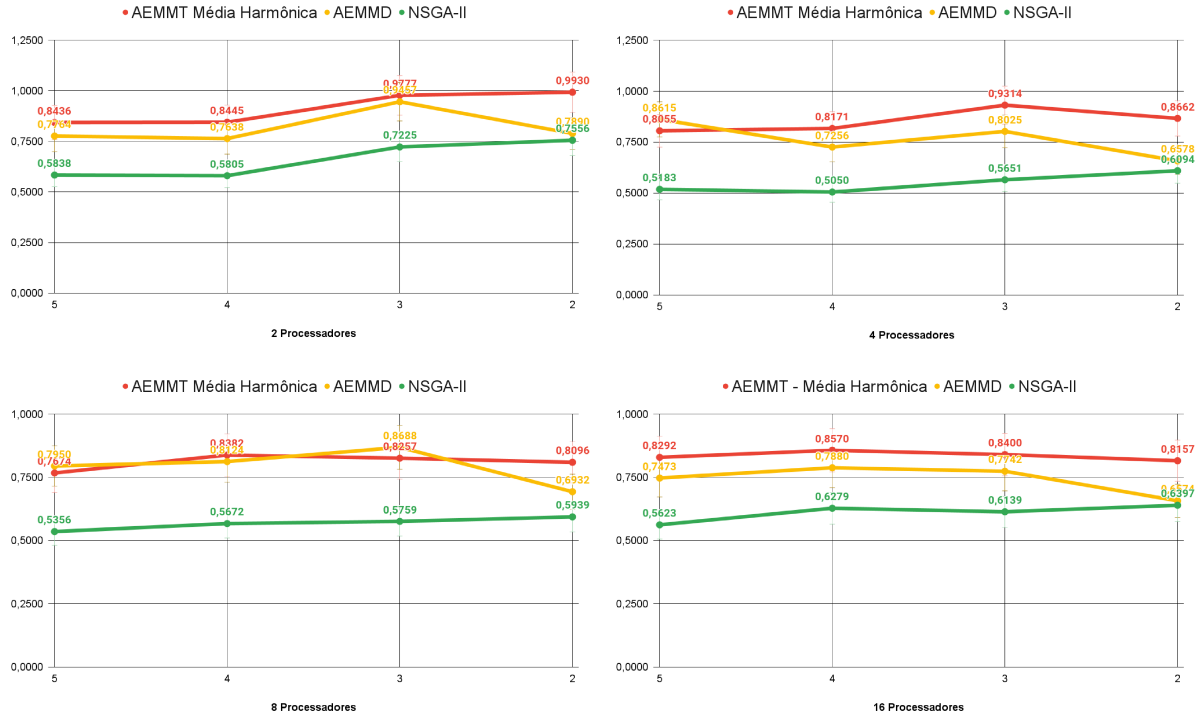
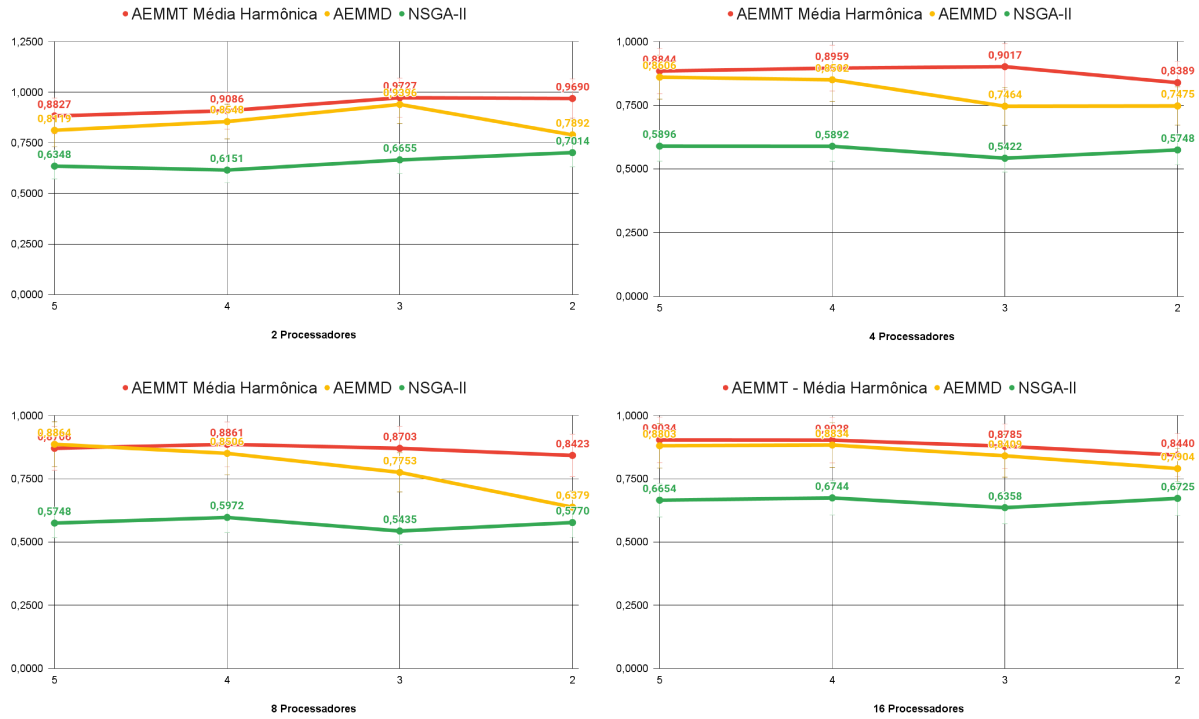
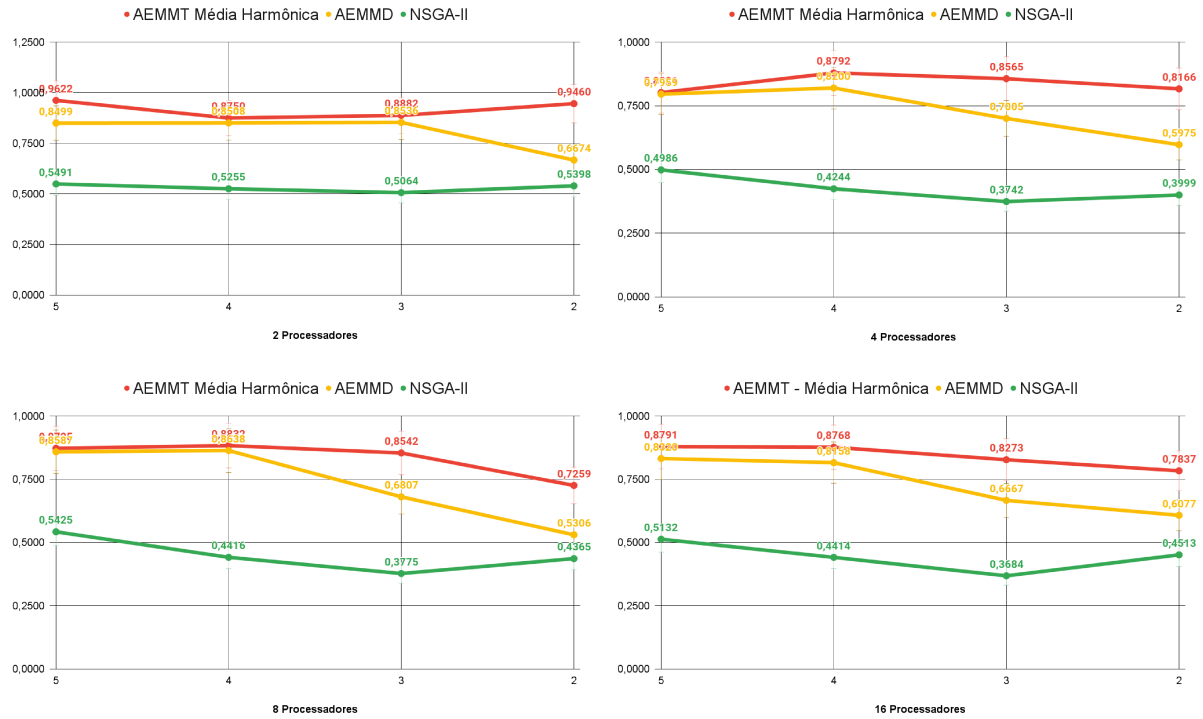


Figura 40 – Tendência do melhor indivíduo por AEMO no DAG de 100 tarefas com redução de objetivos - Média Harmônica



A Figura 41 determina os resultados obtidos através da execução dos AGs pelo DAG de 300 tarefas. O AEMMT2 manteve um padrão nas execuções com 4, 8 e 16 processadores: sempre produzia um indivíduo pior que a execução anterior a partir de 4 objetivos. O AEMMD produz queda nos valores dos indivíduos a partir de 4 objetivos para todos os processadores. O NSGA-II por sua vez apresenta resultados homogêneos: a cada redução no número de objetivos ocorre a queda no valor do indivíduo com exceção da redução para 2 objetivos, ambiente em que normalmente esse algoritmo produz seus melhores resultados.

Figura 41 – Tendência do melhor indivíduo por AEMO no DAG de 300 tarefas com redução de objetivos - Média Harmônica



5.3.4 Teste Estatístico

Para avaliar os resultados obtidos pelo AEMMT, AEMMD e NSGA-II, foi realizado um teste estatístico. De acordo com Demšar (2006), o teste de Wilcoxon pode ser adotado para analisar uma diferença pareada entre duas técnicas sobre um conjunto de experimentos. Considerando um nível de significância $\alpha = 0{,}05$, a hipótese nula afirma que ambas as técnicas são equivalentes para qualquer número de objetivos.

As Tabelas 10 e 11 e demonstram os resultados das três técnicas em termos de média das melhores soluções para 5 objetivos, considerando média simples e média harmônica, respectivamente. Os valores em negrito indicam o melhor desempenho geral para cada

DAG e o número de processadores. Na parte inferior das tabelas, pode-se ver também os resultados do teste de Wilcoxon.

Tabela 10 – Desempenho médio dos melhores indivíduos em termos de média simples para o problema de otimização dos 5 objetivos

DAG	m	AEMMT	AEMMD	NSGA-II
50	2	**0.8178±0.0323**	0.7871±0.0366	0.5487±0.0237
	4	**0.7865±0.0652**	0.7549±0.0550	0.4686±0.0239
	8	**0.7563±0.0354**	0.7198±0.0556	0.5015±0.0189
	16	**0.7543±0.0427**	0.7381±0.0222	0.5248±0.0276
100	2	**0.8395±0.0171**	0.8075±0.0263	0.5804±0.0295
	4	**0.8070±0.0316**	0.7880±0.0392	0.5494±0.0234
	8	**0.8335±0.0397**	0.8270±0.0406	0.5528±0.0189
	16	**0.8633±0.0198**	0.8513±0.0205	0.6174±0.0256
300	2	**0.8483±0.0320**	0.8184±0.0354	0.5315±0.0172
	4	**0.7947±0.0427**	0.7109±0.0560	0.4891±0.0086
	8	0.8170±0.0531	**0.8384±0.0249**	0.5149±0.0167
	16	**0.7792±0.0455**	0.7683±0.0411	0.4953±0.0134
AEMMT		-	✓	✓
AEMMD		×	-	✓
NSGA-II		×	×	-

Tabela 11 – Desempenho médio dos melhores indivíduos em termos de média harmônica para o problema de otimização dos 5 objetivos

DAG	m	AEMMT	AEMMD	NSGA-II
50	2	**0.7446±0.0379**	0.7241±0.0389	0.5487±0.0237
	4	**0.7602±0.0373**	0.7324±0.0564	0.4686±0.0239
	8	**0.7282±0.0354**	0.7098±0.0527	0.5015±0.0189
	16	**0.7471±0.0468**	0.7179±0.0232	0.5248±0.0276
100	2	**0.8272±0.0296**	0.7709±0.0272	0.5804±0.0295
	4	**0.8139±0.0403**	0.7730±0.0431	0.5494±0.0234
	8	0.8090±0.0453	**0.8204±0.0411**	0.5528±0.0189
	16	**0.8662±0.0231**	0.8473±0.0196	0.6174±0.0256
300	2	**0.8228±0.0574**	0.7956±0.0472	0.5315±0.0172
	4	**0.7375±0.0461**	0.6806±0.0625	0.4891±0.0086
	8	0.8095±0.0365	**0.8304±0.0279**	0.5149±0.0167
	16	**0.7674±0.0559**	0.7515±0.0445	0.4953±0.0134
AEMMT		-	✓	✓
AEMMD		×	-	✓
NSGA-II		×	×	-

Para cada tabela, os testes estatísticos rejeitam a hipótese nula em relação a AEMMT vs. AEMMD, AEMMT vs. NSGA-II e AEMMD vs. NSGA-II. Consequentemente, existem evidências de que o AEMMT supera tanto AEMMD quanto NSGA-II e que o AEMMD supera o NSGA-II. Essa relação pode ser percebida tanto no desempenho da média simples quanto da média harmônica.

O resultado obtido é muito relevante, pois destaca as principais características do AEMMT para o problema de escalonamento de tarefas em ambientes multiprocessados. Por fim, vale ressaltar que os resultados atingidos na execução com 5 objetivos e apresentados nas tabelas anteriores, também se concretizaram nas execuções com 4, 3 e 2 objetivos.

Capítulo 6

Conclusões e Trabalhos Futuros

A utilização de sistemas computacionais aumentou significativamente ao longo dos anos. Cada vez mais, espera-se que esses ambientes sejam capazes de atender às demandas massivas dos usuários. Para que essa necessidade seja atendida, os sistemas precisam utilizar um algoritmo de escalonamento eficiente, capaz de distribuir as tarefas de forma inteligente para os processadores disponíveis e buscando otimizar as métricas de desempenho requeridas.

Existem diversas abordagens diferentes para lidar com o problema do escalonamento de tarefas. Para este trabalho, foi escolhida uma abordagem não-determinística, capaz de gerar diferentes resultados dependendo das escolhas realizadas durante sua execução. A abordagem escolhida baseia-se em algoritmos evolutivos.

O objetivo principal deste trabalho foi implementar um AE capaz de encontrar soluções de escalonamento considerando um número relativamente alto de tarefas. Por se tratar de um problema combinatorial, foi necessário um AE eficiente, a fim de obter soluções adequadas em um tempo razoável. Além disso, foi ressaltada a ideia de trabalhar com múltiplos objetivos, de modo a gerar uma contribuição para área de otimização.

Dessa forma, foram implementados três AEMOs: o NSGA-II (*Non-Dominated Sorting Genetic Algorithm II*), um AG bastante abordado na literatura, o AEMMT (Algoritmo Evolutivo Multiobjetivo com Muitas Tabelas) e o AEMMD (Algoritmo Evolutivo Multiobjetivo com Múltiplas Dominâncias). Os dois últimos fazem parte de uma nova classe de algoritmos, criada para trabalhar com grandes quantidades de objetivos, sendo variações mais recentes (BRASIL; DELBEM; SILVA, 2013; LAFETÁ et al., 2018).

Os experimentos foram realizados em diferentes configurações, contemplando 5, 4, 3 e 2 objetivos. Foram consideradas funções objetivo de minimização que fazem sentido para o problema do escalonamento de tarefas: *makespan*, balanceamento de carga, *flowtime*, custo de comunicação e o tempo de espera. Para atender as metas definidas para esta pesquisa de mestrado, ou seja, trabalhar com grandes quantidades de tarefas, foram utilizados três DAGs diferentes, com 50, 100 e 300 tarefas. Além desses cenários, também houve uma variação na quantidade de processadores disponíveis para executar as tarefas.

Foram realizadas execuções com 2, 4, 8 e 16 processadores. No fim, foram gerados 48 experimentos.

A primeira análise realizada foi voltada para os objetivos de forma individual. Foram realizadas diversas execuções para cada uma das configurações, de modo a obter limites máximos e mínimos para cada um dos objetivos. De posse desses dados, foi feito um estudo para verificar quais AGs estava produzindo as melhores soluções de acordo com as métricas trabalhadas. Essa análise atende o objetivo específico do trabalho de comparar os resultados obtidos com as diferentes métricas utilizadas.

Para o *makespan*, tanto o AEMMT com média simples quanto o AEMMT com média harmônica produziram a maioria dos indivíduos no intervalo dos melhores valores para essa métrica, principalmente na execução para 5 e 4 objetivos. Os valores caíram um pouco para 3 objetivos e despencaram para 2. O AEMMD manteve um certo padrão até 3 objetivos, mas na última execução não gerou nenhum indivíduo no melhor intervalo. Em nenhuma das execuções o NSGA-II forneceu um indivíduo no melhor intervalo de *makespan*. O balanceamento de carga, por sua vez, teve um comportamento diferente. Todo os algoritmos produziram bons resultados para a execução com 2 e 4 processadores considerando todas as variações de objetivos. Contudo, quando o número de processadores aumentou para 8 ou 16, o percentual obtido no melhor intervalo diminuiu, chegando a zero em algumas execuções.

Para o *Flowtime*, as duas versões do AEMMT se sobressaíram sobre os demais algoritmos: a versão com média simples dominou o intervalo de melhores soluções para 5 e 4 objetivos e a versão com média harmônica em 3 objetivos. O AEMMD também conseguiu gerar alguns indivíduos no melhor intervalo, mas o percentual obtido não passava da metade dos valores do AEMMT. O NSGA-II também não gerou bons resultados para essa métrica, ficando aquém dos demais. Na métrica do custo de comunicação, o AEMMD teve melhores resultados que os demais algoritmos, contrariando a superioridade do AEMMT nas anteriores. O AEMMD foi superior na execução com 5 objetivos e empatou com o AEMMT com média harmônica na execução com 4 objetivos. A última métrica utilizada foi o tempo de espera. Para esse objetivo, o AEMMT foi o melhor algoritmo. Sua versão com média simples dominou as execuções com 8 e 16 processadores e a versão com média harmônica dominou as execuções com 2 e 4.

O último objetivo específico do trabalho era comparar os resultados obtidos entre os AGs implementados. Para definir o AG que produziu o melhor indivíduo do conjunto de soluções não-dominadas, foram utilizadas duas estratégias que combinavam os objetivos trabalhados nessa pesquisa em uma função escalar: média aritmética (média simples) e média harmônica. O valor calculado é considerado a função de aptidão do indivíduo.

Primeiramente, fazendo análise da média simples, e considerando o DAG de 50 tarefas, o AEMMT teve um resultado extremamente superior em relação aos demais AGs. Ele obteve o melhor resultado em 81,25% dos cenários enquanto que o AEMMD se sobressaiu

apenas em 18,75%. O NSGA-II não conseguiu vencer os outros AGs em nenhum cenário. O AEMMD conseguiu confrontar o AEMMT apenas na execução com 3 objetivos, onde cada um obteve o melhor resultado em 2 cenários.

No DAG de 100 tarefas, o AEMMT ainda se sobressaiu um pouco mais em relação aos demais. Ele subiu para 87,50% dos cenários como produtor do melhor indivíduo. Com 5 objetivos, o AEMMD conseguiu enfrentá-lo de igual para igual, venceu em 2 cenários (4 e 16 processadores) e perdeu em 2 (2 e 8 processadores), mas em todos os cenários a diferença foi bastante pequena. Ao diminuir a quantidade de objetivos, o AEMMT melhorava seus resultados. Venceu em todos os cenários (4, 3 e 2 objetivos) aumentando a diferença para os demais. O NSGA-II continuou não fazendo frente aos outros algoritmos, porém, se comparado ao DAG de 50 tarefas, melhorou seus resultados. E um detalhe interessante: seus melhores indivíduos foram produzidos na execução com apenas 2 objetivos.

O último DAG utilizado foi de 300 tarefas. Nesse cenário, o AEMMT venceu com todas as combinações de objetivos e de processadores utilizadas e a diferença para os demais AGs foi bastante superior. Percebe-se que a medida que o número de tarefas aumenta, os resultados do AEMMT também melhoram.

Para a média harmônica, considerando o DAG de 50 tarefas, o AEMMT conseguiu produzir o melhor indivíduo em 81,25% dos cenários, similar ao ocorrido na utilização da média simples. A diferença aqui foi no cenário em que o AEMMD conseguiu competir com o AEMMT. Enquanto que na média simples foi com 3 objetivos, para a média harmônica foi com 5 objetivos. Houve um empate. O AEMMT venceu nos extremos (2 e 16 processadores) e o AEMMD no meio (4 e 8 processadores). O NSGA-II não conseguiu competir com os demais, mas assim como na média simples, teve seus melhores resultados com 2 objetivos.

As execuções no DAG de 100 tarefas trouxeram um resultado um pouco melhor. o AEMMT venceu a execução em quase todos os cenários com exceção na execução com 8 processadores para 5 objetivos. Nesse cenário, o AEMMD foi o melhor algoritmo. Mesmo assim, com uma pequena diferença. Com o aumento das tarefas, o NSGA produziu melhores indivíduos. Novamente seu melhor resultado foi para 2 objetivos utilizando 2 processadores. Pelo histórico das gerações, percebe-se que este é o seu melhor cenário.

Por fim, na execução do DAG de 300 tarefas, o resultado obtido com a média harmônica foi o mesmo obtido com a média simples: o AEMMT produziu os melhores indivíduos em todos os cenários. Em alguns cenários o AEMMD ficou bem próximo, mas não conseguir se sobressair. Essa foi a pior configuração para execução do NSGA-II.

No geral, não houve muita variação entre os resultados obtidos com a médias simples e a média harmônica, o que mostra que a normalização dos dados por si só já amenizou as magnitudes distintas dos objetivos, mitigando o impacto dos possíveis *outliers*.

Podemos concluir, a partir de todas as análises realizadas, que o uso dos algoritmos de muitos objetivos (*many-objectives*) nas instâncias dos problemas do escalonamento de

tarefas produziram melhores resultados que a abordagem multiobjetivo padrão (NSGA-II). Além disso, para a função de aptidão utilizada, o AEMMT produziu os melhores resultados sendo bem superior aos seus concorrentes. Em alguns cenários, o AEMMD conseguiu fazer frente ao AEMMT e até vencê-lo, mas na grande maioria dos casos isso não ocorreu.

Esta pesquisa trabalhou com uma mescla de abordagens para avaliar um algoritmo genético multiobjetivo. A solução final produzida pelos algoritmos retornava um conjunto de indivíduos que formava a fronteira de Pareto. Dentro desse conjunto, foi aplicada uma função de aptidão que combinava os valores dos objetivos do indivíduo em uma função escalar, assim era possível determinar a melhor solução e quem a tinha produzido.

Existem outras formas de avaliar um algoritmo multiobjetivo. Algumas métricas que podem ser utilizadas são paramétricas, ou seja, dependem do conhecimento prévio da Fronteira de Pareto. Se não for possível determinar o Pareto ótimo, é gerado um Pareto aproximado (PA). Uma evolução desta pesquisa pode ser justamente trabalhar com essas métricas paramétricas:

- *Pareto subset* (PS): corresponde ao número de soluções de PS que pertencem a PA;
- *Error rate* (ER): é dada pelo percentual de elementos do Pareto PA não encontrados em PS (Pareto Solução obtido em uma execução do AEMO);
- *General distance* (GD): é a distância mínima entre os elementos de PA e PS. Quanto menor a distância, melhor a convergência.

Existem, ainda, outras métricas que não dependem do conhecimento prévio da fronteira de Pareto e que são estudos possíveis de serem abordados:

- *Spread*: busca avaliar o quão uniforme estão distribuídas as soluções de PS ao longo do espaço de busca. Quanto menor a métrica, maior a distribuição dos elementos;
- *Hiper-volume*: avalia a distribuição do conjunto de soluções em relação ao espaço de busca.

Além disso, seria interessante trabalhar com outros algoritmos bastante utilizados na literatura. Como pertencente a classe dos multiobjetivos o SPEA-II, que trabalha com o conceito de força e densidade e o MOEAD/D, que avalia os objetivos através de uma função de escala, são algoritmos interessantes de serem implementados e analisados. No que diz respeito a classe *many-objectives*, o NSGA-III é uma opção atraente a ser trabalhada uma vez que ele se diferencia do seu predecessor, NSGA-II, na fase de reinserção ou seleção natural.

Outro estudo interessante seria fazer as outras combinações de objetivos que não foram contempladas ao longo desta monografia. Finalmente, pretende-se incorporar a utilização

de outras funções objetivo, como o consumo energético dos processadores, medida que tem se tornado comum em ambientes de computação na nuvem.

Referências

AFRATI, F.; PAPADIMITRIOU, C. H.; PAPAGEORGIOU, G. Scheduling DAGs to minimize time and communication. In: REIF, J. H. (Ed.). **VLSI Algorithms and Architectures**. New York, NY: Springer-Verlag, 1988. (Lecture Notes in Computer Science, v. 319), p. 134–138. Disponível em: <https://doi.org/10.1007/BFb0040381>.

AHMAD, I.; SHEIKH, H. F. A multi-staged niched evolutionary approach for allocating parallel tasks with joint optimization of performance, energy, and temperature. **Journal of Parallel and Distributed Computing**, Elsevier BV, v. 134, p. 65–74, dez. 2019. Disponível em: <https://doi.org/10.1016/j.jpdc.2019.05.009>.

ARABNEJAD, H.; BARBOSA, J. G. List scheduling algorithm for heterogeneous systems by an optimistic cost table. **IEEE Transactions on Parallel and Distributed Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 25, n. 3, p. 682–694, mar. 2014. Disponível em: <https://doi.org/10.1109/TPDS.2013.57>.

BARBOSA, A. M.; RIBEIRO, L. de C.; ARANTES, J. M. de O. Algoritmo genético multiobjetivo: Sistema adaptativo com elitismo. In: **Proceedings of the 9th Brazilian Conference on Dynamics Control and their Applications**. Serra Negra, SP: [s.n.], 2010. p. 940–945.

BRASIL, C. R. S. **Algoritmo evolutivo de muitos objetivos para predição ab initio de estrutura de proteínas**. Tese (Doutorado) — Universidade de São Paulo, São Carlos, SP, maio 2012. Disponível em: <https://doi.org/10.11606/t.55.2012.tde-20072012-163056>.

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

BUENO, M. L.; OLIVEIRA, G. M. Multicast flow routing: Evaluation of heuristics and multiobjective evolutionary algorithms. In: IEEE. **IEEE Congress on Evolutionary Computation**. 2010. p. 1–8. Disponível em: <https://doi.org/10.1109/CEC.2010.5585942>.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **IEEE Transactions on Software Engineering**,

Institute of Electrical and Electronics Engineers, v. 14, n. 2, p. 141–154, 1988. Disponível em: <https://doi.org/10.1109/32.4634>.

CHAPIN, S. J.; WEISSMAN, J. B. Distributed and multiprocessor scheduling. In: **Computer Science Handbook**. 2. ed. [S.l.]: CRC Press, 2004. cap. 88, p. 88–1–88–19.

CHEN, H.; CHENG, A. M. K. Applying ant colony optimization to the partitioned scheduling problem for heterogeneous multiprocessors. **ACM SIGBED Review**, Association for Computing Machinery (ACM), v. 2, n. 2, p. 11–14, abr. 2005. Disponível em: <https://doi.org/10.1145/1121788.1121793>.

CHITRA, P.; VENKATESH, P.; RAJARAM, R. Comparison of evolutionary computation algorithms for solving bi-objective task scheduling problem on heterogeneous distributed computing systems. **Sadhana**, Springer Science and Business Media LLC, v. 36, n. 2, p. 167–180, abr. 2011. Disponível em: <https://doi.org/10.1007/s12046-011-0014-8>.

DE JONG, K. A. **Evolutionary Computation**: A unified approach. Cambridge, MA: The MIT Press, 2006. 268 p. ISBN 9780262529600.

DEB, K. **Multi-Objective Optimization Using Evolutionary Algorithms**. [S.l.]: Wiley, 2001. 518 p. ISBN 9780471873396.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, Institute of Electrical and Electronics Engineers (IEEE), v. 6, n. 2, p. 182–197, abr. 2002. Disponível em: <https://doi.org/10.1109/4235.996017>.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, JMLR. org, v. 7, p. 1–30, 2006.

DHINGRA, S.; GUPTA, S. B.; BISWAS, R. Genetic algorithm parameters optimization for bi-criteria multiprocessor task scheduling using design of experiments. **International Journal of Computer, Electrical, Automation, Control and Information Engineering**, World Academy of Science, Engineering and Technology, v. 8, n. 4, p. 628–634, 2014. Disponível em: <https://doi.org/10.5281/ZENODO.2666329>.

DOĞAN, A.; ÖZGÜNER, F. Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems. **The Computer Journal**, Oxford University Press (OUP), v. 48, n. 3, p. 300–314, mar. 2005. Disponível em: <https://doi.org/10.1093/comjnl/bxh086>.

DONG, F.; AKL, S. G. **Scheduling Algorithms for Grid Computing**: State of the art and open problems. Kingston, Ontario, 2006. (Relatório Técnico No. 2006-504).

DROZDOWSKI, M. **Scheduling for parallel processing**. London, UK: Springer-Verlag, 2009. 386 p. ISBN 9781848823099. Disponível em: <https://doi.org/10.1007/978-1-84882-310-5>.

EIBEN, A. E.; SMITH, J. E. **Introduction to Evolutionary Computing**. 2. ed. Heidelberg: Springer Berlin Heidelberg, 2015. 287 p. Disponível em: <https://doi.org/10.1007/978-3-662-44874-8>.

FRANÇA, T. P. **Estratégias bio-inspiradas aplicadas em problemas discretos com muitos objetivos**. Tese (Doutorado) — Universidade Federal de Uberlândia, Uberlândia, MG, jun. 2018. Disponível em: <https://doi.org/10.14393/ufu.di.2019.368>.

GABRIEL, P. H. R.; DELBEM, A. C. B. **Fundamentos de algoritmos evolutivos**. 33 p. Monografia (Apostila) — Univerisade de São Paulo, São Carlos, SP, 2008. Notas Didáticas do ICMC/USP, volume 75.

GABRIEL, P. H. R.; MELO, V. V. de; DELBEM, A. C. B. Algoritmos evolutivos e modelo HP para predição de estruturas de proteínas. **Revista Controle & Automação**, v. 23, n. 1, p. 25–37, jan. 2012. Disponível em: <https://doi.org/10.1590/S0103-17592012000100003>.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability**: A guide to the theory of np-completeness. EUA: W. H. Freeman and Company, 1979. 340 p. (A Series of Books in the Mathematical Sciences). ISBN 0716710455.

GERASOULIS, A.; YANG, T. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. **Journal of Parallel and Distributed Computing**, Elsevier BV, v. 16, n. 4, p. 276–291, dez. 1992. Disponível em: <https://doi.org/10.1016/0743-7315(92)90012-C>.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning**. Boston, MA, USA: Addison-Wesley Pub. Co., 1989. 412 p. ISBN 0201157675.

GOLUB, M.; KASAPOVIC, S. Scheduling multiprocessor tasks with genetic algorithms. In: **APPLIED INFORMATICS-PROCEEDINGS-**. [S.l.: s.n.], 2002. p. 273–278.

GUZEK, M.; PECERO, J. E.; DORRONSORO, B.; BOUVRY, P. Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems. **Applied Soft Computing**, Elsevier BV, v. 24, p. 432–446, nov. 2014. Disponível em: <https://doi.org/10.1016/j.asoc.2014.07.010>.

HOLLAND, J. H. Outline for a logical theory of adaptive systems. **Journal of the ACM**, Association for Computing Machinery (ACM), v. 9, n. 3, p. 297–314, jul. 1962. Disponível em: <https://doi.org/10.1145/321127.321128>.

_____. **Adaptation in natural and artificial systems**: An introductory analysis with applications to biology, control, and artificial intelligence. Oxford, England: Michigan Press, 1975. ISBN 0262581116.

HOU, E. S. H.; HONG, R.; ANSARI, N. Efficient multiprocessor scheduling based on genetic algorithms. In: **Proceedings of 16th Annual Conference of IEEE Industrial Electronics Society**. IEEE, 1994. Disponível em: <https://doi.org/10.1109/iecon.1990.149314>.

ISHIBUCHI, H.; AKEDO, N.; OHYANAGI, H.; NOJIMA, Y. Behavior of EMO algorithms on many-objective optimization problems with correlated objectives. In: **2011 IEEE Congress of Evolutionary Computation (CEC)**. IEEE, 2011. Disponível em: <https://doi.org/10.1109/cec.2011.5949788>.

JIN, S.; SCHIAVONE, G.; TURGUT, D. A performance study of multiprocessor task scheduling algorithms. **The Journal of Supercomputing**, Springer Science and Business Media LLC, v. 43, n. 1, p. 77–97, jun. 2007. Disponível em: <https://doi.org/10.1007/s11227-007-0139-z>.

KANG, C. C.; CHUANG, Y. J.; TUNG, K. C.; CHAO, C. C.; TANG, C. Y.; PENG, S. C.; WONG, D. S. H. A genetic algorithm-based boolean delay model of intracellular signal transduction in inflammation. **BMC bioinformatics**, Springer, v. 12, n. 1, p. 1–8, 2011. Disponível em: <https://doi.org/10.1186/1471-2105-12-S1-S17>.

KAUR, K.; CHHABRA, A.; SINGH, G. Modified genetic algorithm for task scheduling in homogeneous parallel system using heuristics. **International Journal of Soft Computing**, Medwell Publications, v. 5, n. 2, p. 42–51, feb 2010. Disponível em: <https://doi.org/10.3923/ijscomp.2010.42.51>.

KONAK, A.; COIT, D. W.; SMITH, A. E. Multi-objective optimization using genetic algorithms: A tutorial. **Reliability Engineering & System Safety**, Elsevier BV, v. 91, n. 9, p. 992–1007, set. 2006. Disponível em: <https://doi.org/10.1016/j.ress.2005.11.018>.

KUMAR, M.; SHARMA, S. C.; GOEL, A.; SINGH, S. P. A comprehensive survey for scheduling techniques in cloud computing. **Journal of Network and Computer Applications**, Elsevier BV, v. 143, p. 1–33, out. 2019. Disponível em: <https://doi.org/10.1016/j.jnca.2019.06.006>.

LACERDA, E. G. M. de; CARVALHO, A. C. P. de Leon Ferreira de; LUDERMIR, T. B. Um tutorial sobre algoritmos genéticos. **RITA**, v. 9, n. 3, p. 7–39, 2002.

LAFETÁ, T.; BUENO, M. L. P.; BRASIL, C. R. S.; OLIVEIRA, G. M. B. MEANDS: A many-objective evolutionary algorithm based on non-dominated decomposed sets applied to multicast routing. **Applied Soft Computing**, Elsevier BV, v. 62, p. 851–866, jan 2018. Disponível em: <https://doi.org/10.1016/j.asoc.2017.09.017>.

LI, J.-q.; PAN, Q.-k. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. **Information Sciences**, Elsevier BV, v. 316, p. 487–502, set. 2015. Disponível em: <https://doi.org/10.1016/j.ins.2014.10.009>.

LIEW, C. S.; ATKINSON, M. P.; GALEA, M.; ANG, T. F.; MARTIN, P.; HEMERT, J. I. V. Scientific workflows. **ACM Computing Surveys**, Association for Computing Machinery, v. 49, n. 4, p. 1–39, dez. 2017. Disponível em: <https://doi.org/10.1145/3012429>.

LUDÄSCHER, B.; BOWERS, S.; MCPHILLIPS, T. Scientific workflows. In: LIU, L.; ÖZSU, M. T. (Ed.). **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 2507–2511. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_1471>.

MARICHELVAM, M. K.; PRABAHARAN, T.; YANG, X. S. Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. **Applied Soft Computing**, Elsevier BV, v. 19, p. 93–101, jun. 2014. Disponível em: <https://doi.org/10.1016/j.asoc.2014.02.005>.

MENDES, F. G. S. Preference-guided evolutionary algorithms for optimization with many objectives. Universidade Federal de Minas Gerais, 2014. Disponível em: <https://doi.org/1843/BUOS-9RUHYK>.

NANDA, A.; DEGROOT, D.; STENGER, D. Scheduling directed task graphs on multiprocessors using simulated annealing. In: IEEE. **Proceedings of the 12th International Conference on Distributed Computing Systems**. IEE Compt. Soc. Press, 1992. p. 20–27. Disponível em: <https://doi.org/10.1109/icdcs.1992.235059>.

OMARA, F. A.; ARAFA, M. M. Genetic algorithms for task scheduling problem. **Journal of Parallel and Distributed Computing**, Elsevier BV, v. 70, n. 1, p. 13–22, jan. 2010. Disponível em: <https://doi.org/10.1016/j.jpdc.2009.09.009>.

PACHECO, M. A. C. **Algoritmos genéticos**: Princípios e aplicações. 28 p. Monografia (Apostila) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 1999.

PARETO front. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2023. Disponível em: <https://en.wikipedia.org/wiki/Pareto_front>. Acesso em: 2 fev. 2023.

PORTO, S. C. S.; RIBEIRO, C. C. A tabu search approach to task scheduling on heterogeneous processors under precedence constraints. **International Journal of High Speed Computing**, World Scientific Pub Co Pte Lt, v. 7, n. 01, p. 45–71, mar. 1995. Disponível em: <https://doi.org/10.1142/s012905339500004x>.

RAZALI, N. M.; GERAGHTY, J. Genetic algorithm performance with differentselection strategies in solving TSP. In: **Proceedings of the World Congress on Engineering**. London, UK: [s.n.], 2011. v. 2, p. 1–6.

ROBERT, Y. Task graph scheduling. In: PADUA, D. (Ed.). **Encyclopedia of Parallel Computing**. Boston, MA: Springer US, 2011. p. 2013–2025. Disponível em: <https://doi.org/10.1007/978-0-387-09766-4_42>.

SACHDEVA, S.; PANWAR, P. A review of multiprocessor directed acyclic graph (DAG) scheduling algorithms. **International Journal of Computer Science & Communication**, v. 6, n. 1, p. 67–72, mar. 2015. ISSN 0973-7391.

SADASHIV, N.; KUMAR, S. M. D. Cluster, grid and cloud computing: A detailed comparison. In: **6th International Conference on Computer Science & Education**. Singapura: IEEE, 2011. Disponível em: <https://doi.org/10.1109/iccse.2011.6028683>.

SANTOS, A. C.; DELBEM, A. C. B.; LONDON JR., J. B. A.; BRETAS, N. G. Node-depth encoding and multiobjective evolutionary algorithm applied to large-scale distribution system reconfiguration. **IEEE Transactions on Power Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 25, n. 3, p. 1254–1265, ago. 2010. Disponível em: <https://doi.org/10.1109/tpwrs.2010.2041475>.

SCHAFFER, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In: GREFENSTETTE, J. J. (Ed.). **Proceedings of the 1st International Conference on Genetic Algorithms**. New York: [s.n.], 1985. p. 93–100.

SECARA, I.-A. Challenges and considerations in developing and architecting large-scale distributed systems. **International Journal of Internet and Distributed Systems**, Scientific Research Publishing, Inc., v. 04, n. 01, p. 1–13, 2020. Disponível em: <https://doi.org/10.4236/ijids.2020.41001>.

SHEIKH, H. F.; AHMAD, I.; ARSHAD, S. A. Performance, energy, and temperature enabled task scheduling using evolutionary techniques. **Sustainable Computing: Informatics and Systems**, Elsevier BV, v. 22, p. 272–286, jun. 2019. Disponível em: <https://doi.org/10.1016/j.suscom.2017.10.002>.

SILVA, E. C. da. **Representações de Algoritmos Genéticos para o Problema de Escalonamento Estático de Tarefas em Multiprocessadores**. 231 p. Dissertação (Mestrado) — Universidade Federal de Uberlândia, Uberlândia, MG, jan. 2020. Disponível em: <https://doi.org/10.14393/ufu.di.2020.104>.

SILVA, E. C. da; GABRIEL, P. H. R. A comprehensive review of evolutionary algorithms for multiprocessor DAG scheduling. **Computation**, MDPI AG, v. 8, n. 2, p. 26, abr. 2020. Disponível em: <https://doi.org/10.3390/computation8020026>.

SINGH, J.; SINGH, G. Improved task scheduling on parallel system using genetic algorithm. **International Journal of Computer Applications**, Foundation of Computer Science, v. 39, n. 17, p. 17–22, feb 2012. Disponível em: <https://doi.org/10.5120/4912-7449>.

SIRHAN, N. N.; SERHAN, S. I. Multi-core processors: Concepts and implementations. **International Journal of Computer Science and Information Technology**, Academy and Industry Research Collaboration Center (AIRCC), v. 10, n. 1, p. 01–10, out. 2018. Disponível em: <https://doi.org/10.5121/ijcsit.2018.10101>.

SIVANANDAM, S. N.; VISALAKSHI, P.; BHUVANESWARI, A. Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia. **International Journal of Computer Science & Applications**, v. 4, n. 3, p. 95–106, 2007.

TOPCUOGLU, H.; HARIRI, S.; WU, M.-Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. **IEEE Transactions on Parallel and Distributed Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 13, n. 3, p. 260–274, mar 2002. Disponível em: <https://doi.org/10.1109/71.993206>.

VON ZUBEN, F. J. **Computação evolutiva**: Uma abordagem pragmática. 21 p. Monografia (Apostila) — Unicamp, Campinas, SP, 2000.

WALL, M. B. **A genetic algorithm for resource-constrained scheduling**. Tese (Doutorado) — Massachusetts Institute of Technology, 1996.

WANG, L.; SIEGEL, H. J.; ROYCHOWDHURY, V. P.; MACIEJEWSKI, A. A. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. **Journal of parallel and distributed computing**, Elsevier, v. 47, n. 1, p. 8–22, 1997. Disponível em: <https://doi.org/10.1006/jpdc.1997.1392>.

WANGSOM, P.; LAVANGNANANDA, K.; BOUVRY, P. Multi-objective scientific-workflow scheduling with data movement awareness in cloud. **IEEE Access**, Institute

of Electrical and Electronics Engineers (IEEE), v. 7, p. 177063–177081, 2019. Disponível em: <https://doi.org/10.1109/access.2019.2957998>.

XHAFA, F.; ABRAHAM, A. (Ed.). **Metaheuristics for Scheduling in Distributed Computing Environments**. Berlin: Springer Berlin Heidelberg, 2008. v. 146. 370 p. (Studies in Computational Intelligence, v. 146). Disponível em: <https://doi.org/10.1007/978-3-540-69277-5>.

_____. **Metaheuristics for Scheduling in Industrial and Manufacturing Applications**. Berlin: Springer Berlin Heidelberg, 2008. v. 128. (Studies in Computational Intelligence, v. 128). Disponível em: <https://doi.org/10.1007/978-3-540-78985-7>.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. **SPEA2: Improving the Strength Pareto Evolutionary Algorithm**. Zurich, 2001. (TIK Report, 103). Disponível em: <https://doi.org/10.3929/ethz-a-004284029>.

Apêndices

APÊNDICE **A**

Tabelas de Resultados

Este apêndice traz todos os resultados dos DAGs de 100 e 300 tarefas citados mas não apresentados no Capítulo 5.

A.1 Links do Projeto no GitHub

O projeto implementado para esta pesquisa está disponível no GitHub a partir do link https://github.com/johnataferreira/agscheduling. Segue abaixo alguns diretórios importantes presentes no projeto a partir do link raiz:

- DAGs originais (sem custo de comunicação):
  - /tree/master/src/br/ufu/scheduling/file/dag/without/cost
- DAGs usadas no trabalho (com custo de comunicação):
  - /tree/master/src/br/ufu/scheduling/file/dag/with/cost
- Resultados das execuções de todos os cenários:
  - /tree/master/src/br/ufu/scheduling/agmo/results

A.2 Resultados para o DAG de 100 Tarefas

Tabela 12 – Intervalo de valores obtidos para cada objetivo no DAG de 100 tarefas.

		$m = 2$		$m = 4$		$m = 8$		$m = 16$	
Obj.	**Inter.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**
M	90–100	**536,0**	585,8	**328,0**	382,1	**245,0**	297,1	**233,0**	269,4
	80–90	585,8	635,6	382,1	436,2	297,1	349,2	269,4	305,8
	70–80	635,6	685,4	436,2	490,3	349,2	401,3	305,8	342,2
	60–70	685,4	735,2	490,3	544,4	401,3	453,4	342,2	378,6
	50–60	735,2	785,0	544,4	598,5	453,4	505,5	378,6	415,0
	40–50	785,0	834,8	598,5	652,6	505,5	557,6	415,0	451,4
	30–40	834,8	884,6	652,6	706,7	557,6	609,7	451,4	487,8
	20–30	884,6	934,4	706,7	760,8	609,7	661,8	487,8	524,2
	10–20	934,4	984,2	760,8	814,9	661,8	713,9	524,2	560,6
	0–10	984,2	**1034,0**	814,9	**869,0**	713,9	**766,0**	560,6	**597,0**
L	90–100	**1,00000**	1,01320	**1,00037**	1,02351	**1,01126**	1,04301	**1,04451**	1,09690
	80–90	1,01320	1,02640	1,02351	1,04665	1,04301	1,07476	1,09690	1,14930
	70–80	1,02640	1,03961	1,04665	1,06979	1,07476	1,10651	1,14930	1,20170
	60–70	1,03961	1,05281	1,06979	1,09293	1,10651	1,13825	1,20170	1,25410
	50–60	1,05281	1,06601	1,09293	1,11608	1,13825	1,17000	1,25410	1,30650
	40–50	1,06601	1,07921	1,11608	1,13922	1,17000	1,20175	1,30650	1,35890
	30–40	1,07921	1,09241	1,13922	1,16236	1,20175	1,23350	1,35890	1,41129
	20–30	1,09241	1,10561	1,16236	1,18550	1,23350	1,26524	1,41129	1,46369
	10–20	1,10561	1,11882	1,18550	1,20864	1,26524	1,29699	1,46369	1,51609
	0–10	1,11882	**1,13202**	1,20864	**1,23178**	1,29699	**1,32874**	1,51609	**1,56849**
F	90–100	**26619,0**	29276,8	**16054,0**	18695,2	**12311,0**	14840,1	**11004,0**	13269,3
	80–90	29276,8	31934,6	18695,2	21336,4	14840,1	17369,2	13269,3	15534,6
	70–80	31934,6	34592,4	21336,4	23977,6	17369,2	19898,3	15534,6	17799,9
	60–70	34592,4	37250,2	23977,6	26618,8	19898,3	22427,4	17799,9	20065,2
	50–60	37250,2	39908,0	26618,8	29260,0	22427,4	24956,5	20065,2	22330,5
	40–50	39908,0	42565,8	29260,0	31901,2	24956,5	27485,6	22330,5	24595,8
	30–40	42565,8	45223,6	31901,2	34542,4	27485,6	30014,7	24595,8	26861,1
	20–30	45223,6	47881,4	34542,4	37183,6	30014,7	32543,8	26861,1	29126,4
	10–20	47881,4	50539,2	37183,6	39824,8	32543,8	35072,9	29126,4	31391,7
	0–10	50539,2	**53197,0**	39824,8	**42466,0**	35072,9	**37602,0**	31391,7	**33657,0**
C	90–100	**2138,0**	2626,5	**4020,0**	4642,5	**5048,0**	5706,4	**5700,0**	6386,6
	80–90	2626,5	3115,0	4642,5	5265,0	5706,4	6364,8	6386,6	7073,2
	70–80	3115,0	3603,5	5265,0	5887,5	6364,8	7023,2	7073,2	7759,8
	60–70	3603,5	4092,0	5887,5	6510,0	7023,2	7681,6	7759,8	8446,4
	50–60	4092,0	4580,5	6510,0	7132,5	7681,6	8340,0	8446,4	9133,0
	40–50	4580,5	5069,0	7132,5	7755,0	8340,0	8998,4	9133,0	9819,6
	30–40	5069,0	5557,5	7755,0	8377,5	8998,4	9656,8	9819,6	10506,2
	20–30	5557,5	6046,0	8377,5	9000,0	9656,8	10315,2	10506,2	11192,8
	10–20	6046,0	6534,5	9000,0	9622,5	10315,2	10973,6	11192,8	11879,4
	0–10	6534,5	**7023,0**	9622,5	**10245,0**	10973,6	**11632,0**	11879,4	**12566,0**
W	90–100	**2316,0**	2739,6	**1613,0**	1947,7	**1274,0**	1563,1	**1151,0**	1407,3
	80–90	2739,6	3163,2	1947,7	2282,4	1563,1	1852,2	1407,3	1663,6
	70–80	3163,2	3586,8	2282,4	2617,1	1852,2	2141,3	1663,6	1919,9
	60–70	3586,8	4010,4	2617,1	2951,8	2141,3	2430,4	1919,9	2176,2
	50–60	4010,4	4434,0	2951,8	3286,5	2430,4	2719,5	2176,2	2432,5
	40–50	4434,0	4857,6	3286,5	3621,2	2719,5	3008,6	2432,5	2688,8
	30–40	4857,6	5281,2	3621,2	3955,9	3008,6	3297,7	2688,8	2945,1
	20–30	5281,2	5704,8	3955,9	4290,6	3297,7	3586,8	2945,1	3201,4
	10–20	5704,8	6128,4	4290,6	4625,3	3586,8	3875,9	3201,4	3457,7
	0–10	6128,4	**6552,0**	4625,3	**4960,0**	3875,9	**4165,0**	3457,7	**3714,0**

Tabela 13 – Melhores objetivos encontrados por AGMO em um DAG de 100 tarefas na execução com 5 objetivos.

m	Obj.	AEMMT1		AEMMT2		AEMMD		NSGA-II	
		melhor	média	melhor	média	melhor	média	melhor	média
2	M	538,000	548,400±10,658	**536,000**	549,100±10,661	**536,000**	553,900±16,462	673,000	697,000±19,131
	L	**1,00000**	1,00017±0,00036	**1,00000**	1,00017±0,00036	**1,00000**	1,00000±0,00000	**1,00000**	1,00023±0,00030
	F	**26450,000**	27266,800±557,035	26621,000	27327,000±514,063	26619,000	27364,300±658,132	32341,000	33151,100±761,192
	C	2660,000	2804,000±103,133	2511,000	2730,200±173,380	**2138,000**	2478,300±180,987	2801,000	2882,000±59,920
	W	2236,000	2631,200±316,345	**2112,000**	2597,800±247,007	2461,000	2635,600±172,680	3026,000	3255,200±128,679
4	M	347,000	369,100±15,147	**327,000**	363,900±20,163	346,000	367,100±15,300	496,000	520,500±13,914
	L	1,00277	1,00552±0,00294	1,00260	1,00784±0,00410	**1,00037**	1,00297±0,00168	1,00356	1,00765±0,00284
	F	17073,000	18171,700±847,276	**15856,000**	18070,300±1040,753	17115,000	18153,200±984,708	23139,000	24285,800±801,683
	C	4145,000	4500,500±205,784	4287,000	4609,700±196,396	**4073,000**	4247,900±95,746	4492,000	4584,900±43,895
	W	1777,000	1938,900±101,492	**1648,000**	1830,300±105,872	1699,000	1912,400±117,651	2309,000	2503,800±116,347
8	M	259,000	276,200±11,073	264,000	282,900±15,560	**245,000**	276,100±24,592	382,000	403,200±16,383
	L	1,01572	1,02962±0,00902	1,02110	1,03544±0,00957	**1,01126**	1,01898±0,00531	1,02519	1,03768±0,00696
	F	12761,000	13495,300±524,141	12681,000	13823,900±693,614	**12311,000**	13433,000±952,106	17111,000	18404,900±795,102
	C	5233,000	5432,000±98,869	5310,000	5502,800±143,824	**5185,000**	5273,600±55,096	5390,000	5486,900±49,307
	W	**1340,000**	1500,300±168,754	1360,000	1514,300±109,819	1389,000	1473,800±73,365	1891,000	2042,500±84,467
16	M	236,000	244,900±8,171	**234,000**	244,400±7,777	235,000	247,600±8,396	287,000	320,800±14,242
	L	1,09900	1,11037±0,01353	1,07720	1,10552±0,01180	**1,04451**	1,06096±0,01190	1,08405	1,10128±0,01111
	F	11285,000	11681,000±368,278	**11199,000**	11820,500±364,761	11313,000	11874,200±336,115	13810,000	14798,600±510,866
	C	5798,000	5992,900±87,442	5831,000	5959,800±80,907	**5700,000**	5800,400±49,865	5932,000	5996,900±40,495
	W	1219,000	1283,500±42,327	1177,000	1279,700±52,275	**1165,000**	1278,200±62,105	1552,000	1649,700±64,102

A.3 Resultados para o DAG de 300 Tarefas

Tabela 14 – Intervalo de valores obtidos para cada objetivo no DAG de 300 tarefas.

		$m = 2$		$m = 4$		$m = 8$		$m = 16$	
Obj.	**Inter.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**	**Inf.**	**Sup.**
M	90–100	**1813,0**	2041,9	**1150,0**	1403,0	**886,0**	1087,8	**697,0**	876,5
	80–90	2041,9	2270,8	1403,0	1656,0	1087,8	1289,6	876,5	1056,0
	70–80	2270,8	2499,7	1656,0	1909,0	1289,6	1491,4	1056,0	1235,5
	60–70	2499,7	2728,6	1909,0	2162,0	1491,4	1693,2	1235,5	1415,0
	50–60	2728,6	2957,5	2162,0	2415,0	1693,2	1895,0	1415,0	1594,5
	40–50	2957,5	3186,4	2415,0	2668,0	1895,0	2096,8	1594,5	1774,0
	30–40	3186,4	3415,3	2668,0	2921,0	2096,8	2298,6	1774,0	1953,5
	20–30	3415,3	3644,2	2921,0	3174,0	2298,6	2500,4	1953,5	2133,0
	10–20	3644,2	3873,1	3174,0	3427,0	2500,4	2702,2	2133,0	2312,5
	0–10	3873,1	**4102,0**	3427,0	**3680,0**	2702,2	**2904,0**	2312,5	**2492,0**
L	90–100	**1,00000**	1,00407	**1,00029**	1,01500	**1,00416**	1,02136	**1,01505**	1,04103
	80–90	1,00407	1,00815	1,01500	1,02972	1,02136	1,03857	1,04103	1,06701
	70–80	1,00815	1,01222	1,02972	1,04444	1,03857	1,05577	1,06701	1,09299
	60–70	1,01222	1,01630	1,04444	1,05915	1,05577	1,07298	1,09299	1,11897
	50–60	1,01630	1,02037	1,05915	1,07387	1,07298	1,09018	1,11897	1,14495
	40–50	1,02037	1,02445	1,07387	1,08858	1,09018	1,10739	1,14495	1,17093
	30–40	1,02445	1,02852	1,08858	1,10330	1,10739	1,12460	1,17093	1,19691
	20–30	1,02852	1,03260	1,10330	1,11802	1,12460	1,14180	1,19691	1,22289
	10–20	1,03260	1,03667	1,11802	1,13273	1,14180	1,15901	1,22289	1,24887
	0–10	1,03667	**1,04075**	1,13273	**1,14745**	1,15901	**1,17621**	1,24887	**1,27485**
F	90–100	**273534,0**	305859,5	**183970,0**	216618,9	**127297,0**	157110,7	**88960,0**	113852,2
	80–90	305859,5	338185,0	216618,9	249267,8	157110,7	186924,4	113852,2	138744,4
	70–80	338185,0	370510,5	249267,8	281916,7	186924,4	216738,1	138744,4	163636,6
	60–70	370510,5	402836,0	281916,7	314565,6	216738,1	246551,8	163636,6	188528,8
	50–60	402836,0	435161,5	314565,6	347214,5	246551,8	276365,5	188528,8	213421,0
	40–50	435161,5	467487,0	347214,5	379863,4	276365,5	306179,2	213421,0	238313,2
	30–40	467487,0	499812,5	379863,4	412512,3	306179,2	335992,9	238313,2	263205,4
	20–30	499812,5	532138,0	412512,3	445161,2	335992,9	365806,6	263205,4	288097,6
	10–20	532138,0	564463,5	445161,2	477810,1	365806,6	395620,3	288097,6	312989,8
	0–10	564463,5	**596789,0**	477810,1	**510459,0**	395620,3	**425434,0**	312989,8	**337882,0**
C	90–100	**6816,0**	8149,0	**11877,0**	13614,9	**15217,0**	17013,1	**16665,0**	18657,1
	80–90	8149,0	9482,0	13614,9	15352,8	17013,1	18809,2	18657,1	20649,2
	70–80	9482,0	10815,0	15352,8	17090,7	18809,2	20605,3	20649,2	22641,3
	60–70	10815,0	12148,0	17090,7	18828,6	20605,3	22401,4	22641,3	24633,4
	50–60	12148,0	13481,0	18828,6	20566,5	22401,4	24197,5	24633,4	26625,5
	40–50	13481,0	14814,0	20566,5	22304,4	24197,5	25993,6	26625,5	28617,6
	30–40	14814,0	16147,0	22304,4	24042,3	25993,6	27789,7	28617,6	30609,7
	20–30	16147,0	17480,0	24042,3	25780,2	27789,7	29585,8	30609,7	32601,8
	10–20	17480,0	18813,0	25780,2	27518,1	29585,8	31381,9	32601,8	34593,9
	0–10	18813,0	**20146,0**	27518,1	**29256,0**	31381,9	**33178,0**	34593,9	**36586,0**
W	90–100	**44935,0**	52340,2	**30046,0**	37303,4	**26292,0**	31536,7	**16444,0**	20765,9
	80–90	52340,2	59745,4	37303,4	44560,8	31536,7	36781,4	20765,9	25087,8
	70–80	59745,4	67150,6	44560,8	51818,2	36781,4	42026,1	25087,8	29409,7
	60–70	67150,6	74555,8	51818,2	59075,6	42026,1	47270,8	29409,7	33731,6
	50–60	74555,8	81961,0	59075,6	66333,0	47270,8	52515,5	33731,6	38053,5
	40–50	81961,0	89366,2	66333,0	73590,4	52515,5	57760,2	38053,5	42375,4
	30–40	89366,2	96771,4	73590,4	80847,8	57760,2	63004,9	42375,4	46697,3
	20–30	96771,4	104176,6	80847,8	88105,2	63004,9	68249,6	46697,3	51019,2
	10–20	104176,6	111581,8	88105,2	95362,6	68249,6	73494,3	51019,2	55341,1
	0–10	111581,8	**118987,0**	95362,6	**102620,0**	73494,3	**78739,0**	55341,1	**59663,0**

Tabela 15 – Melhores objetivos encontrados por AGMO em um DAG de 300 tarefas na execução com 5 objetivos.

m	Obj.	AEMMT1		AEMMT2		AEMMD		NSGA-II	
		melhor	**média**	**melhor**	**média**	**melhor**	**média**	**melhor**	**média**
2	M	1794,000	1915,900±76,330	**1764,000**	1976,500±129,174	1871,000	1949,800±71,202	2742,000	2885,000±77,081
	L	**1,00000**	1,00017±0,00026	**1,00000**	1,00037±0,00037	**1,00000**	1,00000±0,00000	**1,00000**	1,00009±0,00010
	F	280963,000	293909,800±11430,037	**266908,000**	299828,100±18156,334	285131,000	296591,400±8657,937	388934,000	403096,200±12229,726
	C	7384,000	7825,700±299,702	**6966,000**	8103,900±614,706	6974,000	7405,200±285,413	8361,000	8545,200±120,056
	W	45349,000	52285,300±4398,554	47093,000	51778,300±3632,476	**44935,000**	50890,400±3730,156	59065,000	62330,600±2221,951
4	M	**1284,000**	1424,700±114,581	1293,000	1461,000±126,120	1337,000	1549,300±150,541	2348,000	2444,900±67,714
	L	1,00121	1,00273±0,00216	1,00077	1,00331±0,00219	**1,00045**	1,00081±0,00033	1,00104	1,00333±0,00106
	F	**184594,000**	206037,300±16084,232	187180,000	212833,100±17278,111	185664,000	225747,800±23483,989	313119,000	332627,900±10620,168
	C	12495,000	13047,100±436,186	12676,000	13308,400±457,447	**12293,000**	12685,000±270,146	13151,000	13500,000±168,474
	W	**34441,000**	40147,500±3474,745	37178,000	41089,500±2884,717	35491,000	39572,400±3090,898	47000,000	50471,600±1752,979
8	M	1006,000	1119,200±128,816	933,000	1045,200±65,635	**930,000**	1033,200±86,823	1694,000	1924,400±82,839
	L	1,00539	1,01595±0,00794	1,00549	1,01446±0,00628	**1,00416**	1,00744±0,00182	1,00939	1,01295±0,00233
	F	138845,000	154005,200±15368,278	135998,000	147835,600±6231,164	**130419,000**	144070,400±8659,670	236258,000	250477,700±9394,553
	C	15259,000	15985,500±349,457	15448,000	16150,700±489,296	**15217,000**	15362,300±120,724	15945,000	16146,200±106,881
	W	**25633,000**	28660,500±2457,531	27132,000	30251,600±2174,378	26636,000	28039,700±1800,425	35562,000	38238,900±857,531
16	M	740,000	790,600±45,841	**639,000**	779,600±92,602	697,000	795,000±59,731	1346,000	1396,300±36,548
	L	1,03546	1,05563±0,01636	1,02273	1,05974±0,01478	**1,01505**	1,02541±0,00583	1,03119	1,03947±0,00587
	F	94277,000	106532,000±6991,992	**89697,000**	104663,500±9332,640	93291,000	104389,500±7602,188	164468,000	177567,100±5987,785
	C	16942,000	17537,700±347,186	17259,000	17796,300±275,823	**16732,000**	16929,200±115,741	17457,000	17597,300±69,905
	W	19519,000	21580,300±1558,650	19755,000	21372,100±1281,970	**19332,000**	21091,600±1230,973	27140,000	28614,800±878,761