UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

FABIANY LAMBOIA

MODELO DE OTIMIZAÇÃO MULTIOBJETIVO BASEADO EM ALGORITMO SHUFFLED FROG LEAPING PARA TRANSPORTE DE PRODUTOS EM REDES DE DUTOS

TESE

CURITIBA

2015

FABIANY LAMBOIA

MODELO DE OTIMIZAÇÃO MULTIOBJETIVO BASEADO EM ALGORITMO SHUFFLED FROG LEAPING PARA TRANSPORTE DE PRODUTOS EM REDES DE DUTOS

Tese apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Doutor em Ciências" – Área de Concentração: Engenharia de Automação e Sistemas.

Orientadora: Prof. Dr. Lúcia Valéria Ramos de Arruda

CURITIBA

2015

Dados Internacionais de Catalogação na Publicação

L225m
2015

Lamboia, Fabiany
Modelo de otimização multiobjetivo baseado em algoritmo *shuffled frog leaping* para transporte de produtos em redes de dutos / Fabiany Lamboia.-- 2015.
156 f.: il.; 30 cm

Texto em português, com resumo em inglês.
Tese (Doutorado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Curitiba, 2015.
Bibliografia: f. 148-156.

1. Processo decisório por critério múltiplo. 2. Otimização matemática. 3. Programação heurística. 4. Programação evolucionária (Computação). 5. Modelos matemáticos. 6. Algoritmos genéticos. 7. Oleodutos de petróleo. 8. Agenda de execução (Administração). 9. Métodos de simulação. 10. Engenharia elétrica - Teses. I. Arruda, Lúcia Valéria Ramos de, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD: Ed. 22 -- 621.3

Biblioteca Central da UTFPR, Câmpus Curitiba

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

Título da Tese Nº. ____

Modelo de Otimização Multiobjetivo Baseado em Algoritmo Shueffled Frog Leaping para Transporte de Produtos em Redes de Dutos

por

Fabiany Lamboia

Orientadora: Profa. Dra. Lúcia Valéria Ramos de Arruda

Esta tese foi apresentada como requisito parcial à obtenção do grau de DOUTOR EM CIÊNCIAS – Área de Concentração: **Engenharia de Automação e Sistemas**, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR, às **9:00h** do dia **20 de novembro de 2015**. O trabalho foi aprovado pela Banca Examinadora, composta pelos doutores:

Profa. Dra. Lúcia Valeria Ramos de Arruda
(Presidente – UTFPR)

Prof. Dr. Júlio Cesar Nievola
(PUC-PR)

Profa. Dra. Franklina Maria Bragion de Toledo
(USP-SC)

Prof. Dr. Flavio Neves
(UTFPR)

Prof. Dr. Ricardo Luders
(UTFPR)

Visto da Coordenação:

Prof. Dr. Emilio Carlos Gomes Wille
(Coordenador do CPGEI)

AGRADECIMENTOS

Agradeço a Deus.

Agradeço a minha orientadora Valéria, que sempre esteve disposta ajudar e sempre esteve presente em todos os momentos solicitados.

Agradeço de todo meu coração ao meu marido e companheiro Daniel Brehm, por todo apoio e paciência nesses anos de doutorado e por cuidar tão bem da nossa filha Julia nos momentos ausentes.

Agradeço a minha família e a todos da família Brehm, pelo apoio, ajuda e compreensão.

Agradecer a todo pessoal do laboratório LASCA, pela ajuda e suporte sempre que necessário.

Apoio financeiro da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis – ANP –, da Financiadora de Estudos e Projetos – FINEP – e do Ministério da Ciência e Tecnologia – MCT – por meio do Programa de Recursos Humanos da ANP para o Setor Petróleo e Gás – PRH-ANP/MCT - PRH10-UTFPR.

RESUMO

LAMBOIA, Fabiany. MODELO DE OTIMIZAÇÃO MULTIOBJETIVO BASEADO EM ALGORITMO SHUFFLED FROG LEAPING PARA TRANSPORTE DE PRODUTOS EM REDES DE DUTOS. 156 f. Tese – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

A modelagem de sistemas envolvidos no gerenciamento das operações de uma rede de dutos é um problema de otimização que envolve complexas restrições operacionais. O transporte por meio de dutos mostra-se confiável e econômico, porém a elevada taxa de ocupação das redes e a quantidade de diferentes produtos levam a cenários operacionais complexos. Uma melhoria na eficiência desse tipo de transporte pode ser obtida através de uma melhor alocação dos recursos disponíveis, contudo além de ser este um problema combinatório de difícil solução, é também um problema de otimização multiobjetivo. Para resolver este tipo de problema, as técnicas baseadas em algoritmos evolucionários, são adequadas pois tratam simultaneamente com um conjunto de soluções possíveis que permite encontrar um conjunto de soluções ótimas de Pareto com a simples execução do algoritmo. Neste contexto, este trabalho tem como objetivo o desenvolvimento de modelos de otimização multiobjetivo aplicados ao escalonamento de operações em rede de dutos existente na indústria P & G, investigando técnicas baseadas em algoritmos evolucionários. Assim, usa-se uma abordagem que propõe o uso de um algoritmo evolucionário multiobjetivo inspirado a partir da evolução memética de um grupo de sapos que procuram por comida: o SFLA (*Shuffled Frog Leaping Algorithm*). Os resultados obtidos a partir das simulações realizadas serão comparados com um algoritmo muito conhecido na literatura, o algoritmo genético (AG). Além disso, como este trabalho utiliza um modelo de otimização multiobjetivo e nestes casos procura-se um conjunto de soluções Pareto-ótimas, uma nova abordagem é proposta para o algoritmo SFLA: o *Modified Shuffled Frog-leaping Pareto Approach* (MSFLPA). Esta abordagem combina o uso de uma pequena população e uma estratégia de arquivamento com um processo de reinicialização da população usando duas memórias auxiliares para armazenar soluções não-dominadas (Conjunto de Pareto) encontradas durante a evolução da população. Para validar o desempenho e a eficiência do algoritmo MSFLPA proposto, cinco funções Zitzler-Deb-Thiele são utilizadas para comparação com dois algoritmos genéticos multi-objetivos conhecidos da literatura: NSGA-II e SPEA2. Os experimentos numéricos indicam que MSFLPA produz soluções bem espalhadas (diversidade) e converge para a verdadeira fronteira de Pareto e verifica-se ser eficiente e competitivo para resolver problemas multiobjetivos. Após essa validação, o MSFLPA é usado para otimizar a alocação dos recursos e para resolver o problema de programação de uma rede de dutos e quando comparado com o NSGA-II e μAG, MSFLPA tem se mostrado uma nova alternativa eficaz para a solução de problemas multiobjetivos com mais de dois objetivos, como é o caso dos problemas de escalonamento de redes de dutos.

Palavras-chave: Metaheurísticas, Algoritmos Evolucionários, Otimização Multiobjetivo, Escalonamento, Rede de Dutos

ABSTRACT

LAMBOIA, Fabiany. MULTIOBJECTIVE OPTIMIZATION MODEL BASED ON SHUFFLED FROG LEAPING ALGORITHM FOR TRANSPORTING PRODUCTS IN PIPELINE NETWORKS. 156 f. Tese – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

The development of model to support pipeline network operation management is an optimization problem which involves complex operational constraints. The product transport through pipelines proves reliable and economical, especially for large volumes. However, the high occupancy rate of the distribution networks and the amount of different products should be transported under different operating conditions lead to complex operational scenarios. An efficiency improvement of products transport through pipeline networks can be obtained by a better allocation of available resources. However that is a hard solution combinatorial problem with multiobjective optimization characteristics. An alternative to efficient solve this type of problem is the use of metaheuristics such Multiobjective Evolutionary Algorithms (MOEA). MOEA uses a population of solutions in its search, and multiple Pareto-optimal solutions can, in principle, be found in one single run. This work aims to develop a model of multi-criterion optimization applied to scheduling operations in a real-world pipeline network in the oil industry. We use a metaheuristic optimization method inspired from the memetic evolution of a group of frogs when seeking for food: SFLA (Shuffled Frog Leaping Algorithm). The results obtained from the simulations are compared to an algorithm well known in the literature: genetic algorithm (GA). Moreover, this works then introduces a new approach of the original shuffled frog leaping algorithm to create a modified form of the algorithm: the Modified Shuffled frog-leaping Pareto Approach (MSFLPA). The main goal of MSFLPA is to represent and recover the entire Pareto front to a modeled problem, moreover an efficient and competitive algorithm to solve multi-objective scheduling problems with more than two conflicting objectives. This new approach combines the use of a small population and an archiving strategy with a procedure to restart the population using two auxiliary memories to store nondominated solutions (Pareto set) found during population evolution. To validate the performance and efficiency of the proposed MSFLPA in spread Pareto front, five Zitzler-Deb-Thiele functions are examined and compared against two well-known multi-objective genetic algorithms: NSGA-II and SPEA2. The numerical experiments indicate that MSFLPA yields spread solutions and converges to the true Pareto front and it is verified to be efficient and competitive for solving multi-objective problem. After this validation, the MSFLPA is used to optimize the allocation of the resources and to solve the scheduling problem of a real world pipeline network and if compared with NSGA-II and μGA, MSFLPA is verified to be a new effective alternative for solving of multi-objective problems with more than two objectives as it is the case of the pipeline scheduling problems.

Keywords: Metaheuristics, Evolutionary Algorithms, Multi-criterion Optimization, Scheduling, Pipeline Network

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Devido à busca crescente pelas áreas de planejamento e programação da produção por parte da indústria do petróleo, gás e biocombustíveis (P & G), há uma grande preocupação e necessidade de desenvolver ferramentas computacionais de auxílio à tomada de decisões, em especial aquelas que empregam técnicas de otimização. Além de visar a utilização dos recursos de uma forma mais eficiente, segura e lucrativa, há uma motivação adicional, já que esses problemas, muitas vezes, oferecem desafios devido à sua dimensão e complexidade (FELIZARI, 2009; FELIZARI et al., 2009).

A atividade de distribuição e transporte de derivados de petróleo faz parte da cadeia de suprimentos petrolífera e é uma atividade que deve coordenar objetivos de produção e compromissos contratados com clientes e distribuidores, de modo a otimizar o desempenho econômico do sistema. Segundo Rejowski e Pinto (2003), a indústria P & G , em especial, pode ter um ganho econômico considerável através de uma boa execução e prática de um planejamento e uma programação otimizada da produção e distribuição.

O transporte de produtos na indústria petrolífera, desde sua origem até o consumo, pode ser realizado através de diversos modais, tais como, transporte rodoviário, ferroviário, marítimo e dutoviário. Em particular, no território brasileiro, grande parte do transporte de óleos e derivados é realizado utilizando dutos (BOSCHETTO, 2011). O transporte de produtos derivados de petróleo através de dutos tem se mostrado um dos meios mais confiáveis e econômicos, principalmente para o transporte de grandes quantidades (SASIKUMAR et al., 1997).

Embora seja uma atividade fundamental, a programação relacionada à distribuição de produtos através de dutos ainda não dispõe de uma solução consolidada. O objetivo desta atividade é obter um bom desempenho sem comprometer o nível de serviço oferecido aos clientes quanto aos aspectos de qualidade, quantidade e prazo de entrega dos produtos (FELIZARI, 2009; BOSCHETTO, 2011).

Dentro deste contexto, este trabalho tem como objetivo geral propor um nova proposta

baseada em metaheurística para solucionar problemas de transferência de derivados de petróleo utilizando polidutos, em especial desenvolve-se modelos que usam técnicas evolucionárias na solução do sub-problema de alocação e sequenciamento de bateladas de produtos em redes de dutos. A seguir, este capítulo traz uma breve descrição da motivação e objetivos deste trabalho.

1.1 MOTIVAÇÃO

A modelagem de sistemas envolvidos no gerenciamento das operações de uma rede de dutos é um problema de otimização que envolve complexas restrições operacionais. Este tipo de transporte por meio de dutos mostra-se confiável e econômico, principalmente para grandes volumes. Porém, a elevada taxa de ocupação das redes de distribuição e a quantidade de diferentes produtos que devem ser transportados sob condições operacionais diferenciadas levam a cenários operacionais complexos em que a tomada de decisão por parte dos operadores é dificultada.

Uma rede de dutos é composta por refinarias, terminais portuários, áreas de armazenamento, mercados consumidores e vários dutos de tamanho e vazões variáveis que transportam diferentes derivados de petróleo. Dentro deste cenário, a busca por modelos de escalonamento que possam ser implementados na prática, considerando uma carga computacional aceitável torna-se um desafio para os pesquisadores e programadores operacionais de rede de dutos (NEVES-JR F. et al., 2007). Desta forma, uma abordagem de decomposição do problema deve ser utilizada para tornar viável a implementação de um sistema de apoio à tomada de decisão para operadores de redes de dutos. A subdivisão adotada nesta tese é baseada nos três elementos chave do escalonamento: a determinação e alocação dos recursos a serem utilizados (*assignment*), o sequenciamento de atividades (*sequencing*) e a temporização (*timing*) do uso dos recursos pelas atividades (REKLAITIS, 1992).

Dessa forma, o uso mais eficiente dessas redes pode ser alcançado através de uma melhor alocação dos recursos a partir de análises de planos de produção, consumo, estocagem e da análise das restrições da rede. Por outro lado, a alocação de recursos além de ser um problema combinatório de difícil solução é em geral um problema multiobjetivo. Para resolver este tipo de problema os métodos exatos são geralmente ineficientes, sendo portanto uma alternativa, o uso de metaheurísticas adaptadas para problemas multiobjetivos, como os algoritmos evolucionários multiobjetivos (YAMAMOTO, 2009).

Um problema multiobjetivo caracteriza-se pela existência de vários critérios conflitantes que não podem ser otimizados simultaneamente. Decorrente deste fato, admite-

se como solução não apenas uma única solução (ótima), mas um conjunto de soluções ótimas em que nenhuma solução do conjunto pode ser considerada superior às outras quando todos os critérios são analisados separadamente. Este conjunto é denominado conjunto ótimo de Pareto e as soluções que o compõe são conhecidas como soluções não-dominadas (COELLO, 2003).

As técnicas baseadas em metaheurísticas populacionais, em especial os algoritmos evolucionários, são adequados para resolver os problemas de otimização multiobjetivo porque tratam simultaneamente com um conjunto de soluções possíveis (população) que permite encontrar um conjunto de soluções ótimas de Pareto com a simples execução do algoritmo, em vez de ter de realizar uma série de operações separadas, como no caso das técnicas tradicionais de programação matemática. Além disso, os algoritmos evolucionários são menos suscetível aos problemas derivados do formato ou continuidade da fronteira de Pareto, ao passo que estas duas questões são uma real preocupação para as técnicas de programação matemática (COELLO, 1999).

Durante os últimos anos, técnicas metaheurísticas baseadas em algoritmos evolucionários foram estudadas e usadas para resolver problemas de escalonamento de sistemas de produção complexos. Elas têm se mostrado eficazes na busca de soluções satisfatórias em baixo tempo computacional (HERTZ; WIDMER, 2003; DREO et al., 2006). No entanto, no que diz respeito ao escalonamento de redes de dutos com características multiobjetivos, poucos trabalhos utilizando metaheurísticas foram publicados. Por exemplo, um modelo de otimização multiobjetivo baseado em Algoritmo Genético com restrições é proposto em (CRUZ et al., 2003) para o problema da distribuição de produtos petrolíferos através de uma rede de dutos (duas refinarias, dois depósitos intermediários, três clientes finais e 10 conexões). Os dois objetivos a serem alcançados são a entrega de produtos exigidos em um mínimo de tempo e a minimização da interface entre produtos diferentes. O mesmo problema multiobjetivo é modelado e resolvido por meio de algoritmo multiobjetivo genético com ranqueamento baseado em nichos (RNGA) em (WESTPHAL; ARRUDA, 2007).

O trabalho apresentado em (ARRUDA et al., 2010) desenvolve um modelo multiobjetivo baseado em algoritmo genético (MOGA) aplicado a uma rede de dutos (três refinarias, um porto, cinco clientes finais e 15 dutos) apresentada em (NEVES-JR F. et al., 2007). No entanto, o modelo MOGA proposto apresentou uma carga computacional que limita seu uso a exemplos com um pequeno número de bateladas. Recentemente, o trabalho apresentado em (RIBAS et al., 2013) apresenta um modelo multiobjetivo com base em Algoritmo micro Genético (μAG) (COELLO, 2003) para o escalonamento de curto prazo para rede de dutos descrita em (BOSCHETTO et al., 2010) que contém quatro refinarias, dois

portos, cinco centros de distribuição, dois clientes finais e trinta dutos bidirecionais. O modelo μAG foi comparado com o modelo MOGA (ARRUDA et al., 2010) para vários cenários com uma quantidade diferente de produtos, volumes e horizonte de programação. Embora o modelo μAG tenha apresentado melhor desempenho em todos os cenários testados, a distribuição e diversidade de soluções na fronteira de Pareto não foram asseguradas.

1.2 OBJETIVOS

Este trabalho tem como proposta principal o desenvolvimento de métodos de otimização multiobjetivo aplicado ao escalonamento de operações em rede de dutos existente na indústria P & G, investigando técnicas baseadas em algoritmos evolucionários multiobjetivos que auxiliem na tomada de decisões neste contexto. A rede de dutos em estudo é a rede de escuros, que é parte integrante do sistema dutoviário do estado de São Paulo, que é operado pela Transpetro.

Como citado anteriormente, o problema de escalonamento de operações de distribuição de produtos derivados do petróleo é um problema complexo, que é resolvido por um grupo de especialistas (YAMAMOTO, 2009; BOSCHETTO, 2011). Desta forma, este trabalho tem como objetivo desenvolver um método computacional que forneça soluções operacionais otimizadas atendendo aos requisitos do problema.

Um problema de escalonamento de distribuição de produtos em uma rede de dutos deve satisfazer às restrições operacionais, restrições de distribuição, demanda de produtos, restrições de sequenciamento e restrições operacionais dos dutos. O transporte desses produtos é motivado pelo cumprimento por um ou mais dos seguintes fatores: i) demanda de mercado; ii) demanda de estoque; iii) demanda de fornecimento de uma produção mínima, ou ainda por qualquer outro motivo operacional.

Para desenvolver o modelo proposto para a rede de escuros estudada, algumas das restrições citadas acima foram consideradas como objetivos de otimização, configurando assim um problema multiobjetivo. Para a solução de problemas de otimização multiobjetivo, as técnicas baseadas em metaheurísticas populacionais, em especial os algoritmos evolucionários, tem se mostrado eficientes, pois estes algoritmos tratam simultaneamente com um conjunto de soluções possíveis (população) que permite encontrar um conjunto de soluções ótimas de Pareto com a simples execução do algoritmo (COELLO et al., 2007).

Desta forma, neste trabalho usa-se uma abordagem que propõe o uso de métodos baseados em algoritmos evolucionários multiobjetivo (AEMOs) para otimizar as operações

de uma rede de dutos que envolve um conjunto de restrições operacionais que influenciam significativamente seu desempenho.

Dentre os diversos métodos AEMOs existentes, um algoritmo que vem sendo utilizado como um método AEMO combinando suas boas características de convergência com a otimização multiobjetivo, é o algoritmo evolucionário SFLA (*Shuffled Frog Leaping Algorithm*). Este algoritmo foi escolhido neste trabalho como uma alternativa eficaz para a solução dos modelos desenvolvidos.

O algoritmo SFLA é uma metaheurística inspirada a partir da evolução memética para otimização combinatória em primeiro lugar projetado para resolver um problema de um sistema de distribuição de água (EUSUFF; LANSEY, 2003). Em essência, ele combina os benefícios dos algoritmos meméticos de base genética e o comportamento social do algoritmo PSO (*particle swarm optimization*). A característica mais proeminente de SFLA é a sua rápida convergência (ELBELTAGI et al., 2005). Além disso, tem as vantagens de concepção simples, poucos parâmetros, bom desempenho e programação fácil. O algoritmo foi testado em diversas funções de *benchmark* mostrando sua eficiência para muitos problemas de otimização global (EUSUFF et al., 2006). Além disso, o SFLA têm sido usado para resolver problemas de otimização discreto, assim como problemas de otimização contínua (LAKSHMI; RAO, 2010).

Dessa forma, considerando-se três aspectos dos AEMOs: i) a utilização de pequenas populações reduz a carga computacional, mas não é suficiente para assegurar uma boa distribuição na fronteira de Pareto; ii) uma estratégia de arquivamento para salvar as soluções não-dominadas associada a um processo de reinicialização pode impedir uma convergência prematura para regiões específicas da froneira de Pareto, e considerando também que iii) o SFLA é uma boa alternativa para resolver problemas de programação de grande porte com objetivos conflitantes como sugerido pela literatura; propõe-se neste trabalho uma versão modificada do SFLA para resolver a otimização multiobjetivo de uma rede de dutos do mundo real. O algoritmo proposto tem uma população pequena e utiliza uma estratégia de arquivamento combinadas com as vantagens de uma rápida convergência do SFLA.

Além disso, propõe-se um novo procedimento para reiniciar a população usando duas memórias auxiliares (uma de curto e outra de longo prazo) para armazenar soluções não-dominadas encontradas durante a evolução da população. O objetivo é representar e recuperar boa parte da fronteira de Pareto para o problema modelado. Esta nova abordagem é chamada de **Modified Shuffled Frog-Leaping Pareto Approach** (MSFLPA).

A eficácia do MSFLPA para construir/recobrir a fronteira de Pareto é analisada neste trabalho, resolvendo cinco problemas multiobjetivo bem conhecidos da literatura. Esta eficácia

é avaliada por duas métricas e em estudo comparativo com dois algoritmos de evolução multiobjetivo da literatura, NSGA-II (DEB et al., 2002a) e SPEA2 (ZITZLER et al., 2001). Além disso, experimentos numéricos e um estudo de caso para calcular a programação de curto prazo de um cenário da rede de dutos também são realizados para verificar o desempenho do método proposto (MSFLPA). Seu desempenho também é comparado com NSGA-II por (DEB et al., 2002a) e o Algoritmo micro Genético (μAG) por (RIBAS et al., 2013) que foram adaptados para modelar a rede de dutos estudada.

Para analisar os resultados obtidos, diversas simulações foram realizadas com os modelos multiobjetivos desenvolvidos utilizando o SFLA original, a nova abordagem proposta o MSFLPA, o Algoritmo micro Genético (μAG) e o NSGA-II. Os resultados obtidos serão apresentados da seguinte forma:

1. Um estudo de caso: uma comparação utilizando um cenário teste que foi resolvido pelo SFLA original e pelo Algoritmo micro Genético (μAG) (com o objetivo de mostrar a eficácia da busca local e a rápida convergência do SFLA) e dois cenários testes com 4 e 8 produtos que foi resolvido pela nova abordagem: o MSFLPA (com o objetivo de mostrar os bons resultados preliminares da abordagem proposta);
2. Uma comparação utilizando um cenário teste comparando o MSFLPA, NSGA-II e o μAG (com o objetivo de mostrar o desempenho satisfatório do MSFLPA quando comparado a dois outros algoritmos multiobjetivos da literatura);
3. Resultados utilizando dois cenários reais de rede de escuros resolvidos com a nova abordagem MSFLPA (para validar a eficiência do MSFLPA quando aplicado a cenários reais de uma rede de dutos).

Portanto, o principal objetivo deste trabalho é apresentar o algoritmo MSFLPA como um algoritmo eficiente e competitivo para resolver problemas de escalonamento multiobjetivo com mais de dois objetivos conflitantes. Especificamente, mostra-se que o MSFLPA é uma alternativa boa e eficaz para modelar a alocação de recursos e de transporte do produto através de redes de dutos.

1.3 CONTRIBUIÇÕES/PUBLICAÇÕES

As principais contribuições alcançadas ao longo deste trabalho são:

- A caracterização de um problema real de escalonamento relacionado a programação de operação e transporte de derivados de petróleo em uma rede de dutos;

- A utilização de uma estratégia de decomposição em subproblemas (alocação de recursos, sequenciamento e temporização) para tornar viável a implementação de sistema, com o foco principal na alocação de recursos que pode tornar o uso mais eficiente das redes de dutos através da análise de um conjunto de informações referentes à rede;

- A modelagem e representação da rede de escuros que é parte integrante do sistema dutoviário do estado de São Paulo que é operado pela Transpetro;

- A proposta de uma abordagem de otimização multiobjetivo que propõe o uso de modelos baseados em algoritmos evolucionários multiobjetivo (AEMOs) para otimizar as operações da rede de escuros, que envolve um conjunto de restrições operacionais que influenciam no seu desempenho;

- Desenvolvimento de um novo algoritmo, chamado de **Modified Shuffled Frog-Leaping Pareto Approach** (MSFLPA), que combina o uso de uma pequena população e uma estratégia de arquivamento com um processo de reinicialização da população usando duas memórias auxiliares para armazenar soluções não-dominadas (Conjunto de Pareto) encontradas durante a evolução da população;

- Uma comprovação da eficácia do MSFLPA para construir/recobrir a fronteira de Pareto resolvendo cinco funções de teste (*benchmarks*) bem conhecidas na área de problemas multiobjetivos propostas por Zitzler-Deb-Thiele (ZDT) (DEB, 1999; ZITZLER et al., 2000);

- Um estudo comparativo com dois algoritmos bem conhecidos de evolução multiobjetivo, NSGA-II (DEB et al., 2002a) e SPEA2 (ZITZLER et al., 2001), a partir de duas métricas importantes que foram definidas em (DEB et al., 2002a) para avaliar um conjunto de soluções obtidas através de um algoritmo de otimização multiobjetivo;

- Uma análise de resultados a partir de diversas simulações realizadas com os modelos multiobjetivos desenvolvidos utilizando o SFLA original, a nova abordagem proposta o MSFLPA, o Algoritmo micro Genético (μAG) e o NSGA-II para solução de escalonamento de curto prazo de alguns cenários da rede de dutos estudada nesta tese.

Como resultado das contribuições citadas, os seguintes trabalhos relacionados à tese, encontram-se já publicados:

- LAMBOIA, F., ARRUDA, L.V.R. e NEVES-JR, F. N. Otimização multiobjetivo do transporte de produtos em redes dutoviárias através do algoritmo Shuffled Frog-leaping

modificado. In: SOBRAPO. XLIV Simposio Brasileiro de Pesquisa Operacional, pp. 2658-2669. Rio de Janeiro, Brasil 2012.

- LAMBOIA, F., ARRUDA, L.V.R. e NEVES-JR, F. A modified shuffled frog-leaping algorithm to model products transport in pipeline networks. In: EngOpt 2014 - IV International Conference on Engineering Optimization, pp. 885-890. Lisboa, Portugal 2014.

- LAMBOIA, F., ARRUDA, L.V.R. e NEVES-JR, F. Modified Shuffled Frog Leaping Algorithm for Improved Pareto-Set Computation: Application to Product Transport in Pipeline Networks. Journal of Control, Automation and Electrical Systems, 2015.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em nove capítulos. O Capítulo 2 descreve os conceitos básicos do problema de escalonamento na indústria P & G, é apresentada uma revisão dos trabalhos referentes à aplicação de metaheurísticas na resolução de problemas de escalonamento de operações em complexos petrolíferos, principalmente relacionados às atividades de produção em refinarias, distribuição e transporte de produtos em sistemas dutoviários.

O Capítulo 3 é dedicado a apresentar a fundamentação teórica de todos os conceitos utilizados neste trabalho. Dentre esses, uma descrição dos algoritmos evolucionários utilizados para a modelagem do problema. Neste capítulo também são descritos os conceitos de problemas de otimização multiobjetivo, métodos e técnicas de solução para problemas multiobjetivos. Por fim, os conceitos de otimalidade de pareto e otimização multiobjetivo evolucionária são apresentados.

O Capítulo 4 apresenta a nova abordagem proposta que é chamada de *Modified Shuffled Frog-leaping Pareto Approach* (MSFLPA) e descreve todos os detalhes desta nova abordagem, assim como os testes numéricos para validação do algoritmo.

O Capítulo 5 expõe a descrição do problema de escalonamento de rede de dutos. Em seguida, é apresentado uma modelagem para o problema, a otimização multiobjetivo e a metodologia desenvolvida.

O Capítulo 6 preliminarmente apresenta os resultados obtidos e mostrados em Lamboia et al. (2012) e Lamboia et al. (2014), executando simulações com o modelo de otimização multiobjetivo com configurações básicas e comparando os resultados entre o algoritmo SFLA e o μAG. Em seguida, os resultados obtidos com algumas alterações no modelo, também

comparando os resultados entre o algoritmo SFLA e o μAG. Por fim, apresenta os resultados obtidos com a nova abordagem proposta neste trabalho, o MSFLPA.

O Capítulo 7 apresenta uma comparação utilizando um cenário teste entre a nova abordagem MSFLPA e os algoritmos NSGA-II e μAG.

No Capítulo 8 dois cenários reais são utilizados para avaliar a eficiência do MSFLPA em resolver problemas multiobjetivos para escalonamento de curto prazo em rede de dutos.

O Capítulo 9 reporta as principais conclusões do trabalho, as contribuições para o estudo do problema de programação de redes de dutos e os principais aspectos a serem explorados em desenvolvimentos futuros.

2 ESTADO DA ARTE

Este capítulo primeiramente descreve os conceitos básicos do problema de escalonamento na indústria P & G e a seguir, apresenta uma revisão dos trabalhos referentes à aplicação de metaheurísticas na resolução de problemas de escalonamento de operações em complexos petrolíferos, principalmente relacionados às atividades de produção em refinarias, distribuição e transporte de produtos em sistemas dutoviários.

2.1 ESCALONAMENTO (*SCHEDULING*)

No contexto da indústria P & G, tanto as atividades de escalonamento (*scheduling*) das operações de produção quanto o planejamento (*planning*) de atividades relacionam-se a alocação de recursos e equipamentos limitados, num determinado período de tempo, para executar tarefas de processamento de um ou mais produtos (PEKNY; ZENTNER, 1993). Enquanto o planejamento tem uma visão a longo prazo, gerando decisões sobre escalas longas de tempo, medidas em meses, trimestres ou anos, o escalonamento objetiva a previsão de tempo capturando a dinâmica do sistema, focando nas decisões em escalas curtas de tempo, como dias ou semanas.

O escalonamento refere-se aos procedimentos e processos de decisão em que os recursos são alocados para atividades de modo a alcançar os resultados desejados em tempo hábil e/ou de baixo custo. O escalonamento é necessário sempre que há uma competição entre as atividades para um capital limitado e recursos operacionais que estão disponíveis ao longo de um período de tempo finito (REKLAITIS, 1995).

Em problemas de escalonamento, três componentes importantes estão presentes: a determinação e alocação dos recursos a serem utilizados (*assignment*), o sequenciamento de atividades (*sequencing*) e a temporização (*timing*) do uso dos recursos pelas atividades (REKLAITIS, 1992).

A alocação envolve a seleção de um grupo apropriado de recursos para uma dada

atividade. O sequenciamento se refere à ordenação da execução das atividades alocadas aos recursos, enquanto a temporização envolve a determinação do início e final específico para cada uma das atividades programadas (FELIZARI, 2009).

De forma geral, um problema de escalonamento possui os seguintes elementos (REKLAITIS, 1992):

1. Conjunto de equipamentos;
2. Conjunto de recursos humanos, materiais secundários e de utilidades;
3. Conjunto de receitas e custos associados a produtos e processos;
4. Demanda de matérias-primas ou de produtos finais;
5. Estratégia operacional;
6. Otimização de um ou mais objetivos.

Em problemas de escalonamento, determinadas escolhas para a realização de uma tarefa podem gerar consequências importantes ao longo do horizonte de tempo. Alem disso, deve ser levado em consideração o grande número de atividades diferentes que precisam ser programadas ao mesmo tempo. Para (LEE; DAGLI, 1997) outros fatores contribuem para a dificuldade dos problemas de escalonamento, tais como: as ações tomadas pelo programador de operações geralmente dependem de outras decisões, externas à sua atribuição (BODINGTON; SHOBRYS, 1995); a existência de eventos aleatórios; o processo possuir múltiplos objetivos, muitas vezes conflitantes, a serem otimizados simultaneamente.

A partir dessas dificuldades, conclui-se que problemas de escalonamento são inerentemente difíceis, sendo comum a presença de incertezas associadas à solução de problemas desta complexidade. Apesar das dificuldades existentes, o escalonamento é necessário sempre que for verificada uma competição entre tarefas de um processo, distintas ou não, por recursos limitados que se encontram disponíveis dentro de um período de tempo (FELIZARI, 2009).

2.1.1 MODELOS DE ESCALONAMENTO

Um modelo é uma representação matemática simplificada de uma realidade, devendo equilibrar a necessidade de contemplar suas principais propriedades com a viabilidade de encontrar soluções adequadas (GOLDBARG; LUNA, 2000).

Algumas características são destacadas em (FELIZARI, 2009) para elaboração de modelos de escalonamento:

- **Representação do Tempo**

  Segundo Pinto e Grossmann (1995) a representação do tempo é um dos maiores desafios na formulação de modelos de escalonamento. Basicamente, a representação do tempo pode assumir duas importantes formas:

  ✓ *Representação discreta do tempo*: os intervalos de tempo possuem duração fixa, a representação do domínio de tempo é chamada discreta (LEE et al., 1996; JOLY, 1999; MORO, 1999; STEBEL et al., 2002; MAGALHãES, 2004; MAGATãO et al., 2004);

  ✓ *Representação contínua do tempo*: nesta representação, o horizonte de planejamento é dividido em intervalos de tempo de duração variável. A tendência dos trabalhos atuais aponta para o desenvolvimento de formulações em tempo contínuo, de forma a explorar características particulares de cada problema (PINTO; GROSSMANN, 1995; MOCKUS; REKLAITIS, 1997; IERAPETRITOU; FLOUDAS, 1998; MORO, 1999; MÉNDEZ; CERDÁ, 2002; MAGATãO, 2005; STEBEL, 2006; CAFARO; CERDÁ, 2008; FELIZARI, 2009; BOSCHETTO, 2011).

  A forma de representação do tempo influencia decisivamente no número de variáveis utilizadas. Em uma representação discreta o número de variáveis tende a ser bem maior. Com representações contínuas de tempo, o número de variáveis pode ser significativamente menor, ainda que a formulação do problema se torne mais complexa.

- **Incerteza nos Parâmetros**

  Dependendo da inclusão ou não de incertezas nos parâmetros do problema, como por exemplo: datas de entrega, taxas de produção/demanda e custos; os modelos de escalonamento podem ser caracterizados como estocásticos ou determinísticos (ALLE, 2003).

- **Modelagem Matemática**

  O desenvolvimento de modelos matemáticos baseia-se num conjunto de processos estruturados, algumas das principais etapas são destacadas por Linares et al. (2001): Identificação do problema; Formulação matemática; Resolução; Verificação e validação; Interpretação e análise dos resultados; e Implantação e manutenção.

- **Métodos de Solução**

  O problema de escalonamento pode ser modelado como um problema de otimização combinatória, uma vez que visa a utilização ótima de um conjunto de recursos em função de sua disponibilidade, com o objetivo de minimização ou de maximização de critérios que podem ser econômicos ou operacionais (FELIZARI, 2009).

  Existem várias técnicas utilizadas para resolver problemas de otimização combinatória, dentre elas destacam-se:

  ✓ Programação matemática (inteira mista) linear ou não linear;

  ✓ Programação lógica por restrições;

  ✓ Métodos heurísticos;

  ✓ Metaheurística (algoritmos genéticos, *simulated annealing*, busca tabu, etc);

  ✓ Métodos híbridos.

  Programação matemática e metaheurísticas estão entre as mais exploradas na resolução de problemas de escalonamento. Enquanto programação matemática visa alcançar uma solução ótima, metaheurísticas preocupam-se com soluções que satisfaçam critérios de viabilidade e qualidade em tempo computacional aceitável (ALLE, 2003).

Uma revisão de trabalhos que usam técnicas de programação matemática em otimização linear e não-linear na resolução de problemas de escalonamento da indústria P & G pode ser encontrada nos trabalhos de Magalhães (2004), Stebel (2006), Rejowski e Pinto (2003) e Felizari (2009).

Como neste trabalho a proposta é apresentar uma metaheurística para a resolução do problema, na próxima seção, a revisão bibliográfica é restrita a uma descrição detalhada sobre metaheurísticas.

2.2 METAHEURÍSTICAS

O termo metaheurística foi introduzido pela primeira vez por Glover (1986), este termo deriva da composição de duas palavras gregas: heurística deriva do verbo *heuriskein* que significa “encontrar”, enquanto o sufixo meta significa “além de, em um nível superior” (BLUM; ROLI, 2003).

Heurística significa um método que, baseado na experiência, provavelmente encontre uma solução razoável para um problema, mas não garante a geração de uma solução matematicamente ótima (SILVER, 2002).

Uma metaheurística pode ser formalmente definida, segundo Osman e Laporte (1996), como um processo de geração iterativo que guia uma heurística pela combinação inteligente de diferentes conceitos para exploração e aproveitamento do espaço de busca, em que estratégias de aprendizagem são utilizadas para estruturar informações a fim de encontrar de forma eficiente soluções ótimas ou próximas do ótimo.

Conforme Silver (2002), uma metaheurística está particularmente focada em não ficar restrita a um ótimo local (para os problemas que têm múltiplos ótimos locais) e/ou de uma maneira prudente reduzir seu espaço de busca.

Para Hertz e Widmer (2003) as metaheurísticas são técnicas gerais de otimização combinatória, que não são dedicadas à solução de um problema em particular, mas são desenvolvidas com o objetivo de ser suficientemente flexíveis para lidar com diversos problemas combinatórios. Essas técnicas gerais rapidamente demonstraram sua utilidade e eficiência na resolução de problemas difíceis.

As metaheurísticas mais conhecidas são: Colônias de Formigas (*Ant Colony Optimization* - ACO), Algoritmos Evolucionários (AE) incluindo Algoritmos Genéticos (AG) e outros, Busca Local Iterativa (*Iterated Local Search* - ILS), *Simulated Annealing* (SA) e Busca Tabu (BT).

Segundo Blum e Roli (2003) as propriedades fundamentais que caracterizam as metaheurísticas são:

- Metaheurísticas são estratégias que guiam o processo de busca;
- O objetivo é explorar eficientemente o espaço de busca para encontrar soluções ótimas ou próximas do ótimo;
- As técnicas que constituem as metaheurísticas podem variar de simples procedimentos de busca local até complexos processos de aprendizagem;
- Algoritmos de metaheurísticas são aproximados e geralmente não-determinísticos;
- Os algoritmos podem incorporar mecanismos para evitar que fiquem presos em áreas restritas do espaço de busca;
- Os conceitos básicos das metaheurísticas permitem um nível abstrato de descrição;

- As metaheurísticas não são desenvolvidas para um problema específico;
- As metaheurísticas podem fazer uso do conhecimento de domínio específico na forma de heurísticas que são controladas por estratégias de alto nível;
- Algumas metaheurísticas mais avançadas usam a experiência da busca (incorporada em algum tipo de memória) para guiar a busca.

Sucintamente, pode-se dizer que as metaheurísticas são estratégias de alto nível para explorar espaços de busca através de diferentes métodos.

Para Blum e Roli (2003) existem diversas filosofias diferentes, visíveis nas metaheurísticas existentes. Algumas delas podem ser vistas como extensões "inteligentes´´ de algoritmos de busca local. O objetivo deste tipo de metaheurística é escapar de mínimos locais para prosseguir a exploração do espaço de busca e tentar encontrar outros mínimos locais melhores. Por exemplo, é o caso da Busca Tabu, Busca Local Iterativa, *Variable Neighborhood Search*, GRASP e *Simulated Annealing*. Estas metaheurísticas (também chamadas de métodos de trajetória) trabalham em uma ou várias estruturas de vizinhança imposta sobre as soluções do espaço de busca.

Em algoritmos como Otimização por Colônia de Formigas (ACO) e Computação Evolucionária (CE) encontra-se uma filosofia diferente. Estes incorporam um componente de aprendizagem, de modo que, implicitamente ou explicitamente tentam aprender correlações entre as variáveis de decisão para identificar áreas de alta qualidade no espaço de busca. De certo modo, este tipo de metaheurística realiza uma amostragem tendenciosa do espaço de busca. Por exemplo, em Computação Evolucionária isto é alcançado pela recombinação de soluções, enquanto que em Otimização por Colônia de Formigas, isto é feito pela amostragem do espaço de busca em cada iteração de acordo com uma distribuição de probabilidade.

As metaheurísticas são flexíveis e podem solucionar diversas classes de problemas. Esta flexibilidade deve-se principalmente ao fato de que cada metaheurística tem um ou mais parâmetros ajustáveis. Mas para qualquer aplicação, estes parâmetros requerem uma "calibração cuidadosa" sobre um conjunto de instâncias do problema, bem como testes em um conjunto independente de instâncias (SILVER, 2002).

Em metaheurísticas existem dois conceitos muito importantes que são a intensificação e diversificação. Estas são duas diretrizes que determinam amplamente o comportamento de uma metaheurística. Elas são de algum modo contrárias, mas também complementares uma da outra. Considera-se de grande importância que um equilíbrio dinâmico seja dado entre

diversificação e intensificação. O termo diversificação geralmente se refere à exploração do espaço de busca, enquanto que o termo intensificação se refere a exploração da experiência acumulado de busca (BLUM; ROLI, 2003).

Existem diferentes formas de classificar e descrever algoritmos de metaheurística. Segundo Blum e Roli (2003) dependendo das características selecionadas para diferenciá-los, diversas classificações são possíveis, cada uma delas sendo o resultado de um ponto de vista específico. As classificações mais importantes são:

- *Algoritmos inspirados ou não na natureza*: a maneira mais intuitiva de classificar as metaheurísticas baseia-se nas origens do algoritmo. Existem os algoritmos inspirados na natureza, como os Algoritmos Genéticos e o ACO, e não inspirados na natureza como Busca Tabu e Busca Local Iterativa. Para Blum e Roli (2003) esta classificação não é muito significativa por duas razões. Primeiro, existem muitos algoritmos híbridos que não se encaixam em nenhuma das classe (ou eles podem se encaixar em ambas ao mesmo tempo). Em segundo lugar, é difícil claramente atribuir um algoritmo para uma das duas classes.

- *Busca baseada em população ou busca com única solução*: outra característica que pode ser usada para a classificação de metaheurísticas é o número de soluções utilizadas ao mesmo tempo: o algoritmo pode trabalhar sobre uma população de soluções ou em uma única solução. Algoritmos que trabalham com soluções únicas são chamados métodos de trajetória e abrangem as metaheurísticas baseadas em buscas locais, como Busca Tabu, Busca Local Iterativa e Busca de Vizinhança Variável. Todos estes utilizam a propriedade de descrever uma trajetória no espaço de busca durante o processo de busca. Já as metaheurísticas baseadas em população descrevem a evolução de um conjunto de pontos no espaço de busca.

- *Função objetivo dinâmica ou estática*: as metaheurísticas também podem ser classificadas de acordo com a maneira que fazem uso da função objetivo. Enquanto alguns algoritmos mantem a função objetivo dada na representação do problema, outros, como por exemplo o *Guided Local Search* (GLS), podem modificá-la durante a busca. A ideia desta abordagem é escapar de mínimos locais modificando o cenário da busca. Assim, durante a busca a função objetivo é alterada para tentar incorporar as informações coletadas durante o processo de busca.

- *Estrutura de vizinhança única ou variada*: a maioria dos algoritmos de metaheurística utiliza uma única estrutura de vizinhança. No entanto, existem algumas, tais como a

Busca de Vizinhança Variável (VNS), que usam um conjunto de estruturas de vizinhança que oferece a possibilidade de diversificar a busca.

- *Métodos com ou sem (ou pouco) uso de memória*: uma característica importante para classificar as metaheurísticas é o uso que fazem do histórico de busca, isto é, se elas usam a memória ou não. Algoritmos sem (ou pouco) uso de memória utilizam exclusivamente o estado atual do processo de busca para determinar a próxima ação. Existem várias maneiras diferentes de fazer uso da memória. Faz-se uma diferenciação entre o uso da memória a curto e longo prazo. O primeiro geralmente mantém um registro de movimentos realizadas recentemente, soluções visitadas ou, em geral, decisões tomadas. A segunda é geralmente uma acumulação de parâmetros sobre a busca.

2.3 METAHEURÍSTICAS POPULACIONAIS

As metaheurísticas muito utilizadas na resolução de problemas de otimização combinatória, como as atividades de escalonamento, são baseadas em população, especialmente os algoritmos evolucionários. A seguir serão especificadas e descritas as metaheurísticas baseadas em população (especialmente os algoritmos evolucionários), as quais são objeto de estudo deste trabalho.

2.3.1 ALGORITMOS EVOLUCIONÁRIOS

Os algoritmos evolucionários são técnicas de otimização inspiradas nos processos evolutivos naturais. Eles trabalham com uma população de indivíduos que evoluem com a ajuda de procedimentos de troca de informações. Cada indivíduo também pode evoluir de forma independente.

Em cada iteração desses algoritmos, os períodos de auto-adaptação podem alternar com os períodos de cooperação. A auto-adaptação significa que os indivíduos evoluem de forma independente, enquanto a cooperação implica em uma troca de informações entre os indivíduos. Em geral, um algoritmo evolucionário pode ser definido como um algoritmo que inclui um mecanismo de melhoria para intensificar a exploração de algumas regiões do espaço de busca e uma estratégia de diversificação que ajude a escapar de ótimos locais neste espaço de busca (HERTZ; KOBLER, 2000).

Os algoritmos evolucionários são inspirados na capacidade da natureza em evoluir os seres vivos bem adaptados ao seu ambiente. Sucintamente, esses algoritmos podem ser caracterizados como modelos computacionais de processos evolutivos. A cada iteração um

número de operadores é aplicado aos indivíduos da população atual para gerar os indivíduos da população da próxima geração (iteração).

O princípio mais importante dos algoritmos evolucionários é a seleção de indivíduos com base em sua aptidão, que pode ser o valor de uma função objetivo ou o resultado de um experimento de simulação, ou algum outro tipo de medida de qualidade. Um indivíduo com maior aptidão têm maior probabilidade de ser escolhido como membro da população da próxima iteração (ou como os pais para a geração de novos indivíduos). Isto corresponde ao princípio de sobrevivência do mais apto na evolução natural. Esta capacidade da natureza em se adaptar a um ambiente em mudança é a inspiração para os algoritmos evolucionários (BLUM; ROLI, 2003).

No trabalho de Hertz e Kobler (2000) é apresentada uma visão geral das principais características dos algoritmos evolucionários e das possibilidades para definí-los:

- **Descrição dos indivíduos**: os algoritmos evolucionários trabalham com populações de indivíduos. Estes indivíduos não são necessariamente as soluções do problema considerado, eles podem ser soluções parciais, ou conjuntos de soluções, ou qualquer objeto que possa ser transformado em uma ou mais soluções de forma estruturada. O mais utilizado em otimização combinatória é a representação das soluções como cadeias de bits ou como permutações de n números inteiros. Árvores-estruturadas ou outras estruturas complexas também são possíveis. No contexto de algoritmos genéticos, os indivíduos são chamados genótipos, enquanto que as soluções que são codificados por indivíduos são chamados fenótipos. Isto é para diferenciar entre a representação de soluções e as soluções em si. A escolha de uma representação apropriada é crucial para o sucesso de um algoritmo evolucionário.

- **Processo de evolução**: em cada iteração tem que ser decidido quais os indivíduos entrarão na população da próxima iteração. Isto é feito por um esquema de seleção. Esta seleção dos indivíduos para a próxima população, exclusivamente com indivíduos descendentes, é chamada de substituição de gerações. Se for possível transferir indivíduos da população corrente para a população seguinte, então é chamado de estado estacionário no processo de evolução. A maioria dos algoritmos evolucionários trabalham com populações de tamanho fixo, mantendo pelo menos o melhor indivíduo da população atual. Também é possível ter um tamanho de população variável. No caso de uma população onde o tamanho possa diminuir continuamente, se ocorrer a situação em que apenas um indivíduo é deixado na população (ou nenhum parceiro para o cruzamento possa ser encontrado para qualquer membro da população) pode ser uma das condições de parada do algoritmo.

- **Estrutura de vizinhança**: se um indivíduo pode ser recombinado com qualquer outro indivíduo (como, por exemplo, no Algoritmo Genético) é chamado de populações não-estruturadas, caso contrário, é chamado de populações estruturadas.

- **Fontes de Informação**: a forma mais comum de fonte de informação para criar descendência (isto é, novos indivíduos) é um casal de pais. Mas há também operadores de recombinação que operam em mais de dois indivíduos para criar um novo indivíduo. Alguns algoritmos usam estatísticas populacionais para gerar os indivíduos da população seguinte.

- **Inviabilidade**: uma característica importante de um algoritmo evolucionário é a forma de tratar os indivíduos infactíveis. Ao recombinar indivíduos, a descendência pode ser potencialmente inviável. Existem basicamente três maneiras diferentes de lidar com tal situação: a ação mais simples é de rejeitar os indivíduos inviáveis; no entanto, para muitos problemas pode ser muito difícil encontrar indivíduos viáveis. Portanto, em alguns casos é mais apropriada a estratégia de penalizar os indivíduos inviáveis na função que mede a qualidade de um indivíduo; e a terceira possibilidade consiste em tentar reparar uma solução inviável.

- **Estratégia de intensificação**: em muitas aplicações já foi comprovado ser bastante proveitoso usar mecanismos de melhoria para melhorar a aptidão dos indivíduos. Algoritmos evolucionários que aplicam um algoritmo de busca local para cada indivíduo de uma população são frequentemente chamados de algoritmos meméticos. Embora a utilização de uma população assegure uma exploração do espaço de busca, o uso de técnicas de buscas locais ajudam a identificar rapidamente boas áreas no espaço de busca. Outra estratégia de intensificação é o uso de operadores de recombinação que explicitamente tentar combinar "boas" partes de indivíduos. Isto pode orientar a busca realizada por algoritmos evolucionários para áreas com indivíduos que possuam "boas" propriedades.

- **Estratégia de diversificação**: uma das principais dificuldades dos algoritmos evolucionários (especialmente quando usam uma busca local) é a convergência prematura para soluções sub-ótimas. O mecanismo mais simples para diversificar o processo de busca é a utilização de um operador de mutação. Um operador simples de mutação executa apenas uma pequena perturbação aleatória em um indivíduo, introduzindo um tipo de ruído. A fim de evitar a convergência prematura, existem formas de manter a diversidade da população. Uma das estratégias mais conhecidas é a pré-seleção, que pode usar algum método probabilístico ou seguir alguma regra de seleção pré-definida.

Os algoritmos evolucionários representam uma grande classe de metodologias na resolução de problemas, com os algoritmos genéticos (AGs), sendo um dos mais amplamente conhecidos. Mas existem diversos outros algoritmos evolucionários utilizados para resolver diferentes tipos de problemas, entre esses um algoritmo relativamente recente baseado em população e com bons resultados para a classe de problema apresentado neste trabalho é o *Shuffled Frog Leaping Algorithm* (SFLA). Dessa forma, no próximo capítulo será feita uma breve descrição dos AGs, utilizado neste trabalho como base de comparação e uma apresentação e descrição detalhada do algoritmo SFLA, o qual foi escolhido para ser o objeto de estudo deste trabalho.

Na seção seguinte é apresentada uma revisão bibliográfica de trabalhos que aplicaram metaheurísticas na solução das atividades de escalonamento com ênfase especial em trabalhos que abordam otimização multiobjetivo.

2.4 REVISÃO BIBLIOGRÁFICA

Um método heurístico para o problema de escalonamento de dutos para a distribuição de produtos derivados do petróleo foi desenvolvido por Sasikumar et al. (1997). A solução consiste de quatro componentes: o produto a ser enviado, sua quantidade, como o produto é distribuído até seu destino e a sequencia de envio. A resolução leva em consideração restrições no tamanho da batelada, no sequenciamento de envio, e de tancagem na origem e no destino. O objetivo é obter as datas de envio e chegada do produto nas diferentes localidades. As soluções alcançadas foram consideradas aceitáveis pelos usuários para o problema de escalonamento mensal.

Em Crane et al. (1999) é apresentado um método aplicando algoritmo genético para a resolução do problema simplificado de escalonamento de dutos. A rede de dutos modelada é composta por 8 nós terminais com 2 tanques em cada, ligados por 7 dutos unidirecionais, não permitindo a reversão, 2 produtos e 3 níveis de capacidade de armazenamento dos produtos nos tanques. Encontrou-se uma solução ótima para este problema, mostrando, assim, a possibilidade de resolução de problema de escalonamento de dutos através da aplicação de algoritmo genético.

O trabalho de Castro (2001) analisou os principais aspectos envolvidos na atividade de programação da produção de uma refinaria de petróleo, considerando o desenvolvimento de aplicativos computacionais para apoiar esta atividade. Utilizou como base um algoritmo genético, ressaltando a adequabilidade desta técnica para a otimização de problemas não

lineares que envolvem variáveis discretas e contínuas. Por fim, desenvolveu um método baseado em algoritmos genéticos para resolução da programação de produção do sistema de armazenamento de Gás Liquefeito de Petróleo da Refinaria Henrique Lage da Petrobras.

Em Almeida (2001) foi desenvolvido um método de solução baseado em Algoritmos Genéticos (GAs) aliado a um Sistema Baseado em Regras para encontrar e otimizar as soluções geradas para o problema de programação da produção de Óleos Combustíveis e Asfalto na REVAP (Refinaria do Vale do Paraíba). Foram desenvolvidos neste trabalho dois modelos baseados em algoritmos genéticos que são utilizados para encontrar a sequência e os tamanhos dos lotes de produção dos produtos finais. O Sistema Baseado em Regras é utilizado na escolha dos tanques que recebem a produção e os tanques que atendem à demanda dos diversos centros consumidores existentes. Um novo operador de mutação - Mutação por Vizinhança - foi proposto para minimizar o número de trocas operacionais na produção.

No trabalho de Cruz et al. (2003) foi apresentado um modelo de otimização multiobjetivo utilizando algoritmo genético para o problema de distribuição de derivados de petróleo. Neste trabalho, várias funções objetivo são consideradas e restrições são incluídas como parte da função objetivo. Prioridades e penalidades são impostas às funções objetivo. O modelo foi aplicado em um problema simplificado, com o objetivo principal de satisfazer a demanda de produtos em um tempo mínimo, minimizando a interface entre diferentes produtos.

O trabalho apresentado em Garcia et al. (2004) pode ser considerado uma evolução do trabalho de Cruz et al. (2003). Neste trabalho é apresentada uma abordagem híbrida utilizando duas técnicas: Método Heurístico através de um algoritmo multiobjetivo evolucionário e Programação Matemática usando o método das restrições na programação linear inteira mista. O trabalho comparou soluções obtidas através do método Híbrido, PLIM e Computação Evolucionária onde concluiu que métodos híbridos podem obter soluções ótimas em intervalos de tempo de execução menor.

O trabalho de Barboza (2005) aborda o problema programação da produção (escalonamento) envolvendo estocagem e distribuição de óleo diesel em uma refinaria de petróleo. Como solução foram utilizados modelos de Programação Linear Inteira Mista (PLIM), modelos híbridos combinando Algoritmos Genéticos, Algoritmos Transgenéticos e Programação Linear (PL). Os resultados apresentados mostraram que métodos baseados em metaheurísticas podem substituir métodos baseados em PLIM, achando soluções factíveis em um menor tempo computacional.

Em Westphal (2006), e também (WESTPHAL; ARRUDA, 2007) foi utilizado um algoritmo genético com estratégias de elitismo baseado em castas, considerando a otimização

multiobjetivo em um problema de escalonamento em uma rede simplificada de distribuição de derivados de petróleo. Uma aproximação do método da ponderação de objetivos (COELLO, 1999) foi usada para gerar prioridades entre os critérios de otimização. Os resultados apresentados alcançaram boas soluções em curto espaço de tempo.

Alves (2007) desenvolveu uma ferramenta computacional, baseada em Algoritmos Genéticos, para encontrar soluções viáveis para uma versão simplificada do Problema de Transporte de produtos na Rede de Escuros operada pela Transpetro. As simplificações do problema incluem: considerar que os dutos da rede podem operar em apenas um sentido e que não existem restrições de compatibilidade entre os produtos que são transportados contiguamente pelos dutos. Pelos experimentos computacionais realizados foram alcançadas soluções viáveis para as instâncias testadas.

Nos trabalhos de Yamamoto (2009) e (ARRUDA et al., 2010) realizou-se a implementação de várias técnicas metaheurísticas (Algoritmo Genético, Busca Tabu, GRASP, *Simulated Annealing*, Colônia de Formigas, *Scatter Search* e *Variable Neighbour Search*) para a resolução do problema de sequenciamento de bateladas de produtos derivados do petróleo para uma rede de dutos. A rede de dutos utilizada foi a rede de claros que faz parte da malha dutoviária da indústria P & G brasileira. Esta rede realiza o transporte dos produtos das refinarias e portos aos centros de distribuição.

Foram analisados os resultados alcançados com o uso de diferentes funções objetivo na otimização. Os resultados mostraram que dentre as metaheurísticas estudadas, o GRASP, VNS e AG foram as que apresentaram melhor desempenho em relação ao tempo de processamento e o melhor valor da função objetivo. Posteriormente, foi utilizada uma técnica de otimização multiobjetivo, o MOGA (*multiple objective genetic algorithm*), que apresentou bons resultados além de uma maior diversificação dos resultados finais.

No trabalho apresentado em (RIBAS et al., 2013), foi desenvolvido um modelo de otimização híbrido baseado em algoritmo genético micro-AG (μAG) e de programação linear inteira mista (PLIM) para auxiliar as atividades de escalonamento para uma rede de dutos que faz parte da malha dutoviária da indústria P & G brasileira. O algoritmo μAG proposto utiliza o modelo PLIM desenvolvido por (BOSCHETTO et al., 2010) para a classificação (ranqueamento) da população. Os resultados obtidos com o modelo μAG proposto quando comparado com outro modelo MOGA são semelhantes em termos de qualidade das soluções, o número total de soluções não-dominadas e dispersão das soluções na fronteira de Pareto. Em relação ao tempo computacional, o modelo μAG é muito menor que o modelo MOGA para todos os cenários estudados no trabalho. No entanto, a função de *fitness* utilizado

por ambos os modelos são baseados na solução de um modelo PLIM cujo tempo computacional aumenta com a complexidade do cenário.

2.5 CONSIDERAÇÕES

A partir dos trabalhos citados, nota-se que o uso de metaheurísticas na resolução de problemas de escalonamento em malha de polidutos é muito utilizado. Deste modo, com o objetivo de apresentar uma solução com desempenho satisfatório para o problema de escalonamento multiobjetivo em uma rede dutos, o trabalho de Yamamoto (2009) apresenta uma solução baseada em MOGA. No entanto, o modelo desenvolvido apresentou elevada carga computacional, limitando a utilização para modelos com um número limitado de bateladas.

Diante deste cenário, este trabalho propõe continuar a pesquisa do trabalho apresentado em Yamamoto (2009), investigando e aplicando outras técnicas metaheurísticas para a resolução do problema de escalonamento, mais precisamente para a alocação e sequenciamento de transporte de produtos em rede de dutos.

Para isto, desenvolveu-se modelos de otimização multiobjetivo baseado em uma técnica metaheurística conhecida como SFLA (*Shuffled Frog Leaping Algorithm*) associado ao principio da otimalidade de Pareto para determinar o conjunto de soluções do modelo multiobjetivo proposto. Para validar o modelo será feita uma comparação de resultados com os algoritmos microgenéticos (WESTPHAL et al., 2011; RIBAS et al., 2013) e um algoritmo bem conhecido na literatura para otimização multiobjetvo: o NSGA-II (DEB et al., 2002a).

Além disso, o trabalho de Yamamoto (2009) utilizou a rede de claros em seu modelo para solucionar o problema de escalonamento e para este trabalho propõe-se a modelagem da rede de escuros que é parte integrante do sistema dutoviário do estado de São Paulo operado pela Transpetro.

Nesse contexto, o próximo capítulo apresenta a fundamentação teórica dos algoritmos evolucionários utilizados neste trabalho. Assim como descreve os problemas de otimização multiobjetivo, príncipio de otimalidade de Pareto e otimização multiobjetivo evolucionária, com o objetivo de apresentar conceitos e definições que servem de apoio para a compreensão do método proposto neste trabalho.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma descrição dos algoritmos evolucionários utilizados neste trabalho. Em seguida, apresenta-se de uma forma breve o problema de otimização multiobjetivo, bem como os métodos e técnicas de solução para problemas multiobjetivos, o princípio da otimalidade de Pareto e otimização multiobjetivo evolucionária.

3.1 ALGORITMOS EVOLUCIONÁRIOS

Os algoritmos evolucionários são métodos de busca estocástica que imitam a evolução biológica natural e/ou o comportamento social das espécies. Esses algoritmos têm sido desenvolvidos para se chegar a soluções ótimas ou próximas do ótimo para problemas de otimização complexos e de grande escala (ELBELTAGI et al., 2005).

Neste trabalho foram utilizados três algoritmos evolucionários, o *Shuffled frog-leaping Algorithm* (SFLA) com uma pequena modificação proposta por (ELBELTAGI et al., 2007), o Algoritmo micro Genético (μAG) proposto por (WESTPHAL et al., 2011; RIBAS et al., 2013) e algoritmo NSGA-II (DEB et al., 2002a).

Nas seções seguintes, primeiramente é descrito a versão original de cada um dos algoritmos utilizados neste trabalho (AG, SFLA e NSGA), e em seguida são apresentadas as versões dos algoritmos com as modificações utilizados neste trabalho.

3.1.1 ALGORIMOS GENÉTICOS - AGS

Os princípios básicos dos Algoritmos Genéticos (AGs) foram desenvolvidos por Holland (1975). Este algoritmo é um método de busca baseada em mecanismos de evolução natural e genética (GOLDBERG, 1989).

Os AGs são motivados pela forma que as espécies evoluem e se adaptam ao seu ambiente com base no princípio Darwiniano de seleção natural (GENDREAU; POTVIN, 2005). Nesse contexto, uma população de soluções evolui de uma geração para a seguinte, através da

aplicação de operadores que imitam os encontrados na natureza, ou seja, a seleção do mais adaptado, cruzamento e mutação. Através do processo de seleção, apenas as melhores soluções são permitidas a se tornarem pais e gerar descendência. O processo de geração de descendentes, chamado *crossover*, seleciona duas soluções pais e combina as suas melhores características para criar uma ou duas soluções descendentes. Isto é repetido até que uma nova população de soluções descendentes seja criada.

Antes de substituir a antiga população, cada membro da nova população é submetido (com uma pequena probabilidade) a pequenas perturbações aleatórias através do operador de mutação. A partir de uma população inicial gerada aleatoriamente ou heuristicamente, este ciclo de renovação é repetido por um dado número de iterações, e a melhor solução encontrada é devolvida ao final. Nos AGs, uma população inicial é gerada aleatoriamente ou utilizando métodos heurísticos, essas soluções representam pontos espalhados no espaço de busca. Cada solução possível do espaço de busca é representada sob a forma de uma cadeia ou vetor (geralmente representada por bits ou números inteiros), chamado cromossomo, consistindo de um conjunto de elementos, chamados genes, que possuem um conjunto de valores para as variáveis de otimização (GOLDBERG, 1989).

Assim, a população é formada por um conjunto de soluções factíveis codificadas em cromossomos. Cada cromossomo representa um indivíduo na população e cada indivíduo tem um valor que mede seu grau de aptidão. Esse valor é denominado *fitness* e geralmente é o que identifica a qualidade da solução.

Um AG padrão utiliza três operadores genéticos: a seleção, cruzamento ou recombinação (*crossover*) e a mutação. Os operadores genéticos são aplicados aos indivíduos da população com o objetivo de reproduzir novos e melhores indivíduos a partir dos já existentes. As operações são necessárias para permitir a diversidade dos indivíduos, bem como explorar outras regiões do espaço de busca.

Os principais parâmetros do AG são: o tamanho da população, número de gerações e a taxa de probabilidade de *crossover* e mutação. Uma breve descrição do algoritmo pode ser vista na Tabela 1.

Seleção

Para realizar o processo de cruzamento ou recombinação no algoritmo, é feita uma seleção entre os indivíduos pertencentes à população. Esta seleção ocorre após a avaliação de cada indivíduo, geralmente esta seleção é baseada no princípio da sobrevivência dos melhores indivíduos.

```
Inicio
Gerar uma população inicial de M indivíduos;
Calcular o fitness para cada indivíduo da população M;
Ordenar os M indivíduos por ordem decrescente do valor de fitness;
| Repetir por um número I de gerações:
|−→| Selecionar indivíduos;
|−→| Cruzar indivíduos selecionados;
|−→| Mutar indivíduos selecionados;
|−→| Avaliar o fitness da população;
| Fim;
Fim.
```

Tabela 1: Pseudo-código do Algoritmo Genético básico

Os indivíduos com os melhores níveis de aptidão possuem uma maior probabilidade de serem mantidos e selecionados para a etapa de cruzamento. Da mesma forma, os cromossomos com níveis baixos de aptidão possuem pouca probabilidade de permanecer e, consequentemente, podem ser eliminados da população. Portanto, o operador de seleção tem o objetivo de escolher os indivíduos que devem continuar o processo evolutivo conforme estratégia adotada.

As principais estratégias de seleção de um indivíduo para a próxima geração são: seleção *rank*, seleção por roleta, seleção por torneio e técnica elitista. Uma descrição detalhada destas estratégias pode ser vista em Michalewicz (1996).

Crossover

Após o processo de seleção, os indivíduos selecionados passam para o processo de *crossover*. O *crossover* ou recombinação genética é a etapa de cruzamento realizado pelos indivíduos selecionados. Nesta etapa é realizada a troca de genes entre dois ou mais indivíduos, que são denominados pais, formando novos indivíduos que são chamados de filhos. O objetivo deste processo é a troca de informações entre diferentes soluções candidatas (GOLDBERG, 1989).

O processo de *crossover* nos indivíduos selecionados ocorre com certa probabilidade, que pode ser definida como um parâmetro inicial de projeto. O valor dessa taxa de probabilidade influencia na convergência do algoritmo, como por exemplo, valores altos reduzem as chances de convergência para um máximo local, mas acabam por resultar em maiores perdas computacionais devido à exploração de regiões não promissoras dentro do espaço de busca.

Existem diversos operadores de recombinação propostos na literatura, com aplicação direcionada a diferentes tipos de codificações e problemas. Os mais comuns são recombinação ponto único, recombinação multiponto e recombinação uniforme (MICHALEWICZ, 1996).

Mutação

O objetivo do processo de mutação é diversificar a genética da população. Este operador realiza uma alteração aleatória de uma ou mais características de um cromossomo selecionado, propiciando assim, a introdução de novos elementos na população.

Geralmente, o processo de mutação é aplicado aos indivíduos após o processo de *crossover*. O operador mutação é aplicado aos indivíduos com certa probabilidade. Se esta probabilidade for baixa, pode haver comprometimento na diversidade dos indivíduos. Contudo se a probabilidade for alta pode haver perturbações aleatórias de tal forma que os filhos provavelmente perderão suas semelhanças com os pais podendo assim comprometer a convergência do método (WESTPHAL, 2006).

3.1.2 ALGORITMO MICRO GENÉTICO (μAG)

A extensão do AG para aplicações em problemas multiobjetivos requer a inclusão de um mecanismo para ranqueamento da população que realiza uma comparação entre todos os indivíduos a fim de determinar sua dominância no sentido de Pareto e um segundo mecanismo para manter a diversidade a fim de evitar a convergência prematura do algoritmo devido à reprodução preponderante de super-indivíduos (COELLO, 2003). Estes dois mecanismos são responsáveis pelo aumento da carga computacional dos métodos multiobjetivos baseados em AG.

Neste contexto, os métodos baseados em algoritmo micro-genético permitem a redução do tempo de processamento enquanto mantém uma precisão aceitável, se comparado com soluções obtidas via AG padrão (COELLO; PULIDO, 2001a; WESTPHAL et al., 2011).

Essa redução do tempo computacional deve-se ao fato que o pequeno número de indivíduos na população reduz o esforço para avaliação e ranqueamento de cada indivíduo. Além disso, o uso de estratégias de reinicialização da população após poucas gerações garante a convergência do algoritmo micro-genético e mantém a diversidade da população, evitando uma convergência prematura para um super-indivíduo.

A Figura 1 ilustra como funciona o algoritmo μAG proposto por (COELLO, 2005). Inicialmente uma população aleatória é gerada. Essa população aleatória "alimenta" a memória da população, que está dividida em duas partes: uma substituível e um parte não substituível. A porção não substituível da memória da população não muda durante toda a execução e se destina a fornecer a diversidade para o algoritmo. Já na parte substituível ocorrem mudanças depois de cada ciclo do μAG.

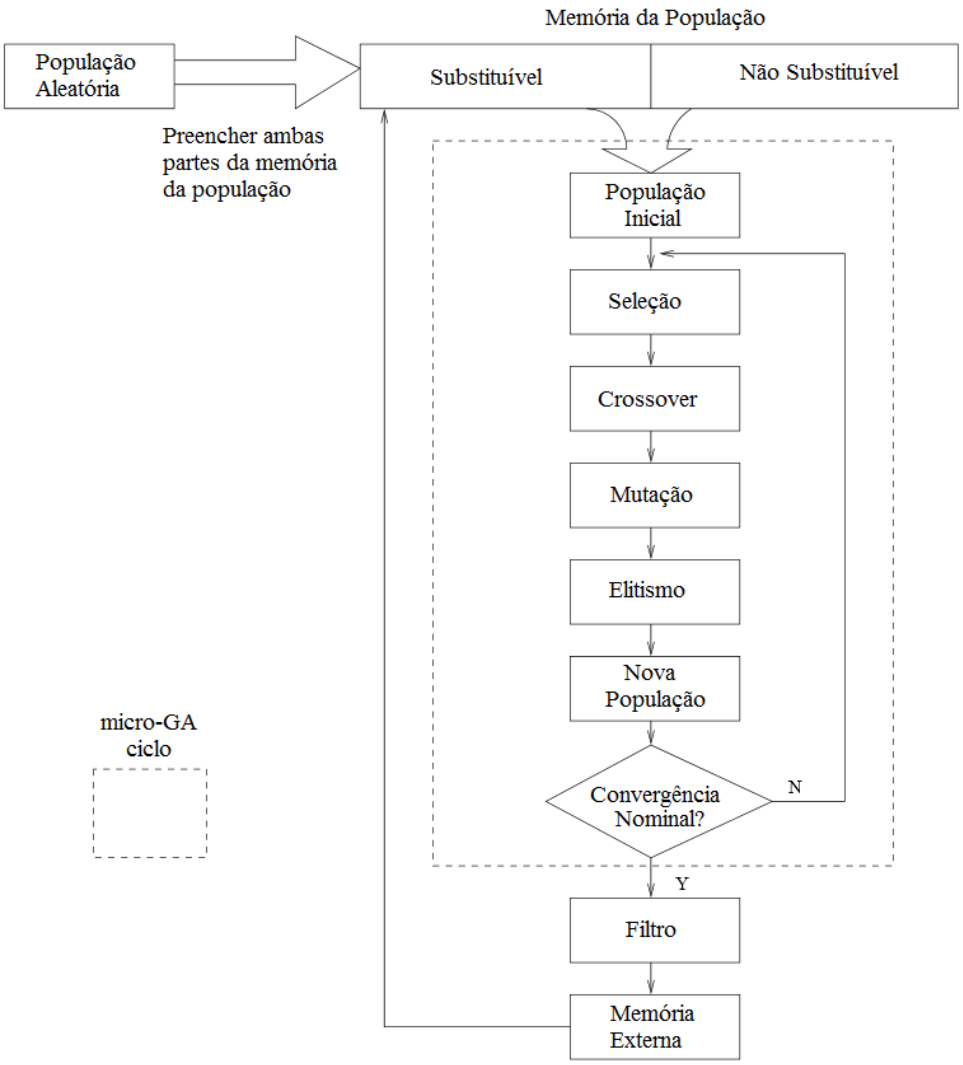


Figura 1: Diagrama que ilustra como o algoritmo micro-AG trabalha para um problema multiobjetivo.

Fonte: Adaptado de Coello (2005)

A população do μAG no início de cada um dos seus ciclos é preenchida (com uma certa probabilidade) com ambas porções da memória de população de forma que ocorra uma mistura de indivíduos gerados aleatoriamente (parte não substituível) e indivíduos evoluídos (parte substituível). Durante cada ciclo, o algoritmo utiliza os operadores genéticas convencionais.

Após o μAG terminar um ciclo, dois indivíduos não-dominados são escolhidos a partir da população final e são comparados com o conteúdo da memória externa (esta memória está inicialmente vazia). Se qualquer um deles (ou ambos) permanecer como não-dominado depois de comparará-los com os indivíduos presentes na memória externa, então eles são incluídos na memória externa. Este é o arquivo histórico de indivíduos não-dominados. Todos os indivíduos dominados contidos na memória externa são eliminados. O μAG então utiliza três formas de

elitismo: (1) retém as soluções não-dominadas encontradas dentro do ciclo interno do μAG, (2) utiliza uma memória substituível cujo conteúdo é parcialmente "atualizado"em determinados intervalos, e (3) substitui a população do μAG pelas melhores soluções produzidas depois de um ciclo completo interno do μAG.

Neste trabalho utiliza-se a versão do algoritmo microgenético multiobjetivo com formação de nicho e casta apresentado em Westphal et al. (2011) e em (RIBAS et al., 2013), o qual foi originalmente desenvolvido para um problema de escalonamento de rede de dutos semelhante ao problema apresentado neste trabalho.

O algoritmo μAG citado combina técnicas de ranqueamento e elitismo para a formação de nicho e castas que garante ao final da evolução, um conjunto de boas soluções em relação aos diferentes objetivos do problema de otimização.

A utilização de uma pequena população com uma memória externa e uma casta interna garante a diversidade na população e permite alcançar um conjunto de soluções bem distribuído ao longo da fronteira de Pareto do problema multi-objetivo. Mais detalhes podem ser encontrados em Westphal et al. (2011) e (RIBAS et al., 2013).

3.1.3 *SHUFFLED FROG LEAPING ALGORITHM* - SFLA

O algoritmo *Shuffled Frog Leaping Algorithm* (SFLA), proposto por Eusuff et al. (2006), é uma metaheurística desenvolvida para procurar uma solução ótima global, realizando uma busca heurística através de qualquer função matemática.

O SFLA é de base populacional e utiliza uma busca cooperativa inspirada na memética natural onde os indivíduos interativos realizam troca global de informação entre si. O termo memética vem de "meme" (DAWKINS, 1976). Os memes podem ser considerados como uma unidade de evolução das culturas. As idéias evoluem de uma maneira análoga à evolução biológica (EUSUFF; LANSEY, 2003).

Dawkins define o meme simplesmente como uma unidade de informação intelectual ou cultural que sobrevive o tempo suficiente para ser reconhecido como tal e que pode passar de mente para mente. Exemplos de memes são canções, idéias, slogans, roupas da moda e maneiras de fazer potes ou de construir arcos. O conteúdo real de um meme, chamado memotipo, é análogo ao do cromossomo de um gene. Um padrão de idéia ou informação não é um meme até alguém replicá-lo ou repeti-lo para outra pessoa. Todo o conhecimento transmitido é memético (EUSUFF et al., 2006).

A evolução memética e genética estão sujeitas aos mesmos princípios básicos, ou

seja, as possíveis soluções são criadas, selecionadas de acordo com alguma medida de *fitness*, combinada com outras soluções e possivelmente sofrem alguma mutação. No entanto, a evolução memética é um mecanismo muito mais flexível. Os objetivos implícitos de genes e memes são diferentes na medida em que eles usam diferentes mecanismos de troca de informações de um membro da população para outro membro. Os genes só podem ser transmitidos de pais para filhos. Os memes, em princípio, podem ser transmitidos entre quaisquer dois indivíduos.

Na evolução genética, a transmissão dos genes normalmente ocorre entre as gerações. Na evolução memética, se uma ideia melhorada é encontrada, ela pode ser incorporada em outros memes imediatamente, em vez de esperar para uma geração completa de genes a serem replicados. Além disso, a replicação do gene é limitada pelo número relativamente pequeno de descendentes que um único pai pode ter, ao passo que o número de indivíduos que podem assumir um meme a partir de um único indivíduo é quase ilimitado. Outra diferença entre memes e genes é que os memes são processados e possivelmente melhorados pela pessoa que os detém, algo que não pode acontecer com os genes (EUSUFF et al., 2006).

O SFLA usa a evolução memética sob a forma de infecção de idéias entre os indivíduos na busca local, e uma estratégia de embaralhamento que permite a troca de informações entre as buscas locais para se mover em direção a um ótimo global. No processo de embaralhamento, a população é dividida em vários grupos e depois de um número definido de evoluções, os grupos são forçados a misturar-se e novos grupos são formados. Esta estratégia ajuda a melhorar a solução através da partilha de informações e de propriedades de forma independente, adquirida por cada grupo (EUSUFF et al., 2006).

O *Shuffled Frog Leaping Algorithm* representa um grupo de sapos saltando em um pântano (espaço de busca) que tem um número de pedras localizadas em diversas regiões (possíveis soluções), nas quais os sapos (soluções) podem saltar para encontrar a pedra que possui a maior quantidade de alimento disponível. Os sapos comunicam-se uns com os outros para melhorarem seus memes (idéias) usando a informação dos outros. A Figura 2 ilustra essa representação geral do algoritmo.

O SFLA progride pela troca de posições dos sapos por meio da evolução memética, os sapos possuem um conjunto de memes descritos por um vetor memético. Como observado anteriormente, memes e memotipos são análogos aos genes e cromossomos, respectivamente. A melhoria de um meme resulta em alterar a posição de um único sapo pela mudança no tamanho do salto (memotipo) do sapo. A seguir, é feita uma breve descrição do passo a passo do algoritmo SFLA.

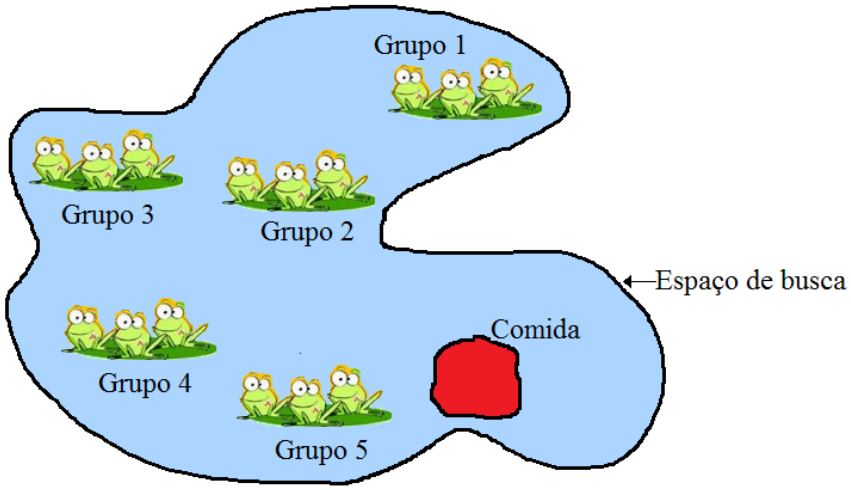


Figura 2: SFLA - Busca dos grupos por comida.
Fonte: Adaptado de Elbeltagi et al. (2005)

Primeiramente, uma população inicial de F sapos é criada aleatoriamente. Para problemas de S dimensões, cada sapo i é representado por S variáveis: $X_i = (x_{i1}, x_{i2}, ..., x_{iS})$. Os sapos são ordenados em ordem decrescente do valor do *fitness*. Então toda a população é dividida em m grupos chamados de *memeplexes*, cada um contendo n sapos $(F = m \times n)$. Neste processo, o primeiro sapo da população vai para o primeiro *memeplex*, o segundo sapo vai para o segundo *memeplex*, o sapo m vai para o mth *memeplex*, e o sapo $m+1$ vai para o primeiro *memeplex* e assim por diante.

Dentro de cada *memeplex*, os sapos com o melhor e o pior *fitness* são identificados como X_b e X_w, respectivamente. Além disso, o sapo com o melhor *fitness* global é identificado como X_g. Então um processo de evolução é aplicado para melhorar o sapo com o pior *fitness* em cada ciclo. A posição do sapo com o pior *fitness* é ajustada conforme as seguintes fórmulas:

$$S_+ = min\{int[rand()(X_b - X_w)], S_{max}\} \tag{1}$$

$$S_- = max\{int[rand()(X_b - X_w)], -S_{max}\} \tag{2}$$

$$U_q = X_w + S \tag{3}$$

onde S_+ e S_- são o passo positivo e negativo, respectivamente. O *rand()* é uma função aleatória no intervalo $[0, 1]$, S_{max} é o tamanho máximo do passo permitido para mudar a posição

do sapo. A nova posição do sapo é calculada pela equação (3).

Se o processo produzir um sapo melhor, o sapo X_w é substituído. Caso contrário, os cálculos com as equações (1), (2) e (3) são repetidos, mas utilizando o sapo com o melhor fitness global X_g.

Se mesmo assim não houver melhora na posição do sapo, então um novo sapo é gerado aleatoriamente para substituir o sapo X_w. O processo continua por um número N de passos evolutivos em cada *memeplex*.

Os principais parâmetros do SFLA são: o número de sapos F, número de memeplexes m e o número de passos evolutivos para cada *memeplex*. Uma breve descrição do algoritmo pode ser vista na Tabela 2.

```
Inicio
Gerar população aleatória de F sapos;
Calcular a aptidão para cada sapo F;
| Repetir por I de iterações:
|→| Ordenar os F sapos pelo valor de aptidão;
|→| Dividir os F sapos em m memeplexes;
|→| Para cada memeplex:
|—→| Repetir por um número N de evoluções:
|———→| Determinar o melhor e o pior sapo do memeplex;
|———→| Melhorar a posição do pior sapo;
|———→| Calcular a nova aptidão;
|———→| Se a posição for melhorada:
|—————→| substituir o sapo;
|———→| Se não:
|—————→| gerar um novo sapo aleatoriamente e substituir o sapo;
|—→| Fim;
|→| Fim;
|→| Embaralhar os memeplexes evoluídos;
| Fim;
Fim.
```

Tabela 2: Pseudo-código do Algoritmo *Shuffled Frog Leaping*

O *Shuffled Frog-Leaping algorithm* (SFLA) tem como característica mais relevante, a velocidade de convergência (ELBELTAGI et al., 2005). Além disso, tem as vantagens de ser simples, poucos parâmetros, boa performance e programação fácil. O algoritmo foi testado em várias funções de *benchmark* mostrando sua eficiência em muitos problemas de otimização global (EUSUFF et al., 2006). Baseado nisso, o SFLA foi escolhido para ser estudado e utilizado neste trabalho como uma nova opção de aplicação de metaheurística. O algoritmo SFL ou SFLA (*Shuffled Frog Leaping Algorithm*) têm sido usado para resolver problemas de

otimização discreto, assim como problemas de otimização contínua. Por exemplo, a eficácia e adequação do algoritmo SFLA para problemas complexos de otimização foi demonstrado no trabalho de (ELBEHAIRY et al., 2006), quando este foi aplicado ao problema de reparação de *bridge deck*. Além disso, um novo algoritmo multiobjetivo híbrido baseado no SFLA e otimização bacteriana (BOS), chamado HMOSFLA, é proposto em (RAHIMI-VAHED et al., 2009) para encontrar soluções ótimas de Pareto para um problema de escalonamento do tipo *flow shop*. Os resultados sugerem que HMOSFLA tem um bom desempenho, especialmente para problemas de grande porte.

No trabalho de Rahimi-Vahed e Mirzaei (2007), o SFLA é utilizado em um problema de escalonamento multi-critério para a programação em *flow shop* permutacional. Devido a complexidade do problema, os autores propõem uma nova abordagem, o *multi-objective shuffled frog-leaping algorithm* (MOSFLA) para buscar localmente a fronteira de Pareto. Para provar a eficiência do algoritmo proposto, é feito uma comparação com três algoritmos genéticos multi-objetivos: PS-NC GA, NSGA-II e SPEA-II. Os resultados mostram que o algoritmo MOSFLA proposto tem um melhor desempenho do que os algoritmos genéticos citados, em particular para os problemas de grande porte.

Em Chung e Lansey (2009), um modelo de otimização geral de larga escala de um sistema de abastecimento de água foi formulado como um problema inteiro não linear (PNL). A formulação foi aplicada a duas redes hipotéticas de água: uma rede médio porte e outra rede grande. Com dificuldades em resolver o problema com base nos métodos PNL, oa autores utilizaram o SFLA (*Shuffled Frog Leaping Algorithm*) e obtiveram uma pequena melhoria nos resultados.

O trabalho de Lakshmi e Rao (2010) propõe uma versão híbrida do algoritmo SFLA para resolver um problema de otimização de estruturas de compósitos laminados. Esta versão hibrida do algoritmo SFLA incorpora um algoritmo de busca de vizinhanças e um fator de busca adaptativa para acelerar as características de convergência. Além disso, um operador de cruzamento popularmente usado na computação evolutiva é adequadamente incorporado na versão híbrida de SFLA. Os estudos realizados mostram que o algoritmo proposto supera o estado da arte de algoritmos evolucionários como NSGA-II, PAES e microAG.

Um novo algoritmo multiobjetivo chamado *Shuffled Frog Leaping Algorithm* (MOSFLA) é aplicado para resolver as operações de controle de reservas de enchentes em (LI et al., 2010). No algoritmo proposto, uma estratégia de arquivamento baseado no método de nicho auto-adaptativo é incorporada para manter as soluções não dominadas. Os resultados demonstram que o MOSFLA pode gerar um conjunto solução com

boa propagação e convergência para problemas com objetivos conflitantes.

Existem vários outros trabalhos na literatura que utilizam o algoritmo SFLA: Eusuff e Lansey (2003) usaram o SFLA para determinar tamanhos ótimos de dutos para novas redes de tubulação e para a expansão da rede; Amiri et al. (2009) propuseram uma aplicação do SFLA para agrupamento de dados; Elbeltagi et al. (2005) compararam o SFLA com outros algoritmos baseados em evolução, tais como: algoritmos genéticos (AG), algoritmos meméticos (AM), otimização por enxame de partículas (PSO) e otimização por colônia de formigas (ACO). A comparação revela que o SFLA é uma técnica de otimização relativamente boa. Ele tem um desempenho semelhante ao PSO e supera o AG em termos de taxa de acerto, qualidade da solução e tempo de processamento.

O SFLA é, portanto, uma abordagem promissora para problemas combinatórios multiobjetivos (LAKSHMI; RAO, 2010) como o estudado neste trabalho. Neste trabalho foram utilizados dois algoritmos evolucionários modificados e melhorados, o Algoritmo micro Genético (μAG) e o *Shuffled Frog-Leaping Algorithm* (SFLA) modificado. Além disso, uma nova versão do algoritmo SFLA é proposta, o *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA), que utiliza o conceito de otimalidade de Pareto para encontrar um conjunto de soluções não dominadas.

3.1.4 *SHUFFLED FROG-LEAPING ALGORITHM* (SFLA) MODIFICADO

Uma versão modificada do algoritmo SFLA foi utilizada neste trabalho. No algoritmo SFLA padrão, cada *memeplex* pode evoluir de forma independente para buscas locais em diferentes regiões do espaço da solução. Além disso, no processo de evolução todos os *memeplexes* são embaralhados e novamente divididos em um novo conjunto de *memeplexes*, o que resulta em uma busca global através da troca de informação entre *memeplexes*. Assim, o algoritmo SFLA tenta encontrar um equilíbrio entre uma busca ampla no espaço de solução e uma busca mais "profunda" em localizações promissoras que estão perto de um ótimo local.

Conforme as equações 1 e 2 mostradas na Seção 3.1.3, cada sapo em um *memeplexe* está tentando mudar sua posição em relação a melhor sapo dentro do *memeplex* ou do melhor sapo global. Como mostrado nessas equações, quando a diferença da posição entre o pior sapo X_w (ou seja, o sapo em evolução) e o melhores sapos (X_b ou X_g) torna-se pequena, a mudança da posição do sapo X_w será muito pequena e portanto, pode estagnar em um ótimo local e levar à convergência prematura. Para superar tal ocorrência, o lado direito das equações 1 e 2 é multiplicado por um fator C chamado de "fator de aceleração de busca" (ELBELTAGI et al., 2007).

$$S_+ = min\{int[rand() \cdot C \cdot (X_b - X_w)], S_{max}\} \tag{4}$$

$$S_- = max\{int[rand() \cdot C \cdot (X_b - X_w)], -S_{max}\} \tag{5}$$

Atribuir um valor grande para o fator C no início do processo de evolução irá acelerar a busca global, permitindo uma maior mudança na posição do sapo, consequentemente, aumentará o espaço de busca global. Então, como o processo de evolução continua e um local promissor é identificado, o fator C de aceleração, será o foco do processo em uma busca local mais profunda, uma vez que permitirá os sapos mudarem suas posições. O fator de aceleração, que pode ser um valor positivo constante, linear, não linear ou função de tempo, fornece os meios para o equilíbrio entre a busca global e local (ELBELTAGI et al., 2007).

3.1.5 *NON-DOMINATED SORTING GENETIC ALGORITHM* - NSGA-II

O NSGA-II é uma versão melhorada do NSGA e foi apresentada em (DEB et al., 2002a). Os principais mecanismos acrescentados ao algoritmo NSGA-II são: o elistimo, para preservar as boas soluções no processo de busca; o procedimento *Fast Nondominated Sorting* (FNS) que classifica a população em diferentes níveis segundo a dominância de Pareto e o procedimento *Crowding Distance Assignment* que garante a diversidade da população.

A seguir, uma descrição detalhada do algoritmo NSGA-II:

1. Inicialização da População: A população é inicializada com base no domínio das variáveis do problema e restrições, se houver.

2. Rank não-dominadas: A população inicializada é classificado com base na dominância de Pareto. O procedimento FNS é descrito abaixo:

   - Para cada indivíduo p na população principal P:
     - Inicializar $S_p = 0$. Este conjunto deverá conter todos os indivíduos que são dominados por p.
     - Inicializar $n_p = 0$. Este seria o número de indivíduos que dominam p.
     - Para cada indivíduo q em P
       - se p domina q então
         adiciona q no conjunto S_p, isto é, $S_p = S_p \cup q$

    * se q domina p então

      incrementa o contador de dominação para p, isto é $n_p = n_p + 1$

  - Se $n_p = 0$, ou seja, não há indivíduos que dominam p então p pertence a primeira fronteira; Definir o rank do indivíduo p para um, isto é, $p_{rank} = 1$. Atualize o primeiro conjunto da fronteira pela adição de p a fronteira ,isto é, $F_1 = F_1 \cup p$

- Inicializar o contador de fronteira para um. $i = 1$
- A seguinte etapa é realizada enquanto a ith fronteira não está vazio ou seja $F_i \neq 0$
  - $Q = 0$. O conjunto para armazenar os indivíduos para $(i+1)^{th}$ fronteira.
  - para cada indivíduo p na fronteira F_i
    * para cada indivíduo q em S_p (S_p é o conjunto de indivíduos dominados por p)

      . $n_q = n_q - 1$, decrementa o contador de dominação para o indivíduo q.

      . se $n_q = 0$ então nenhum dos indivíduos nas fronteiras subsequentes dominaria q. Então definir $q_{rank} = i + 1$. Atualizar o conjunto Q com o indivíduo q, isto é $Q = Q \cup q$
  - Incremente o contador de fronteira em um.
  - Agora o conjunto Q é a próxima fronteira e portanto $F_i = Q$

Este algoritmo é melhor do que o NSGA inicial, uma vez que utiliza as informações sobre o conjunto que um indivíduo domina (S_p) e número de indivíduos que dominam o indivíduo (n_p).

3. *Crowding Distance*: depois de terminar o procedimento FNS, a distância de multidão (*Crowding Distance*- CR) é atribuída aos indivíduos. Uma vez que os indivíduos são selecionados com base no rank e *crowding distance*, para todos os indivíduos da população é atribuído um valor de *crowding distance*. O *crowding distance* é calculado da seguinte forma:

   - Para cada fronteira F_i, n é o número de indivíduos.
     - Inicializar a distância com o valor zero para todos os indivíduos, isto é, $F_i(d_j) = 0$, onde j corresponde ao $j-$iésimo indivíduo na fronteira F_i.
     - Para cada função objetivo m
       * Classifique os indivíduos da fronteira F_i baseado no objetivo m, isto é, $I = sort(F_i, m)$.

- Atribua distância infinita para os valores extremos(maior e menor valor para a função objetivo) para cada indivíduo do F_i, isto é, $I(d_1) = \infty$ e $I(d_n) = \infty$
- Para $k = 2$ até $(n-1)$
  . $I(d_k) = I(d_k) + \frac{I(k+1).m - I(k-1).m}{f_m^{max} - f_m^{min}}$
  . $I(k).m$ é o valor da $m-$ésima função objetivo do $k-$ésimo indivíduo em I.

A ideia básica por trás do *crowding distance* é encontrar a distância euclidiana entre cada indivíduo em uma fronteira com base em seus m objetivos no espaço hiper dimensional m. Os indivíduos dos limites/extremos são sempre selecionados já que eles têm atribuição de distância infinita.

4. Seleção: uma vez que os indivíduos são ranqueados com base no conceito de não-dominância e com os valores de *crowding distance* atribuídos, a seleção é realizada utilizando o ***crowded-comparison-operator*** ($\prec_n$). A comparação é efetuada tal como a seguir com base em:

   (1) No rank de não-dominação p_{rank}, isto é, os indivíduos na fronteira F_i terão seu rank como $p_{rank} = i$

   (2) *crowding distance* $F_i(d_j)$

   - $p \prec_n q$ se
     - $p_{rank} < q_{rank}$
     - ou se p e q pertencerem a mesma fronteira F_i então $F_i(d_p) > F_i(d_q)$, isto é, o valor do *crowding distance* deverá ser maior.

   Os indivíduos são selecionados usando uma seleção de torneio binário com o ***crowded-comparison-operator***.

5. Operadores genéticos: usa para soluções real-codificadas o *Simulated Binary Crossover*-(SBX) (DEB; AGRAWAL, 1995) para operação de *crossover* e mutação polinomial.

6. Recombinação e Seleção: A população descendente é combinada com a geração da população atual e a seleção é realizada para definir o conjunto de indivíduos da geração seguinte. Uma vez que todos os melhores indivíduos, tanto os anteriores como os atuais são adicionados na população, o elitismo é garantido. A população então é classificada com base no conceito de não-dominância. A nova geração é preenchida por cada fronteira subsequente até que o tamanho da população exceda o tamanho atual da população. Se ao adicionar todos os indivíduos da fronteira F_j a população excede N, então todos os

indivíduos da fronteira F_j são selecionados com base em seu *crowding distance* na ordem descendente até que o tamanho da população seja N. E, portanto, o processo se repete para gerar as gerações subsequentes. Na Figura 3 é possível ver a representação gráfica desse procedimento.

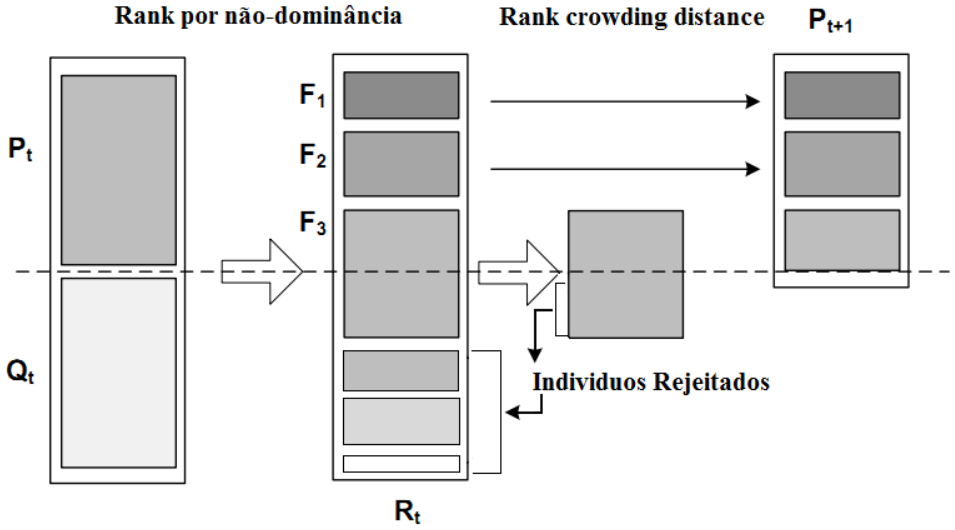


Figura 3: NSGA-II - procedimento de escolha da população
Fonte: Adaptado de Deb et al. (2002a)

3.2 PROBLEMAS DE OTIMIZAÇÃO MULTIOBJETIVO

Um problema de otimização multiobjetivo (POM) envolve problemas nos quais há vários objetivos (ou critérios) a serem alcançados. Geralmente, esse objetivos são conflitantes entre si e muitas vezes não existe uma solução que seja ótima para todos os objetivos simultaneamente.

O princípio da otimização multiobjetivo é diferente daquele de um único critério. Na otimização de um único critério, a intenção é encontrar a melhor solução que corresponde ao valor mínimo ou máximo da função objetivo. Por outro lado, em uma otimização multicritério com objetivos conflitantes, não há uma única solução ótima. A interação entre os diferentes objetivos dá origem a um conjunto de soluções ótimas (DEB, 1999).

As funções objetivos formam a descrição matemática do critério de desempenho, os quais geralmente são conflitantes. Assim, o termo otimizar pode significar descobrir uma

solução, para a qual os valores de todas as funções objetivos são consideradas aceitáveis pelo projetista ou decisor (WESTPHAL, 2006).

Um problema geral de otimização multiobjetivo pode ser descrito por um vetor de função f de i objetivos que depende de um vetor x de variáveis de decisão. Formalmente:

$$\min \setminus \max y = F(x) = (f_1(x), f_2(x), ..., f_i(x)) \tag{6}$$

Sujeito a:

$$g_j(x) \leq 0, j = 1, ..., m \tag{7}$$

$$h_k(x) = 0, k = 1, ..., n \tag{8}$$

onde g_j e h_k são respectivamente j-ésima e k-ésima restrições, e m representa as restrições expressas como inequações e n as restrições expressas como equações.

Os problemas multiobjetivos são caracterizados pelas formas de medição de desempenho (objetivos) que podem ser (in)dependentes e/ou não mensuráveis. Os múltiplos objetivos otimizados quase sempre conflitam, estabelecendo uma posição parcial, ao invés de total, no espaço de busca. Na realidade, a busca de um ótimo global para um POM geral é um problema NP-Completo. A solução perfeita, onde todas as variáveis de decisão satisfazem todas as restrições e a função objetivo alcança o mínimo global, pode não existir (VELDHUIZEN, 1999).

Como os problemas de otimização multiobjetivo tem mais de uma função objetivo, cada uma destas pode ter uma solução individual ótima. Se existe uma diferença suficiente nas soluções ótimas correspondentes a diferentes critérios, as funções objetivo são muitas vezes consideradas como conflitantes entre si. A otimização multiobjetivo com tais funções objetivo dão origem a um conjunto de soluções ótimas, em vez de uma única solução ótima. Essas soluções têm um nome especial - soluções Pareto-ótimas (DEB, 1999).

Estas soluções Pareto-ótimas seguem um conceito importante na determinação de um conjunto de soluções do POM, este conceito é conhecido como Otimalidade de Pareto. O conjunto de soluções de um problema de otimização multiobjetivo consiste em todos os vetores de decisão para os quais não pode haver uma melhora com relação a um objetivo sem que haja uma degradação com relação a pelo menos algum outro objetivo. Estes vetores são conhecidos

como ótimos de Pareto (ZITZLER; THIELE, 1999).

3.2.1 PRINCÍPIO DA OTIMALIDADE DE PARETO

O conceito de ótimo de Pareto foi formulado por Vilfredo Pareto no século XIX (PARETO, 1896), e por si só constitui a origem da pesquisa em otimização multiobjetivo. Para um problema que tem mais do que uma função objetivo ($f_i, i = 1,...,M$ e $M > 1$), quaisquer duas soluções $x^{(1)}$ e $x^{(2)}$ podem apresentar duas possibilidades: uma domina a outra ou nenhuma das soluções domina a outra (DEB, 1999).

A solução $x^{(1)}$ domina a solução $x^{(2)}$, se ambas condições são verdadeiras (denota-se o operador $\prec$ para pior e $\succ$ para melhor):

1. A solução $x^{(1)}$ não é pior que a solução $x^{(2)}$ em todos os objetivos:

$$f_i(x^{(1)}) \not\prec f_i(x^{(2)}), \text{ para todos } i = 1,2,...,M \text{ objetivos} \tag{9}$$

2. A solução $x^{(1)}$ é estritamente melhor que a solução $x^{(2)}$ em pelo menos um objetivo:

$$f_{\bar{\imath}}(x^{(1)}) \succ f_{\bar{\imath}}(x^{(2)}), \text{ para pelo menos um } \bar{\imath} \in \{1,2,...,M\} \tag{10}$$

Se qualquer uma destas condições é violada, a solução $x^{(1)}$ não domina a solução $x^{(2)}$.

O conjunto Pareto-ótimo é formado pelo conjunto de todas as soluções não-dominadas, dentre as soluções factíveis.

As soluções Pareto-ótimas são também denominados de soluções de não-inferioridade, admissíveis, ou eficientes; seus vetores correspondentes são chamados de não-dominados. Estas soluções podem não ter relação clara entre si, além de sua participação no conjunto ótimo de Pareto. Este é o conjunto de todas as soluções cujos vetores correspondentes são não dominados quando comparados a todos os outros vetores. Quando plotados no espaço das funções objetivo, os vetores não-dominados são coletivamente conhecidos como fronteira de Pareto (VELDHUIZEN, 1999).

A fronteira de Pareto é formada pelos pontos no espaço das funções objetivo que correspondem ao conjunto Pareto-ótimo. Não havendo diferença na relevância relativa entre os objetivos a serem atendidos, todos os pontos na fronteira de Pareto são qualitativamente equivalentes, sob a perspectiva de otimização.

A Figura 4 ilustra o conceito de otimalidade de Pareto e a fronteira de Pareto para um problema de otimização multiobjetivo para dois objetivos $f1$ e $f2$. Tomando-se como referência o ponto C, pode-se determinar que ele domina os pontos E e F; é dominado pelos pontos A e B; não domina e nem é dominado pelos pontos D e G. Observando-se agora o ponto B, que pertence à fronteira de Pareto, nota-se que ele domina os pontos C, D, E e F; não domina e nem é dominado pelos pontos A e G; e não é dominado por qualquer outro ponto do espaço objetivo factível, inclusive pelos outros que pertencem à fronteira de Pareto (MACIEL, 2012).

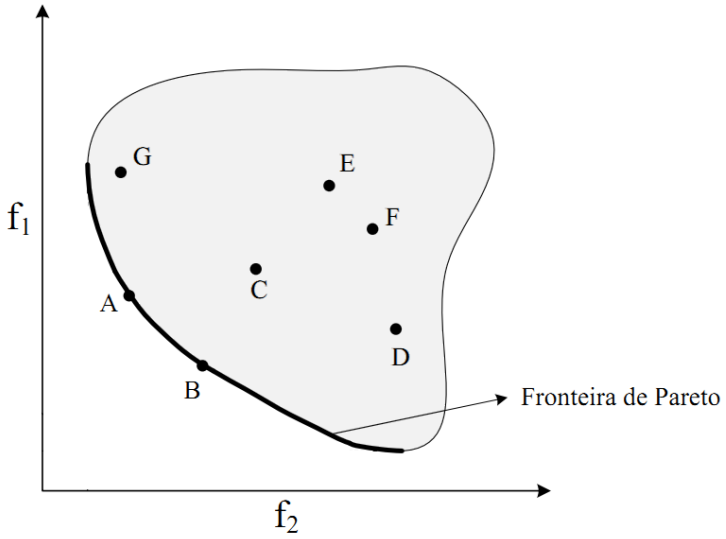


Figura 4: Conceitos de dominância e otimalidade de Pareto para um problema multiobjetivo. Fonte: Adaptado de Maciel (2012)

Conforme Deb (1999) há basicamente dois objetivos que um algoritmo de otimização multi-critério deve atingir:

1. Orientar a busca para a região global Pareto-ótima, e
2. Manter a diversidade da população na fronteira Pareto-ótimo.

A primeira tarefa é um objetivo comum de qualquer algoritmo de otimização. A segunda tarefa é exclusiva para otimização multi-critério. Desde que não há uma solução no conjunto Pareto-ótima que pode ser dita melhor que a outra, o que um algoritmo pode fazer de melhor, é encontrar muitas soluções Pareto-ótimas diferentes quanto possível.

3.3 OTIMIZAÇÃO MULTIOBJETIVO EVOLUCIONÁRIA

Os algoritmos evolucionários (AEs) como descrito na Seção 2.3 são metaheurísticas que usam a seleção natural como principal ferramenta de busca para resolver problemas. O uso de AEs como ferramenta de busca e otimização se tornou muito popular, com um constante desenvolvimento de novos algoritmos e aplicações. Uma das áreas de pesquisa em que os AEs se tornaram mais populares é a otimização multiobjetivo. Em problemas de otimização multiobjetivo como descrito na Seção 3.2, tem-se duas ou mais funções objetivo a serem otimizadas, ao mesmo tempo, em vez de ter apenas uma. Desta forma, não existe uma única solução para os problemas de otimização multiobjetivo, mas em vez disso, o foco principal é encontrar todas as boas soluções disponíveis (conjunto ótimo de Pareto).

A principal motivação para o uso de AEs para resolver problemas de otimização multiobjetivo é porque os AEs lidam simultaneamente com um conjunto de soluções possíveis (chamada de população), as quais permitem encontrar várias soluções do conjunto ótimo de Pareto em uma única execução do algoritmo, em vez de ter de realizar uma série de operações isoladas, como no caso das técnicas tradicionais de programação matemática (MIETTINEN, 1998). Além disso, AEs são menos suscetíveis à forma ou continuidade da fronteira de Pareto, por exemplo, eles podem facilmente lidar com fronteiras de Pareto descontínuas e côncavas. Vale destacar que estas duas questões são uma real preocupação para as técnicas de programação matemática (COELLO et al., 2002).

3.3.1 ALGORITMOS EVOLUCIONÁRIOS MULTIOBJETIVOS

A primeira implementação real do que hoje é chamado de um algoritmo evolucionário multiobjetivo (ou AEMOs) foi *Schaffer's Vector Evaluation Genetic Algorithm* (VEGA), que foi introduzido em meados da década de 1980, tendo como objetivo resolver problemas no aprendizado de máquina (SCHAFFER, 1984, 1985). Desde então, uma grande variedade de algoritmos têm sido propostos na literatura. Conforme (COELLO, 2005) pode-se dividir de uma forma superficial os AEMOs nos seguintes tipos:

- Funções Agregadas;
- Abordagens de base populacional;
- Abordagens baseadas em Pareto.

3.3.1.1 FUNÇÕES AGREGADAS

Talvez a abordagem mais simples e direta de lidar com múltiplos objetivos em qualquer técnica é combinar todos os objetivos em um único objetivo utilizando ou uma adição, multiplicação ou qualquer outra combinação de operações aritméticas. Estas técnicas são geralmente conhecidas como "funções agregadas" porque elas combinam (ou "agregam") todos os objetivos do problema em um único objetivo. De fato, as abordagens de agregação são os métodos mais antigos de programação matemática para otimização multiobjetivo (COELLO, 2005). Um exemplo desta abordagem é a soma linear de pesos da seguinte forma:

$$min \sum_{i=1}^{k} w_i \cdot ft_i(x) \tag{11}$$

onde $w_i \geq 0$ são os coeficientes de ponderação que representam a importância relativa das k funções objetivo do problema. Geralmente assume-se que:

$$\sum_{i=1}^{k} w_i = 1 \tag{12}$$

As funções agregadas podem ser lineares (como no exemplo anterior) ou não-linear. Ambos os tipos de funções agregadas foram utilizadas com algoritmos evolucionários em várias estudos com relativo sucesso (COELLO, 2005). Alguns exemplos que utilizam essa abordagem: método do critério global (COELLO; CHRISTIANSEN, 2000), método das ponderações (COELLO, 1999) e método ε-*Constraint* (COELLO et al., 2007).

As funções agregadas lineares tem uma limitação bem conhecida, isto é, elas não podem gerar porções não-convexas da fronteira de Pareto independentemente da combinação de peso utilizada (DAS; DENNIS, 1997). No entanto, as funções agregadas não-lineares não estão necessariamente sujeitas a tal limitação (COELLO et al., 2002). De fato, até mesmo as funções agregadas lineares podem ser habilmente definidas de tal forma que fronteiras de Pareto côncavas possam ser geradas (JIN et al., 2001).

3.3.1.2 ABORDAGENS DE BASE POPULACIONAL

Nesta técnica, a população de um AE é usada para diversificar a busca, mas o conceito de dominância de Pareto não é diretamente incorporada no processo de seleção. O exemplo clássico desse tipo de abordagem é o *Vector Evaluated Genetic Algorithm* (VEGA), proposto por Schaffer (SCHAFFER, 1985). O VEGA consiste basicamente de um algoritmo

genético simples, com um mecanismo de seleção modificado. Em cada geração, um número de sub-populações são geradas através da seleção proporcional de acordo com cada função objetivo. Assim, para um problema com k objetivos, tem-se k sub-populações de tamanho M/k (assumindo que M é o tamanho total da população). Estas sub-populações são misturadas para obter uma nova população de tamanho M, nas quais são aplicados os operadores de cruzamento e mutação do algoritmo genético (AG).

O VEGA tem alguns problemas, dos quais o mais sério é que seu sistema de seleção se opõe ao conceito de dominância de Pareto. Se, por exemplo, há um indivíduo que codifica uma boa solução para todos os objetivos, mas não é o melhor em qualquer um deles, ele será descartado. Note, no entanto, que tal indivíduo deveria realmente ser preservado porque ele codifica uma solução Pareto-ótimo. Schaffer sugeriu algumas heurísticas para lidar com este problema. Por exemplo, usar uma abordagem de seleção heurística preferencial para os indivíduos não-dominados em cada geração, para proteger os indivíduos que codificam soluções Pareto-ótimas, mas não são os melhores em qualquer uma das funções objetivo. Além disso, cruzamento entre as "espécies" poderiam ser incentivados pela adição de alguma seleção heurística de companheiro em vez de usar a seleção aleatória de companheiro do AG tradicional. No entanto, o fato de que a dominância de Pareto não é diretamente incorporada ao processo de seleção do algoritmo continua a ser a sua principal desvantagem.

Um aspecto interessante do VEGA é que, apesar das suas desvantagens ele permanece em uso corrente por alguns pesquisadores, principalmente porque é apropriado para problemas em que se deseja que o processo de seleção seja tendencioso e no qual tenha que lidar com um grande número de objetivos, por exemplo, quando se quer lidar com restrições como objetivos em uma otimização de um único objetivo (COELLO, 2000b).

3.3.1.3 ABORDAGENS BASEADAS EM PARETO

Tomando como base as principais desvantagens do VEGA, Goldberg discutiu nas páginas 199-201 do seu livro sobre algoritmos genéticos (GOLDBERG, 1989) uma forma de solucionar os problemas multiobjetivos. O procedimento proposto por ele consiste em um esquema de seleção com base no conceito de otimalidade de Pareto. Goldberg não só sugeriu o que se tornaria o padrão AEMO por vários anos, mas também indicou que o "ruído" estocástico faria tais algoritmos inúteis a menos que fosse adotado um mecanismo especial para bloquear a convergência. O *Niching* ou *fitness sharing* (DEB; GOLDBERG, 1989) foi sugerido por Goldberg como uma forma de manter a diversidade e evitar a convergência do algoritmo genético (AG) para uma única solução.

As abordagens baseadas em Pareto podem ser historicamente divididas em duas gerações (COELLO, 2005). A primeira geração é caracterizada pelo uso do *fitness sharing* e *niching* combinado com ranking de Pareto (como definido por Goldberg ou adotando uma ligeira variação). Os algoritmos mais representativos de primeira geração são os seguintes:

1. ***Non-dominated Sorting Genetic Algorithm*** (NSGA): este algoritmo foi proposto por Srinivas e Deb (SRINIVAS; DEB, 1994). A abordagem é baseada em várias camadas de classificações dos indivíduos, conforme sugerido por Goldberg (GOLDBERG, 1989). Antes da seleção ser realizada, a população é classificada com base na não-dominância: todos os indivíduos não-dominados são classificados em uma categoria ou *fronts* (com um valor de *fitness*, o qual é proporcional ao tamanho da população, para fornecer um potencial reprodutivo igual para estes indivíduos). Para manter a diversidade da população, esses indivíduos classificados compartilham seus valores de *fitness* segundo suas distâncias euclidianas. Então este grupo de indivíduos classificados é ignorado e outra camada de indivíduos não-dominados é considerada. O processo continua até que toda a população de indivíduos seja classificada. Desde que os indivíduos no primeiro *front* tenham o valor máximo de *fitness*, eles sempre são mais copiados do que o resto da população. Isso permite a busca para regiões não-dominadas, e resulta na convergência da população em direção a essas regiões. O valor do *sharing* ajuda a distribuir a população sobre esta região (ou seja, a fronteira de Pareto do problema). O NSGA permite lidar com qualquer número de objetivos, para problemas de minimização ou maximização. Entretanto o NSGA é muito sensível ao parâmetro de configuração σshare, que tem grande influência no desempenho da busca (DEB, 1999).

2. ***Niched-Pareto Genetic Algorithm*** (NPGA): proposto pelo Horn et al. (HORN et al., 1994). O NPGA usa um esquema de seleção de torneio baseado em dominância de Pareto. A ideia básica do algoritmo é a seguinte: dois indivíduos são escolhidos aleatoriamente e comparados com um subconjunto de toda a população (tipicamente, cerca de 10% da população). Se um deles é dominado (pelos indivíduos escolhidos ao acaso da população) e o outro não é, então o indivíduo não-dominado vence. Quando ambos os competidores ou são dominados ou não-dominados (ou seja, há um empate), o resultado do torneio é decidido através do valor do *sharing fitness*. De acordo com Coello (COELLO, 1999), essa técnica é rápida e gera uma boa fronteira não dominante. Entretanto, requer um bom valor σshare e um bom valor para o tamanho do conjunto comparação, o que dificulta seu uso na prática.

3. ***Multiobjective Genetic Algorithm*** (MOGA): Proposto por Fonseca e

Fleming (FONSECA; FLEMING, 1993). No MOGA, a classificação de um determinado indivíduo corresponde ao número de cromossomos na população atual pelo qual é dominado. Considere-se, por exemplo, um indivíduo x_i na geração t, o qual é dominado por $p_i^{(t)}$ indivíduos na geração atual. A classificação de um indivíduo é dada por (FONSECA; FLEMING, 1993):

$$rank(x_i,t) = 1 + p_i^{(t)} \tag{13}$$

Todos os indivíduos não-dominados são atribuídos a classificação 1 e os outros indivíduos receberam o valor 1 somado ao número de indivíduos que o dominam. Com isto todos os indivíduos são analisados quanto a sua dominância dentre todos da população. A atribuição do valor do *fitness* para cada indivíduo é feita da seguinte maneira (FONSECA; FLEMING, 1993):

a) Ordenar a população de acordo com a classificação.

b) Atribuir o *fitness* aos indivíduos pela interpolação do melhor (classificação 1) para o pior (classificação $n \leq M$) na forma proposta por Goldberg (1989), de acordo com alguma função, geralmente linear, mas não necessariamente.

c) Média do *fitness* dos indivíduos com a mesma classificação, de modo que todos eles sejam amostrados com a mesma taxa. Este procedimento mantém o *fitness* global da população constante, mantendo o potencial da seleção apropriado, tal como definido pela função utilizada.

A segunda geração dos AEMOs nasceu com a introdução do conceito de elitismo. No contexto da otimização multiobjetivo, elitismo geralmente (embora não necessariamente) refere-se à utilização de uma população externa (também chamada de população secundária) para guardar os indivíduos não-dominados. No entanto, o uso deste arquivo externo levanta diversas questões:

- Como é que o arquivo externo interage com a população principal?
- O que fazer quando o arquivo externo está cheio?
- Deve-se impor critérios adicionais para entrar no arquivo, em vez de apenas usar a dominância de Pareto?

Note que elitismo também pode ser introduzido através da utilização de uma seleção em que os pais competem com os seus filhos, e aqueles que são não-dominados e possivelmente

cumprem alguns critérios adicionais, tais como proporcionar uma melhor distribuição de soluções são selecionados para a geração seguinte.

Os AEMOs mais representativos de segunda geração são os seguintes (COELLO, 2005):

1. ***Strength Pareto Evolutionary Algorithm*** (SPEA): Este algoritmo foi desenvolvido por Zitzler e Thiele (ZITZLER; THIELE, 1999). Esta abordagem foi concebida como uma forma de integrar diferentes AEMOs. SPEA usa um arquivo que contem soluções não-dominadas previamente encontradas (chamado conjunto não-dominado externo). A cada geração, os indivíduos não-dominados são copiados para o conjunto não-dominado externo. Para cada indivíduo neste conjunto externo, um valor de *strength* (resistência) é computado. Este parâmetro é semelhante ao valor de classificação de MOGA, uma vez que é proporcional ao número de soluções que um determinado indivíduo domina.

   No SPEA, o *fitness* de cada indivíduo da população atual é calculado de acordo com as resistências (*strength*) de todas as soluções não-dominadas externas que o domina. Além disso, a técnica de agrupamento conhecida como *average linkage method* é utilizada para manter a diversidade.

2. ***Strength Pareto Evolutionary Algorithm 2*** (SPEA2): Esta abordagem tem três diferenças principais em relação a primeira versão (ZITZLER et al., 2002): (1) incorpora uma estratégia de atribuição de *fitness fine-grained* que leva em consideração para cada indivíduo o número de indivíduos que este domina e o número de indivíduos pelos quais é dominado; usa uma técnica de estimativa de densidade do vizinho mais próximo que direciona a busca de forma mais eficiente, e (3) tem um método avançado de truncamento de arquivo que garante a preservação de soluções da fronteira.

3. ***Pareto Archived Evolution Strategy*** (PAES): Este algoritmo foi desenvolvido por Knowles e Corne (KNOWLES; CORNE, 2000). O PAES consiste de uma estratégia de evolução $(1 + 1)$ (isto é, um único pai que gera uma única descendência) em combinação com um arquivo de histórico que registra algumas das soluções não-dominadas encontradas previamente. Este arquivo é usado como um conjunto de referência para o qual cada indivíduo "mutado" está sendo comparado. Um aspecto interessante deste algoritmo é o procedimento utilizado para manter a diversidade, que consiste num procedimento de aglomeração que divide o espaço objetivo de uma forma recursiva. Cada solução é colocada em uma certa localização em um tipo de grade com base nos valores dos seus objetivos (que são utilizados como as "coordenadas"

ou “localização geográfica”). Um mapa desta grade é mantido, indicando o número de soluções que residem em cada local da grade. Uma vez que o processo é adaptativo, não há parâmetros adicionais (exceto para o número de divisões do espaço objetivo).

4. ***Non-dominated Sorting Genetic Algorithm II*** (NSGA-II): descrição em detalhes na subseção 3.1.5.

5. ***Niched Pareto Genetic Algorithm 2*** (NPGA 2): Erickson et al. (ERICKSON et al., 2001) propôs uma versão revisada do NPGA (HORN et al., 1994) chamando de NPGA 2. Este algoritmo utiliza uma classificação de Pareto mas mantém o torneio por seleção (resolvendo os laços através do *sharing fitness* como no NPGA original). Neste caso, nenhuma memória externa é usada e o mecanismo elitista é semelhante ao adotado pelo NSGA-II. As contagens de nicho no NPGA 2 são calculados usando indivíduos da próxima geração parcialmente completa, em vez de utilizar a geração atual. Essa técnica atualiza o *sharing fitness* continuamente, e foi proposta por (OEI et al., 1991).

6. ***Pareto Envelope-based Selection Algorithm*** (PESA): Este algoritmo foi proposto por Corne et al. (CORNE et al., 2000). Esta abordagem utiliza uma pequena população interna e uma população maior externa (ou secundária). O PESA utiliza a mesma divisão de espaço hiper-grade de fenótipo (ou seja, a função objetivo) adotada pelo PAES para manter a diversidade. No entanto, o seu mecanismo de seleção é baseada na medida de aglomeração utilizado pelo hiper-grade mencionado anteriormente. Esta mesma medida de aglomeração é usada para decidir quais as soluções que serão introduzidas na população externa (ou seja, o arquivo de vetores não-dominados encontrados ao longo do processo evolutivo). Portanto, no PESA, a memória externa desempenha um papel crucial no algoritmo pois determina não só o esquema de diversidade, mas também a seleção efetuada pelo método. Há também uma versão revisada deste algoritmo, chamado PESA-II (CORNE et al., 2001). Este algoritmo é idêntico ao PESA, exceto pelo fato de que a seleção baseada em região é utilizada neste caso. Na seleção baseada em região, a unidade de seleção é um *hyperbox* em vez de um indivíduo. O procedimento consiste na seleção (usando qualquer uma das técnicas de seleção tradicionais (GOLDBERG; DEB, 1991) de uma *hyperbox* e então uma seleção aleatória de um indivíduo dentro dessa *hyperbox*. A principal motivação desta abordagem é reduzir os custos computacionais associados aos AEMOs tradicionais (ou seja, aqueles com base na classificação de Pareto).

7. **Algoritmo Micro-Genético**: Esta abordagem foi desenvolvida por Coello e Pulido (COELLO; PULIDO, 2001a, 2001b). Um algoritmo micro-genético é um AG

com uma população pequena e um processo de reinicialização. Descrição em detalhes na subseção 3.1.2.

3.4 AVALIAÇÃO E TESTE DE DESEMPENHO DOS AEMOS

O desenvolvimento de uma grande variedade de AEs tem resultado em inúmeras comparações com o objetivo de demonstrar a superioridade geral de um algoritmo sobre seus pares. Este processo de comparação é baseado em dois recursos: um grande conjunto de problemas de teste multiobjetivo de fácil implementação, artificialmente construídos, e uma vasta gama de medidas com as quais os resultados podem ser comparados. Considerando o cenário típico de comparação dos AEs (HUBAND et al., 2006):

1. Selecionar os AEs para comparar.
2. Escolher um conjunto de problemas de teste existentes ou criar novos.
3. Escolha um conjunto de medidas para comparar os resultados obtidos pelos AEs.
4. Obter resultados para cada AE em cada problema teste.
5. Gerar medidas para os resultados e comparar os dados.
6. Tirar conclusões.

A fim de tirar conclusões precisas, é necessário que os problemas de testes utilizados sejam bem compreendidos, que as medidas sejam adequadas, e que sejam empregados métodos estatísticos adequados (HUBAND et al., 2006).

A comparação entre os AEMOs requer o uso de conjuntos de teste comuns como *benchmarks* disponíveis na literatura. De acordo com Huband et al. (2006), muitos conjuntos de teste não tinham sido cuidadosamente analisados, o que tornava difícil obter boas conclusões sobre as vantagens e limitações de cada algoritmo.

Os conjuntos de testes devem ser projetados para avaliar condições específicas dos algoritmos, logo devem possuir características adequadas para esse fim.

Seis características gerais apresentadas por Deb et al. (2002b) devem fazer parte de um conjunto de problemas de teste para avaliar adequadamente um AEMO:

1. Os problemas de teste devem introduzir “obstáculos” controláveis para convergir à verdadeira fronteira de Pareto-ótima e também para encontrar um conjunto de soluções

Pareto-ótimas amplamente distribuído. Isso ocorre porque a convergência próxima à fronteira de Pareto-ótima e a manutenção de um conjunto diversificado de soluções são dois objetivos básicos em otimização multiobjetivo.

2. Os problemas de teste devem ser escaláveis para ter qualquer número de variáveis de decisão. Isto porque muitos problemas do mundo real geralmente envolvem um grande número de variáveis de decisão. Na prática, os problemas de teste envolvendo um grande número de variáveis podem ser incluídos em um conjunto de teste.

3. Os problemas de teste devem ser escaláveis para ter qualquer número de objetivos. Embora na maioria dos problemas do mundo real, o número de objetivos pode ser reduzido para quatro ou cinco objetivos (DEB et al., 2002b), problemas de testes envolvendo 15 a 20 objetivos podem ser incluídos no conjunto de testes.

4. Os problemas de teste devem ser simples de construir. Isto pode não ser uma questão essencial para a construção de problemas de testes, mas se as características desejadas podem ser incorporadas em problemas de teste com um procedimento de construção simples, é sempre desejável.

5. A fronteira Pareto-ótima resultante (contínua ou discreta) deve ser fácil de compreender, e sua forma e localização deve ser exatamente conhecida. Os valores das variáveis de decisão correspondentes também devem ser fácil de encontrar.

6. Para tornar os problemas de teste uteis na prática, eles devem apresentar dificuldades semelhantes às presentes na maioria dos problemas do mundo real.

É importante mencionar que a característica 5) pode não ser de muita importância para a comparação de dois ou mais AEMOs. Mas, para avaliar o desempenho de um AEMO na convergência e manutenção da diversidade, o conhecimento exato do conjunto Pareto-ótimo é essencial.

3.4.1 MÉTRICAS

A definição de métricas apropriadas é muito importante para ser capaz de validar um algoritmo. No entanto, quando se trata de problemas de otimização multiobjetivo, há várias razões pelas quais a avaliação qualitativa de resultados torna-se difícil. O primeiro problema é a geração de várias soluções, em vez de apenas uma (o objetivo é gerar o maior número possível de elementos no conjunto Pareto-ótimo). O segundo problema é que a natureza estocástica dos algoritmos evolucionários torna necessário realizar várias execuções para avaliar

o seu desempenho. Assim, os resultados devem ser validados usando ferramentas de análise estatística. Finalmente, é possível estar interessado em medir coisas diferentes. Por exemplo, pode-se estar interessado em ter um algoritmo robusto que se aproxima da fronteira global de Pareto de um problema de forma consistente, ao invés de um algoritmo que converge para a fronteira global de Pareto mas apenas ocasionalmente. Além disso, pode-se estar interessado em analisar o comportamento de um algoritmo evolucionário durante o processo evolutivo, tentando estabelecer suas capacidades para manter a diversidade e convergir progressivamente para um conjunto de soluções próximas da fronteira global de Pareto de um problema.

Normalmente algumas questões devem ser consideradas para projetar uma boa métrica neste domínio (DEB; KALYANMOY, 2001):

1. Minimizar a distância da fronteira de Pareto produzida pelo algoritmo em avaliação e a fronteira global de Pareto (assumindo que a sua localização é conhecida).
2. Maximizar a propagação de soluções encontradas, para que se possa ter uma distribuição suave e uniforme de vetores quanto possível.
3. Maximizar o número de elementos do conjunto Pareto-ótimo.

A pesquisa produzida nos últimos anos tem incluído uma ampla variedade de métricas que avalia o desempenho de um AEMO em um dos três aspectos anteriormente indicados. Alguns exemplos são os seguintes:

1. Taxa de Erro (*Error Ratio*) (ER): Esta métrica foi proposta por Veldhuizen (1999) para indicar a percentagem de soluções (a partir dos vetores não-dominados encontrados até agora) que não são membros do verdadeiro conjunto Pareto-ótimo:

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{14}$$

   onde n é o número de soluções no atual conjunto de soluções não-dominados disponíveis; $e_i = 0$ se a solução i é um membro do conjunto Pareto-ótimo, e $e_i = 1$ caso contrário. Em seguida, deverá ser claro que $ER = 0$ indica um comportamento ideal, uma vez que significaria que todas as soluções geradas pelo AEMO avaliado pertencem ao conjunto de Pareto-ótimo do problema. Esta métrica aborda a terceira questão da lista mencionada anteriormente.

2. Distância da geração (*Generational Distance*) (GD - γ): O conceito de distância de geração foi introduzida por Veldhuizen e Lamont (1998) como uma maneira de estimar

quão longe os elementos no conjunto de soluções não-dominados encontrados estão daqueles no conjunto de Pareto-ótimo e é definido como:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{15}$$

onde n é o número de soluções no conjunto de soluções não-dominadas encontradas até agora e d_i é a distância Euclidiana (medido no espaço objetivo) entre cada uma delas e o membro mais próximo do conjunto de Pareto-ótimo. Deve ficar claro que um valor de $GD = 0$ indica que todos os elementos gerados estão no conjunto de Pareto-ótimo. Portanto, qualquer outro valor irá indicar quão "longe" estão da fronteira global de Pareto do problema. Esta métrica soluciona a primeira questão da lista mencionada anteriormente.

3. Espaçamento/Distribuição (*Spread*) (SP - Δ): Esta métrica deseja medir a distribuição, na fronteira de Pareto, das soluções não-dominados encontrados. Uma vez que o "início" e "fim" da fronteira de Pareto são conhecidos, uma métrica definida adequadamente julga quão bem as soluções dessa fronteira estão distribuídas. Schott (1995) propôs como uma métrica medir a variação de intervalo (distância) de soluções vizinhas nas soluções não-dominados encontrados. Esta métrica é definida como:

$$\Delta = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d-d_i)^2} \tag{16}$$

onde $d_i = min(|f_1^i(x) - f_1^j(x)| + |f_2^i(x) - f_2^j(x)|), i, j = 1, ..., n$, d é a média de todos d_i, e n é o número de soluções não-dominadas encontrados até agora. Um valor de zero para esta métrica indica que todos os membros da fronteira de Pareto atualmente disponíveis são espaçados de forma equidistante. Esta métrica aborda a segunda questão da lista mencionada anteriormente.

3.5 CONSIDERAÇÕES

Com base em toda fundamentação apresentada neste capítulo e considerando-se dois aspectos importantes dos AEMOs: i) A utilização de pequenas populações reduz a carga computacional, mas não é suficiente para assegurar uma boa distribuição na fronteira de Pareto; ii) Uma estratégia de arquivamento para salvar as soluções não-dominadas associada a um processo de reinicialização pode impedir uma convergência prematura para regiões específicas

da fronteira de Pareto, e considerando também que iii) o SFLA é uma boa alternativa para resolver problemas de programação de grande porte com objetivos conflitantes como sugerido pela literatura descrito na seção 3.1.3; propõe-se neste trabalho uma versão modificada do SFLA para resolver a otimização multiobjetivo de uma rede de dutos do mundo real. O algoritmo proposto tem uma população pequena e utiliza uma estratégia de arquivamento combinadas com as vantagens de uma rápida convergência do SFLA.

Além disso, propõe-se um novo procedimento para reiniciar a população usando duas memórias auxiliares (um curta e uma longo prazo) para armazenar soluções não-dominadas encontradas durante a evolução da população. O objetivo é representar e recuperar toda a fronteira de Pareto para o problema modelado. Esta nova abordagem é chamada de *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA) e é apresentada no próximo capitulo.

Também no próximo capítulo a eficácia do MSFLPA para construir/recobrir a fronteira de Pareto é demonstrado resolvendo cinco problemas multiobjetivo bem conhecidos da literatura. Esta eficácia é avaliada por duas métricas e em comparação com dois famosos algoritmos de evolução multiobjetivo, NSGA-II (DEB et al., 2002a) e SPEA2 (ZITZLER et al., 2001). Além disso experimentos numéricos e um estudo de caso para calcular a programação de curto prazo de um cenário também são realizados para verificar o desempenho da abordagem proposta MSFLPA. Seu desempenho também é comparado com NSGA-II por (DEB et al., 2002a) e o Algoritmo micro Genético (μAG) por (RIBAS et al., 2013) que é adaptado para modelar a rede de dutos estudada.

4 PROPOSTA UTILIZANDO O ALGORITMO SHUFFLED FROG-LEAPING ALGORITHM (SFLA)

O algoritmo proposto tem uma população pequena e utiliza uma estratégia de arquivamento combinada com a vantagem de uma rápida convergência de SFLA. Além disso, propõe-se um novo procedimento para reiniciar a população usando duas memórias auxiliares (uma de curto e outra de longo prazo) para armazenar soluções não dominadas encontradas durante a evolução da população. O objetivo é representar e recuperar toda a fronteira de Pareto para um problema modelado. Esta nova abordagem é chamada de *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA), e este capítulo descreve todos os detalhes dessa nova abordagem, assim como os testes numéricos para validação do algoritmo.

4.1 *MODIFIED SHUFFLED FROG-LEAPING PARETO APPROACH*

Como descrito no capitulo anterior na seção 3.1.3, o algoritmo SFLA utiliza um mecanismo de evolução em grupo, em que os indivíduos (sapos) podem evoluir para direções diferentes, o que o torna particularmente adequado para problemas de otimização multiobjetivo. Com base nisto, este trabalho propõe uma nova abordagem para o uso de SFLA modificado para problemas multiobjetivos. Esta nova abordagem, chamada de *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA), tem a mesma estrutura do SFLA original; e utiliza algumas estratégias para facilitar a busca de diferentes pontos na fronteira Pareto-ótima ao longo das gerações. O fluxograma correspondente a MSFLPA pode ser visto na Figura 5 e o SFLA padrão é apresentado no capitulo anterior.

Para reduzir o tempo computacional, a nova abordagem proposta utiliza uma população pequena associada a um processo de reinicialização dessa população a partir de uma memória externa que armazena as soluções não dominadas encontradas ao longo das gerações do método. O uso dessas estratégias de reinicialização da população após poucas gerações, tem o objetivo de assegurar a diversidade da população e evitar convergência prematura (COELLO; PULIDO, 2001a).

O MSFLPA incorpora uma estratégia de arquivamento (*archiving strategy*) com base

em duas memórias auxiliares para manter as soluções não dominadas e melhorar o método de ranqueamento da população. A descrição detalhada do novo algoritmo, o MSFLPA, é apresentada a seguir.

- A primeira memória é denominada conjunto temporário de soluções não dominadas e é usada para guardar as soluções não dominadas obtidas durante a evolução de um determinado número de gerações;
- A segunda é uma memória externa, denominada conjunto final de soluções não dominadas, que é utilizada para guardar o conjunto final de soluções de Pareto encontradas ao longo de todas as evoluções a partir da reinicialização da população. Nesta memória externa, as soluções armazenadas correspondem ao conjunto de todas as soluções não dominadas quando comparados a todas as outras soluções previamente armazenados.

Em AEMOs eficientes, a estratégia de arquivamento é um importante mecanismo para manter um número limitado de soluções não-dominadas. Portanto, uma estratégia de classificação das soluções descrita em (RAHIMI-VAHED; MIRZAEI, 2007; LI et al., 2010) foi usado no MSFLPA. No início, todos os sapos não-dominados da população atual são encontrados e classificados em ordem decrescente de acordo com o valor da distância de aglomeração (*crowding distance*). Neste trabalho, a distância de aglomeração é utilizada como uma medida da densidade ou seja, é dada pela diferença absoluta dos valores das funções de duas soluções adjacentes. Supondo que o problema multiobjetivo tenha r objetivos, a distância de aglomeração dos sapos não-dominados é calculada da seguinte forma:

$$P[i].dist = \sum_{k=1}^{r} |P[i+1].f_k - P[i-1].f_k| \tag{17}$$

onde $P[i].dist$ significa a distância de aglomeração do $i-$ésimo sapo (solução) não-dominado, $P[i+1].f_k$ e $P[i-1].f_k$ são os valores da função objetivo de dois sapos (soluções) não-dominados adjacentes.

As soluções não-dominadas com a maior distância de aglomeração são armazenadas na primeira memória. Todas as outras soluções são descartadas.

O novo algoritmo MSFLPA é resumido nas seguintes etapas:

Passo 1. A população inicial de F sapos (soluções) é criada aleatoriamente. Para problemas de dimensão N, cada sapo i é representado por N variáveis como $X_i = (x_{i1}, x_{i2}, ..., x_{iN})$.

Passo 2. O *fitness* de cada solução (sapo) é avaliado e os sapos então são classificados em ordem decrescente de acordo com o seu valor de *fitness*.

Passo 3. Toda a população é evoluída usando o SFLA modificado de acordo com as etapas descritas na Seção 3.1.3.

Passo 4. Se o algoritmo atinge um número especifico de gerações, sem mudanças na evolução, ele vai para o Passo 5, se não vai para o Passo 2.

Passo 5. Encontrar soluções não-dominadas na população F.

Passo 6. Classificar as soluções não-dominadas encontradas em ordem decrescente de acordo com o valor de distância de aglomeração.

Passo 6. Armazenar as soluções não-dominadas com as maiores distâncias de aglomeração no conjunto temporário.

Passo 8. Se o conjunto final estiver vazio, as soluções não-dominadas do conjunto temporário são automaticamente copiadas para o conjunto final, senão faz-se uma comparação entre as soluções não-dominadas do conjunto temporário e as do conjunto final usando o conceito de otimalidade de Pareto e separando-se as soluções não dominadas.

Passo 9. Atualizar o conjunto final com as novas soluções de Pareto encontrados.

Passo 10. Voltar para passo 1.

Passo 11. Repita os passos 1 a 10 para um número especificado de gerações.

A ideia geral dessa abordagem é que a partir da reinicialização da população depois de um número de gerações, o espaço de busca explorado seja maior, e alcance ao final das iterações, um conjunto final de soluções não dominadas que possa representar e reconstruir uma grande parte da fronteira de Pareto.

4.2 AVALIAÇÃO E TESTE DE DESEMPENHO DO MSFLPA

A fim de validar o desempenho da abordagem proposta o MSFLPA, cinco funções de teste bem conhecidas propostas por Zitzler-Deb-Thiele (ZDT) (DEB, 1999; ZITZLER et al., 2000) são utilizadas nesta seção. A seguir, serão descritas as funções e os resultados obtidos.

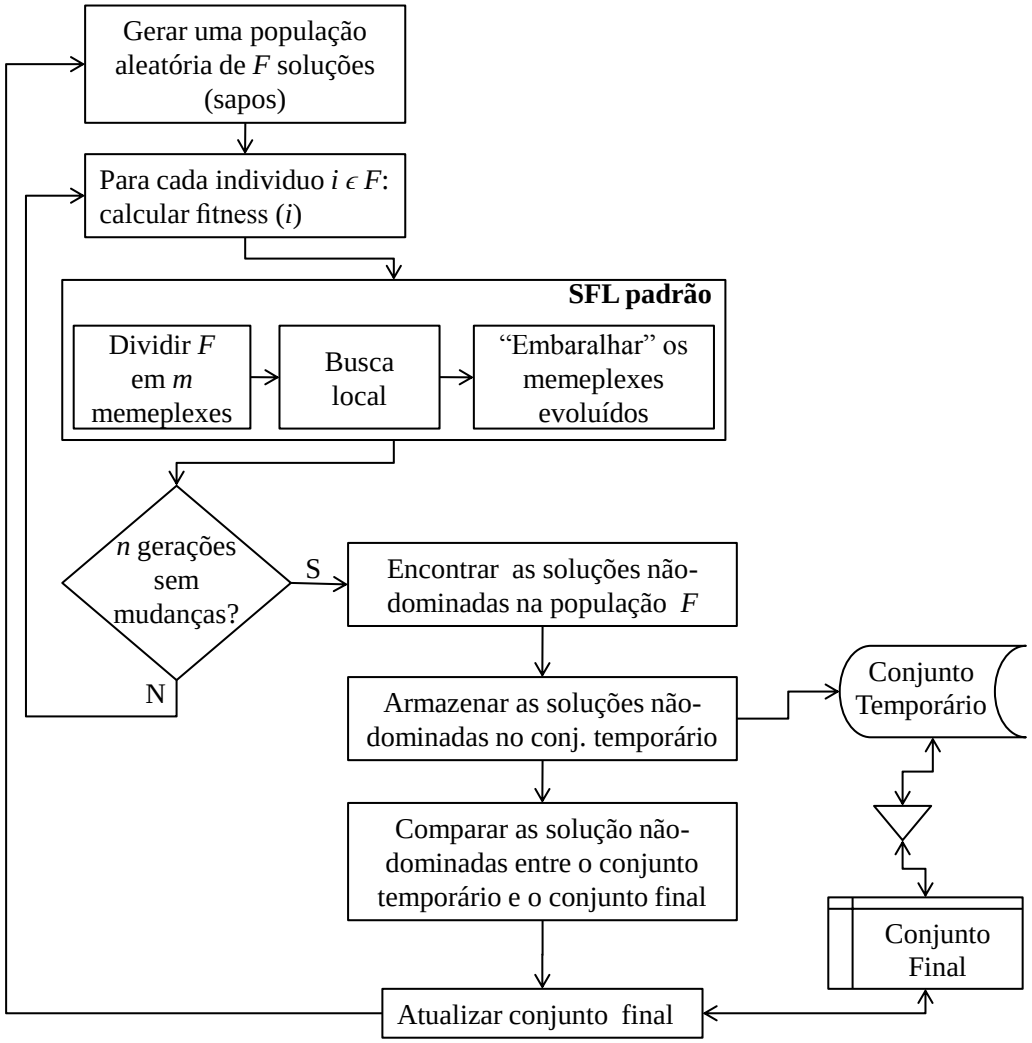


Figura 5: Fluxograma simplificado do MSFLPA.

4.2.1 FUNÇÕES TESTES - ZDT

O conjunto de teste ZDT possui 6 funções testes, mas neste trabalho serão utilizados os problemas ZDT1 a ZDT4 e ZDT6. O problema ZDT5 não foi utilizado neste trabalho porque ele utiliza representação binária, para o qual os modelos implementados não são adequados. Cada uma das funções de teste definidas a seguir, é estruturada da mesma forma, e consiste de três funções:

$$\begin{aligned} \text{Minimizar} \quad & T(x) = (f_1(x_1), f_2(x_2)) \\ \text{sujeito a} \quad & f_2(x_2) = g(x_2,...,x_m).h(f_1(x_1), g(x_2,...,x_m)) \\ \text{onde} \quad & x = (x_1,...,x_m) \end{aligned} \tag{18}$$

Todos as funções ZDT trabalham com três funções f_1, g e h que podem ser definidas com vários níveis de complexidade para criar problemas de teste de otimização de dois objetivos complexos. A seguir, resumem-se as propriedades de um problema de otimização com dois objetivos devido a cada uma das funções acima:

1. A função f_1 testa a capacidade dos AEMOs de achar diversas soluções Pareto-ótimas. Assim, esta função testa a capacidade de um algoritmo de lidar com dificuldades ao longo da fronteira de Pareto-ótima.

2. A função g testa a capacidade dos AEMOs de convergir para a verdadeira (ou global) fronteira Pareto-ótima. Assim, esta função testa a capacidade de um algoritmo de lidar com as dificuldades "laterais"para a fronteira Pareto-ótima.

3. A função h testa a capacidade dos AEMOs para resolver os problemas multiobjetivos convexo, não-convexo, ou fronteiras Pareto-ótimas descontínuas. Assim, esta função testa a capacidade de um algoritmo de lidar com diferentes formas da fronteira Pareto-ótima.

4.2.2 TESTES NUMÉRICOS

Para todas as funções de teste, os parâmetros do MSFLPA são definidos da seguinte forma: população $|F| = 40$, número de memeplexos $m = 4$, iterações evolutivas $ei = 10$, número de gerações $g = 250$ e fator de aceleração $C = 2$.

4.2.2.1 FUNÇÃO DE TESTE ZDT1

Esta função tem uma fronteira Pareto-ótima convexa:

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2,...,x_m) &= 1 + 9.\sum_{i=2}^{m} x_i/(m-1) \\ h(f_1,g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{20}$$

onde $m = 30$ e $x_i \in [0,1]$. A fronteira Pareto-ótima é formada com $g(x) = 1$.

A Figura 6 ilustra os resultados do cálculo da função ZDT1 fornecidos pelo MSFLPA, onde a fronteira Pareto-ótima da função de teste é apresentada como uma linha contínua azul e as soluções não-dominadas encontradas pelo MSFLPA são os pontos pretos nos gráfico.

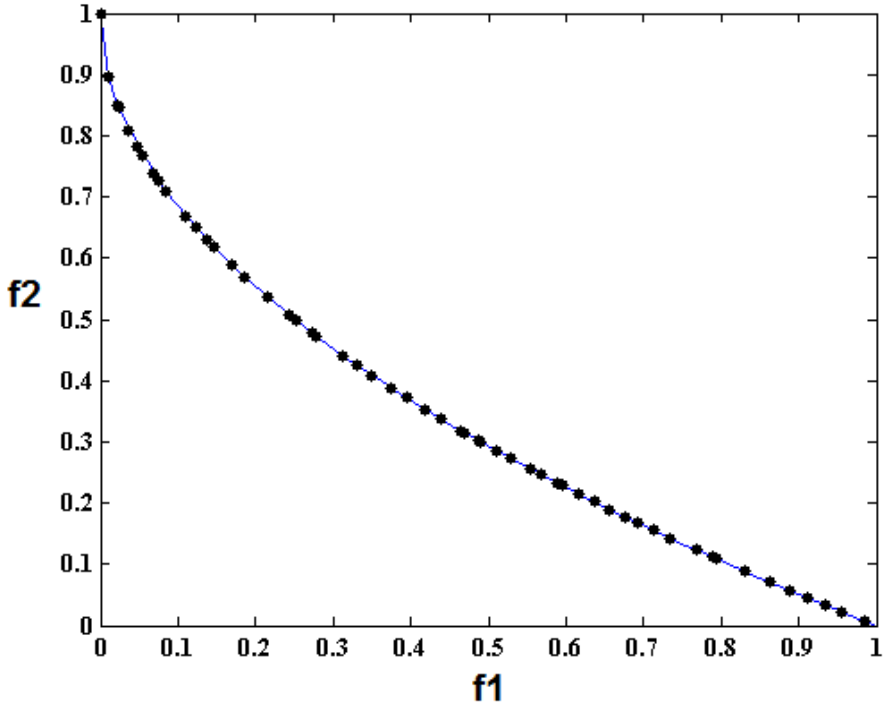


Figura 6: Função teste ZDT1.

4.2.2.2 FUNÇÃO DE TESTE ZDT2

Esta função tem uma fronteira Pareto-ótima não-convexa:

$$\begin{aligned} & f_1(x_1) = x_1 \\ & g(x_2,...,x_m) = 1 + 9.\sum_{i=2}^{m} x_i/(m-1) \\ & h(f_1,g) = 1 - (f_1/g)^2 \end{aligned} \tag{22}$$

onde $m = 30$ e $x_i \in [0,1]$. A fronteira Pareto-ótima é formada com $g(x) = 1$.

A Figura 7 mostra os resultados do cálculo da função ZDT2 fornecidos pelo MSFLPA, onde a fronteira Pareto-ótima da função de teste é apresentada como uma linha contínua azul e as soluções não-dominadas encontradas pelo MSFLPA são os pontos pretos nos gráfico.

4.2.2.3 FUNÇÃO DE TESTE ZDT3

Esta função tem uma fronteira Pareto-ótima não-convexa e descontínua:

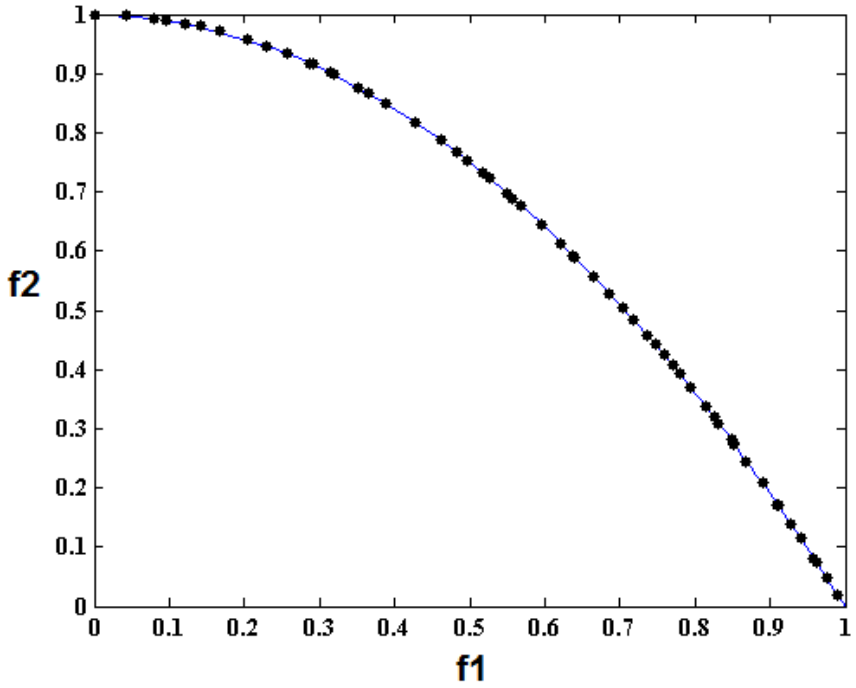


Figura 7: Função teste ZDT2.

$$
\begin{aligned}
& f_1(x_1) = x_1 \\
& g(x_2,...,x_m) = 1 + 9.\sum_{i=2}^{m} x_i/(m-1) \\
& h(f_1,g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)
\end{aligned} \tag{24}
$$

onde $m = 30$ e $x_i \in [0,1]$. A fronteira Pareto-ótima é formada com $g(x) = 1$.

A Figura 8 mostra os resultados do cálculo da função ZDT3 fornecidos pelo MSFLPA, onde a fronteira Pareto-ótima da função de teste é apresentada como uma linha contínua azul e as soluções não-dominadas encontradas pelo MSFLPA são os pontos pretos nos gráfico.

4.2.2.4 FUNÇÃO DE TESTE ZDT4

Esta função tem uma fronteira Pareto-ótima convexa, porém contém 21^9 fronteiras Pareto-ótima locais e testa a habilidade dos AEMOs de lidar com a multimodalidade:

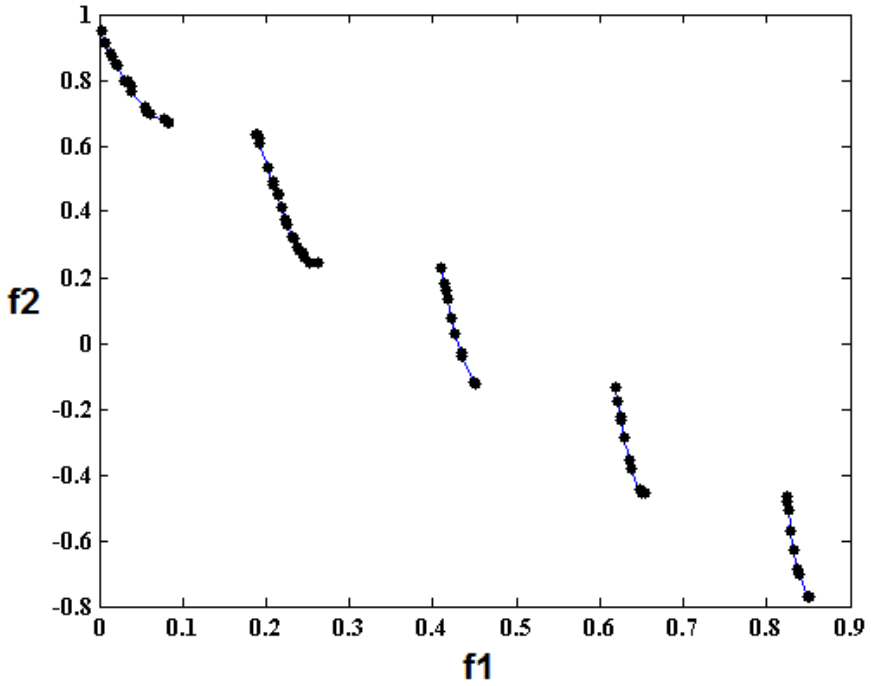


Figura 8: Função teste ZDT.3

$$
\begin{aligned}
& f_1(x_1) = x_1 \\
& g(x_2,...,x_m) = 1 + 10(m-1) + \sum_{i=2}^{m} (x_1^2 - 10\cos(4\pi x_i)) \\
& h(f_1,g) = 1 - \sqrt{f_1/g}
\end{aligned} \tag{26}
$$

onde $m = 10$, $x_i \in [0,1]$ e $x_2,...x_m \in [-5,5]$. A fronteira Pareto-ótima global é formada com $g(x) = 1$, a melhor fronteira Pareto-ótima local é formada com $g(x) = 1.25$.

A Figura 9 mostra os resultados do cálculo da função ZDT4 fornecidos pelo MSFLPA, onde a fronteira Pareto-ótima da função de teste é apresentada como uma linha contínua azul e as soluções não-dominadas encontradas pelo MSFLPA são os pontos pretos nos gráfico.

4.2.2.5 FUNÇÃO DE TESTE ZDT6

Esta função tem uma fronteira Pareto-ótima não-convexa e inclui duas dificuldades causadas pela falta de uniformidade do espaço de busca: as soluções Pareto-ótimas não são uniformemente distribuídas ao longo da fronteira de Pareto global (a fronteira é tendenciosa para soluções nas quais $f_1(x)$ está perto de um); e ainda, a densidade das soluções é menor próximo da fronteira Pareto-ótima e maior longe da fronteira:

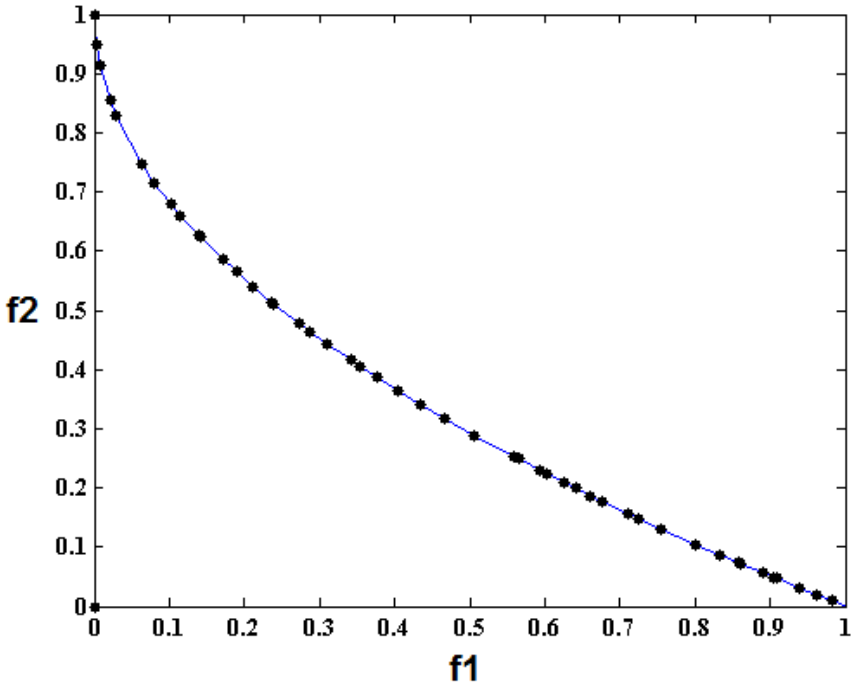


Figura 9: Função teste ZDT4.

$$
\begin{aligned}
f_1(x_1) &= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\
g(x_2,...,x_m) &= 1 + 9.((\sum_{i=2}^{m} x_i)/(m-1))^{0.25} \\
h(f_1,g) &= 1 - (f_1/g)^2
\end{aligned} \tag{28}
$$

onde $m = 10$, $x_i \in [0,1]$ e $x_2,...x_m \in [-5,5]$. A fronteira Pareto-ótima global é formada com $g(x) = 1$, a melhor fronteira Pareto-ótima local é formada com $g(x) = 1.25$.

A Figura 10 mostra os resultados do cálculo da função ZDT6 fornecidos pelo MSFLPA, onde a fronteira Pareto-ótima da função de teste é apresentada como uma linha contínua azul e as soluções não-dominadas encontradas pelo MSFLPA são os pontos pretos nos gráfico.

As Figuras 6 a 10 demonstram claramente que as soluções não-dominadas produzidas pela nova abordagem MSFLPA aproximadamente convergem para a fronteira Pareto-ótima e essas soluções estão espalhadas ao longo da fronteira Pareto-ótima para todas as funções ZDT testada.

Como mencionado anteriormente, existem duas metas principais em uma otimização multiobjetivo: 1) a convergência para o conjunto Pareto-ótimo e 2) a manutenção da

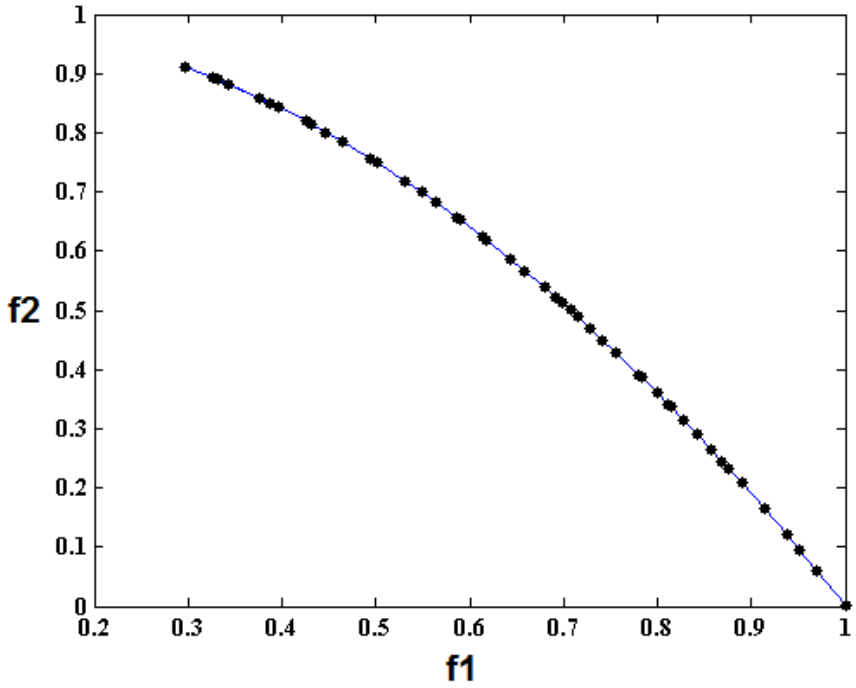


Figura 10: Função teste ZDT6.

diversidade (espalhamento) das soluções do conjunto Pareto-ótimo. Estas duas tarefas precisam ser medidas de forma adequada utilizando métricas de desempenho. Na próxima seção apresenta-se duas métricas de desempenho que foram definidas em (DEB et al., 2002a) para avaliar diretamente cada uma das duas metas, acima referidas, em um conjunto de soluções obtidas através de um algoritmo de otimização multiobjetivo.

4.2.3 MEDIDAS DE DESEMPENHO

A primeira métrica chamada de distância de geração (*generational distance* - GD) - γ mede o grau de convergência para um conjunto conhecido de soluções Pareto-ótimas e a segunda métrica, chamada de espalhamento (*Spread*) - Δ mede a distância de "espalhamento" entre as soluções obtidas na fronteira de Pareto (DEB et al., 2002a). A seguir a descrição destas métricas utilizadas para avaliar o desempenho do algoritmo proposto.

4.2.3.1 *GENERATIONAL DISTANCE* - GD - γ

A métrica γ mede o nível de convergência de um conjunto conhecido de soluções Pareto-ótimas. Primeiramente, para calcular a métrica o algoritmo deve encontrar um conjunto de soluções uniformemente espalhados na verdadeira fronteira Pareto-ótima no espaço objetivo. Para cada solução obtida pelo algoritmo, calcula-se a distância mínima euclidiana entre as

soluções escolhidas e a fronteira Pareto-ótima.

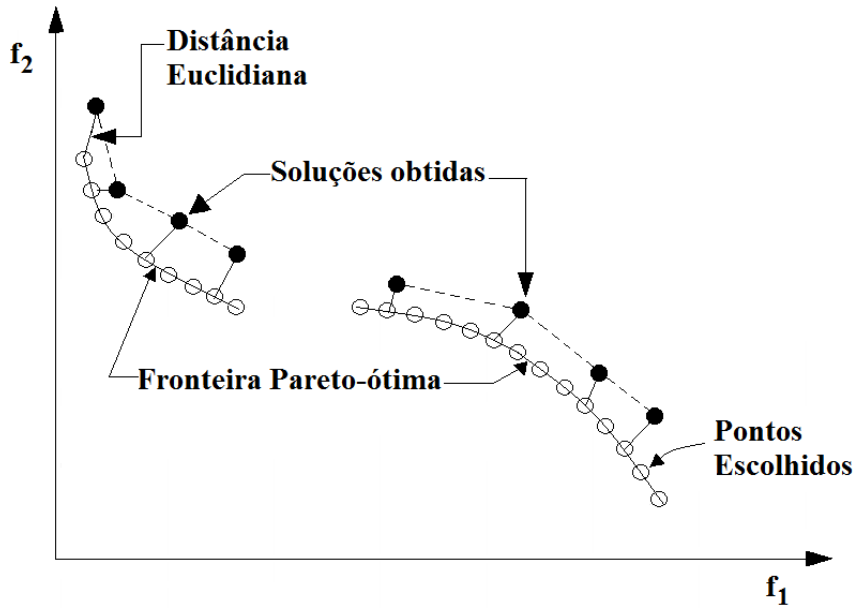


Figura 11: Métrica de distância - γ

Fonte: Adaptado de Deb et al. (2002a)

A Figura 11 mostra o processo de cálculo desta métrica. A linha contínua especifica as soluções Pareto-ótimas. As soluções com círculos abertos são soluções na fronteira Pareto-ótima escolhidas para o cálculo da métrica de convergência e as soluções marcados com círculos pretos são as soluções obtidas por um algoritmo. Quanto menor for o valor dessa métrica, melhor a convergência para a fronteira. Quando todas as soluções obtidas encontram-se exatamente na mesma fronteira das soluções escolhidas, esta métrica tem valor zero. A métrica γ pode ser obtida através da expressão:

$$\gamma = \sum_{i=1}^{n} D_i / n \tag{30}$$

onde n é o número de soluções não-dominadas e D_i é o mínimo da distância Euclidiana entre a $i-$ésima solução não-dominada e a verdadeira fronteira de Pareto no espaço objetivo.

4.2.3.2 *SPREAD* - SP - Δ

A métrica Δ mede a extensão do espalhamento (diversidade) alcançado entre as soluções obtidas. Calcula-se a distância euclidiana entre as soluções consecutivas no conjunto de soluções não-dominadas obtidas. Em seguida, calcula-se a média destas distâncias. Depois, a partir do conjunto de soluções não-dominadas obtido, calcula-se as soluções extremas (no

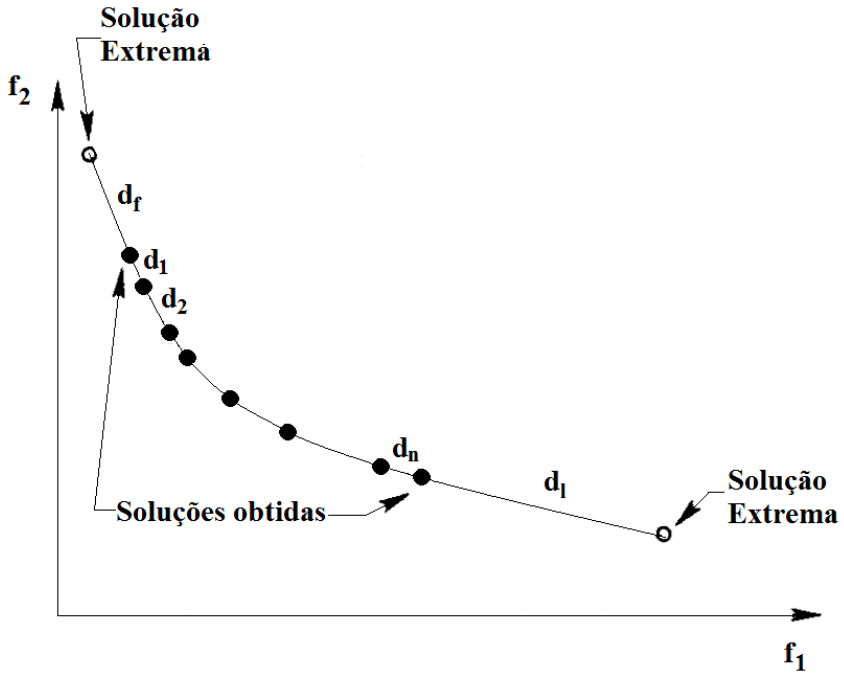


Figura 12: Métrica de diversidade - Δ

Fonte: Adaptado de Deb et al. (2002a)

espaço objetivo) ajustando uma curva paralela à da verdadeira fronteira Pareto-ótima. Então, usa-se a seguinte equação para calcular a não-uniformidade na distribuição das soluções:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} |d_i - \overline{d}|}{d_f + d_l + (n-1)\overline{d}} \tag{31}$$

onde o parâmetro d_i é a distância Euclidiana entre ith e $i+1$ solução no espaço objetivo e $\overline{d}$ é a média da todas, e os parâmetros d_f e d_l são as distâncias Euclidianas entre as soluções extremas e as soluções que delimitam o conjunto não-dominado obtido, como representado na Figura 12. A figura ilustra todas as distâncias referidas na equação acima.

Desta forma, para a validação de desempenho do algoritmo MSFLPA proposto, foi realizado o cálculo das medidas de convergência e de diversidade utilizando as equações 30 e 31 e os resultados obtidos foram comparados com dois algoritmos bem conhecidos na literatura de otimização multiobjetivo evolucionária: NSGA-II (DEB et al., 2002a) e SPEA2 (ZITZLER et al., 2001). Além desses, também foram incluídos na comparação, o algoritmo SFLA original para mostrar que as modificações feitas no SFLA são eficazes para problemas multiobjetivos e o μAG proposto por (RIBAS et al., 2013; ARRUDA et al., 2008).

Os resultados dessas métricas para as funções ZDTs dos algoritmos NSGA-II e SPEA2 foram obtidos de (DEB et al., 2002a) e (DE-MING; ZHI-MING, 2005), respectivamente.

Tabela 3: Média e Variância da métrica de Convergência γ

Algoritmo	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II (M)	0.03348	0.07239	0.11450	0.51305	0.29656
(VAR)	0.00475	0.03168	0.00794	0.11846	0.01314
SPEA2 (M)	0.02329	0.16762	0.01841	4.92710	0.23255
(VAR)	0.00000	0.00082	0.00000	2.70300	0.00495
MSFLPA (M)	**0.00000**	**0.00005**	**0.00039**	**0.00060**	**0.00000**
(VAR)	0.00000	0.00000	0.00001	0.00005	0.00000
SFLA (M)	0.01870	0.02180	0.02460	0.04530	0.00400
(VAR)	0.00017	0.00042	0.00059	0.00025	0.00002
μAG (M)	0.04400	0.01080	0.02880	0.01310	0.11430
(VAR)	0.00110	0.00052	0.00019	0.00110	0.01400

O número de gerações para todos os algoritmos é definido como 250, o mesmo número de iterações do NSGA-II e SPEA2. Os outros parâmetros do MSFLPA e SFLA são: população $|F| = 40$, número de memeplexos $m = 4$, iterações evolutivas $ei = 10$ e fator de aceleração $C = 2$ e os parâmetros do μAG são: população $|F| = 40$, taxa de mutação $tm = 0.1$ e taxa de *crossover* $tc = 0.8$.

As Tabelas 3 e 4 apresentam os resultados da média (M) e Variância (VAR) da comparação, foram realizadas 20 simulações de cada função teste ZDT para cada algoritmo.

Na Tabela 3, pode-se notar que o MSFLPA é capaz de convergir melhor (menor valor γ) em todas as funções de teste, especialmente para ZDT1, ZDT2 e ZDT6. O fator de aceleração combinado com a estratégia de arquivamento do MSFLPA permite alcançar uma melhor convergência para todas as funções de teste. A Tabela 4 mostra que a estratégia de arquivamento com base em duas memórias auxiliares para manter as soluções não-dominadas do MSFLPA obtém melhores medidas de diversidade (menor valor Δ) para todas as funções de teste, exceto para ZDT1 (SPEA2 apresenta melhor desempenho). Estes resultados confirmam a conclusão do estudo comparativo realizado no (TIWARI et al., 2008): um algoritmo genético com população pequena e estratégia de arquivamento eficiente pode funcionar melhor que o NSGA e SPEA.

Os resultados das funções de teste e as medidas de convergência e diversidade mostram que a algoritmo MSFLPA proposto obteve bons resultados quando comparado aos outros algoritmos multiobjetivos e, a partir desses resultados com os testes de validação e desempenho verifica-se ser eficiente e competitivo para a resolução de problemas multiobjetivos.

Para reforçar e fundamentar ainda mais a observação de que os resultados obtidos pelo MSFLPA são satisfatórios para todas as funções testes apresentadas, foram comparados os valores das métricas de convergência e diversidade dos cinco algoritmos usando o teste

Tabela 4: Média e Variância da métrica de Diversidade Δ

Algoritmo	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II (M)	0.3903	0.4307	0.7385	0.7026	0.6680
(VAR)	0.0018	0.0047	0.0197	0.0646	0.0099
SPEA2 (M)	**0.1547**	0.3394	0.4691	0.8239	1.04422
(VAR)	0.0009	0.0017	0.0053	0.0029	0.1581
MSFLPA (M)	0.3128	**0.3007**	**0.2867**	**0.3694**	**0.4818**
(VAR)	0.0003	0.0003	0.0009	0.0016	0.0016
SFLA (M)	0.7930	0.7719	0.7750	0.8122	1.3017
(VAR)	0.0047	0.0087	0.0144	0.0073	0.0077
μAG (M)	0.5270	0.6151	0.4447	0.9754	1.3636
(VAR)	0.0082	0.0213	0.0083	0.0436	0.0169

Tabela 5: Resultados de teste estatístico t de Student - Métrica de Convergência γ

Algoritmo-1 $\leftrightarrow$ Algoritmo-2	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MSFLPA $\leftrightarrow$ NSGA-II	+	$\sim$	+	+	+
MSFLPA $\leftrightarrow$ SPEA2	+	+	+	+	+
MSFLPA $\leftrightarrow$ SFLA	+	+	+	+	+
MSFLPA $\leftrightarrow$ μAG	+	+	+	$\sim$	+

estatístico t de *Student* (WALPOLE et al., 1998).

A Tabela 5 e a Tabela 6 apresentam os resultados estatísticos obtidos por um teste t de *Student* bicaudal, com 38 graus de liberdade a um nível de significância de 0.05. O resultado do Algoritmo-1 $\leftrightarrow$ Algoritmo-2 é mostrado como $+$ ou $\sim$ quando o Algoritmo-1 é significativamente melhor do que ou estatisticamente equivalente ao Algoritmo-2, respectivamente.

A partir da Tabela 5 e Tabela 6, pode-se concluir que o algoritmo MSFLPA tem um melhor ou um desempenho estatístico semelhante em termos de convergência e diversidade na fronteira de Pareto que o NSGA-II, SPEA2 , SFLA e μAG.

De acordo com os resultados do teste t de Student, ambos algoritmos NSGA-II e μAG apresentaram na Tabela 5 um valor estatisticamente equivalente ao algoritmo MSFLPA, assim, eles são considerados para comparação de resultados. No Capítulo 7, os algoritmos MSFLPA, NSGA-II e μAG são usadas para resolver o problema de escalonamento de uma rede de dutos.

Tabela 6: Resultados de teste estatístico t de Student - Métrica de Diversidade Δ

Algoritmo1 $\leftrightarrow$ Algoritmo2	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MSFLPA $\leftrightarrow$ NSGA-II	+	+	+	+	+
MSFLPA $\leftrightarrow$ SPEA2	$\sim$	+	+	+	+
MSFLPA $\leftrightarrow$ SFLA	+	+	+	+	+
MSFLPA $\leftrightarrow$ μAG	+	+	+	+	+

4.3 CONSIDERAÇÕES

Os experimentos numéricos apresentados neste capítulo demonstram que o novo algoritmo proposto, o MSFLPA, é capaz de construir a fronteira de Pareto assim como os algoritmos NGSA-II e SPEA2. Além disso MSFLPA superou ambos os algoritmos no que diz respeito as métricas de convergência e diversidade. Isso demonstra a eficácia do novo algoritmo para resolver problemas multiobjetivos como os apresentado nas funções testes na Seção 4.2.2.

Dessa forma, pode-se concluir que a partir do resultados obtidos, o MSFLPA fornece um conjunto razoável de soluções não-dominadas, que é aproximadamente perto da fronteira de Pareto e uniformemente distribuído ao longo do fronteira obtida, e também pode obter rapidamente uma série de soluções em uma única execução.

No próximo capítulo é descrito o problema estudado neste trabalho, também é apresentado o problema de redes de dutos, a modelagem multiobjetivo utilizada para obter soluções viáveis para este tipo de problema e a metodologia empregada nos modelos multiobjetivos evolucionários.

5 DESCRIÇÃO DO PROBLEMA

Neste capítulo é apresentada a descrição do problema de escalonamento de rede de dutos aqui estudada.

5.1 REDE DE DUTOS

O transporte de produtos em uma rede de distribuição de petróleo mostra-se confiável e econômico, porém é uma atividade complexa. Diante disso, há uma grande necessidade de desenvolver técnicas para uma utilização mais eficiente das redes (BOSCHETTO et al., 2010).

Uma rede de distribuição de petróleo é composta por um dado número de refinarias e terminais, ou áreas, interligadas por um conjunto de oleodutos, ou trechos de dutos, os quais operam o transporte de um conjunto de produtos (petróleos, derivados de petróleos e produtos orgânicos) entre áreas adjacentes.

Geralmente um produto é transportado de refinarias, portos e/ou centro de armazenagens para pontos de destino. Os dutos podem ser unidirecionais ou bidirecionais, em uma conexão unidirecional, os produtos fluem somente em um sentido, de um terminal para a refinaria ou somente da refinaria para o terminal. Já na conexão bidirecional, os produtos podem fluir tanto do terminal para a refinaria quanto da refinaria para o terminal, mas nunca ao mesmo tempo.

A malha dutoviária da indústria brasileira de petróleo é composta basicamente por duas redes principais, a rede de escuros e a rede de claros, diferenciadas pelo tráfego de diferentes tipos de produtos. A rede de escuros realiza o transporte de diversos tipos de petróleos e derivados pesados e a rede de claros transporta diversos tipos de derivados leves de alto valor agregado. A programação das operações envolvidas nestas duas redes é realizada de forma separada, visto que não há uma interdependência operacional significativa entre essas malhas. A Figura 13 apresenta a rede de dutos brasileira, conforme disponibilizado em Transpetro (2013).

Contudo, mesmo considerando as redes de claros e escuros de forma independente,

ainda assim a programação das operações continua sendo uma tarefa difícil. Programações não efetuadas da melhor forma possível conduzem invariavelmente a perdas operacionais, tais como, não atendimento à demanda, manutenção de estoques incoerentes com metas pré-estabelecidas, movimentações e deslocamentos desnecessários de produtos (FELIZARI, 2009).

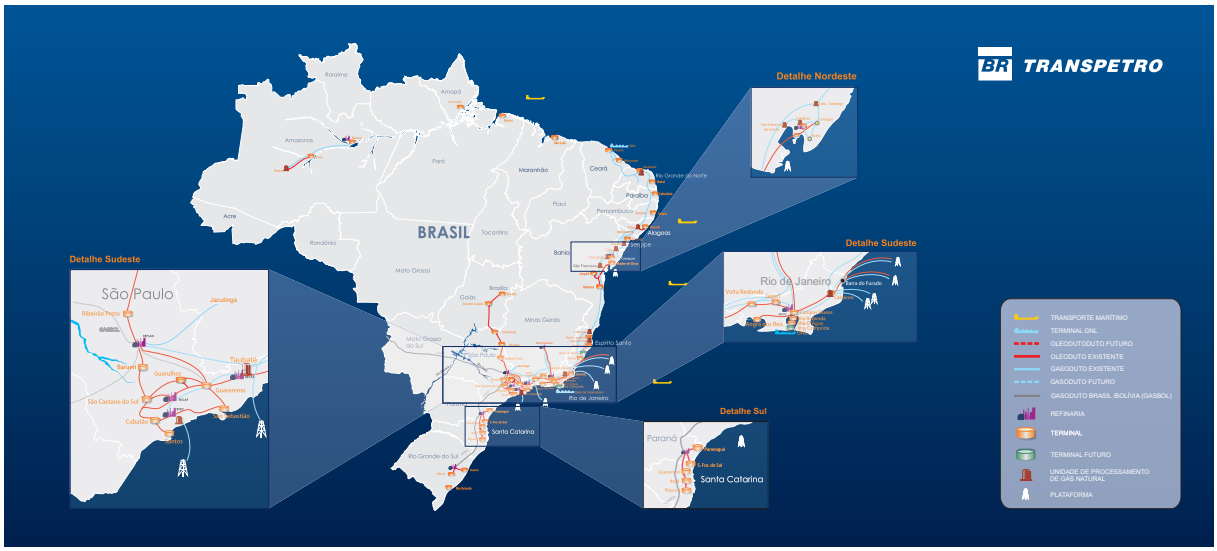


Figura 13: Mapa de Terminais e Dutos

Fonte: Transpetro (2013)

Devido a grande dificuldade de desenvolver uma solução que contemple todas as variáveis envolvidas na programação das atividades de operação de uma rede de dutos, se torna evidente a necessidade de uma ferramenta que auxilie o procedimento de tomada de decisão operacional.

Dentro deste cenário em estudo, a busca por modelos de escalonamento que possam ser implementados na prática, considerando uma carga computacional aceitável torna-se um grande desafio. Desta forma, uma abordagem de decomposição do problema deve ser utilizada para tornar viável a implementação de um sistema de apoio à tomada de decisão para operadores de redes de dutos. A subdivisão adotada é baseada nos três elementos chave do escalonamento: a determinação e alocação dos recursos a serem utilizados (*assignment*), o seqüenciamento de atividades (*sequencing*) e a temporização (*timing*) do uso dos recursos pelas atividades (REKLAITIS, 1992).

Neste trabalho foi desenvolvido um modelo de otimização, para dois elementos do escalonamento, a alocação e sequenciamento operacional para a rede de escuros. A Rede de Escuros é parte integrante do sistema dutoviário do estado de São Paulo, que é operado pela Transpetro. Este sistema, conforme pode ser visto na Figura 14, possui uma malha de

oleodutos que interliga quatro refinarias (REPLAN, REVAP, RECAP e RPBC) a seis terminais de distribuição (Guararema, Guarulhos, Barueri, São Caetano, Cubatão e Santos).

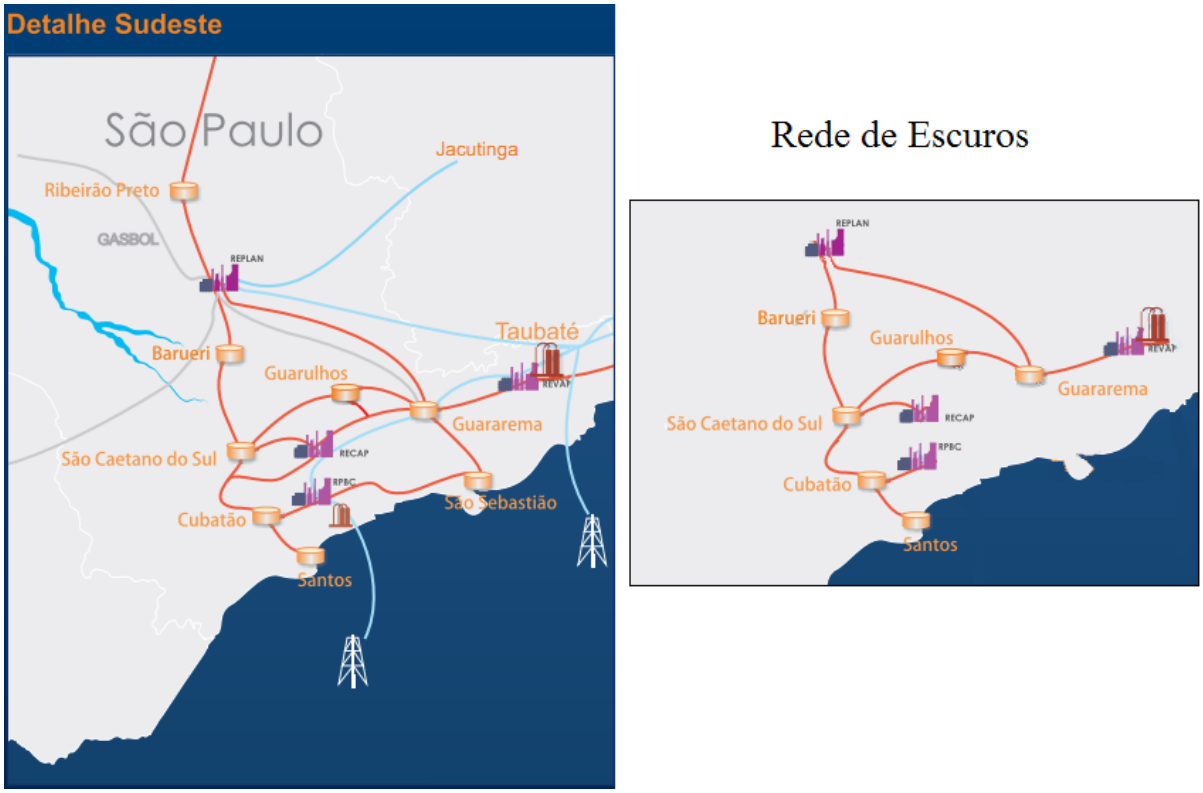


Figura 14: Rede de Dutos - Detalhe Sudeste

Fonte: Transpetro (2013)

A Rede de Escuros é responsável pelo transporte dos derivados pesados do petróleo, como o óleo combustível, óleo combustível marítimo (bunker), gasóleo para craqueamento e resíduo atmosférico, que são produtos com coloração escura e alta viscosidade.

5.2 MODELAGEM DA REDE

A modelagem da rede de escuros, utilizada neste trabalho, pode ser vista na Figura 15. A adaptação desenvolvida a partir da rede real é composta por: oito nós, dos quais quatro deles são refinarias (nós N1, N2, N3 e N4) representando REPLAN, REVAP, RECAP e RPBC, respectivamente. Um porto (nó N8) representado Santos e três centros de distribuição (nós N5, N6 e N7) representando Barueri, São Caetano e Cubatão, respectivamente.

As linhas que unem os nós representam os dutos e as flechas representam a direção que flui o produto transportado. Os nós, N1 a N4, fornecem produtos para N5 a N7 através das conexões D1, D2, D3 e D4. Essas conexões são unidirecionais, ou seja, o produto flui somente

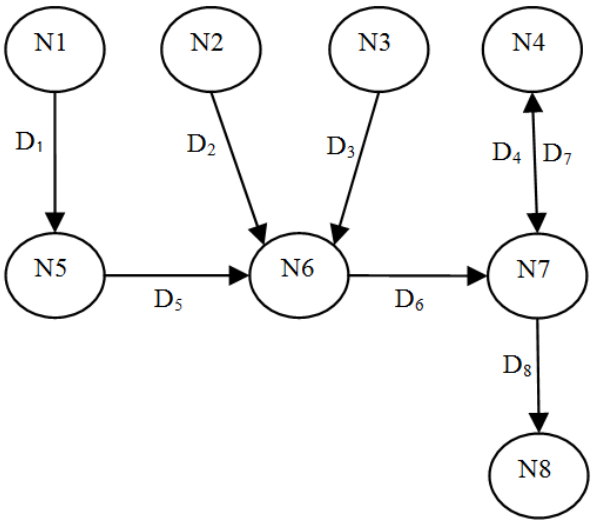


Figura 15: Modelo da Rede de Distribuição de Escuros

na direção da fonte para os nós intermediários. Os nós N4 e N7 são conectados através de um poliduto bidirecional D4 e D7. A conexão D5 conecta o nó N5 à N6, a conexão D6 conecta o nó N6 ao nó N7. A conexão D8 conecta o nó N7 ao nó N8. Os produtos produzidos nas fontes N1, N2, N3 e N4 estão sendo designados de forma genérica e nomeados como produto 1, 2, 3 e 4. O produto que cada fonte produz é configurado conforme desejado. Por exemplo, as fonte N1 e N2 podem ser configuradas para produzir produtos do tipo 1 e 2 e as fontes N3 e N4 para produzir produtos do tipo 3 e 4.

O modelo considera que os produtos são entregues na forma de bateladas discretas. Uma batelada representa um volume mínimo de um produto a ser transportado em uma unidade de tempo. Uma batelada unitária é um volume mínimo que preenche um duto. Cada conexão possui uma distância normalizada em termos de unidades de tempo necessárias para que uma dada batelada seja transportada de um ponto a outro. O modelo assume a discretização do tempo.

Supõe-se que cada terminal ou nó intermediário possui a quantidade de tanques necessária para cada produto que ele possa receber. Todas as conexões têm as mesmas características como diâmetro, volume, velocidade do fluxo etc. As conexões D1, D2,..., D8 possuem distâncias normalizadas em termos de unidades discretas de tempo necessárias para que uma dada batelada seja transportada de um ponto a outro. Portanto o tempo de transporte de uma batelada de um nó a outro na rede é um número inteiro e, os tempos envolvidos em todas as conexões são mostrados na Tabela 7.

A solução para o problema é dada por cada tipo de batelada que é enviada em cada instante de tempo em cada conexão da rede. O modelo de escalonamento da rede deve atender

Tabela 7: Matriz de tempo das conexões na rede

Conexão	Nó origem	Nó destino	Tempo
1	1	5	1
2	2	6	1
3	3	6	3
4	4	7	4
5	5	6	2
6	6	7	2
7	7	4	4
8	7	8	3

aos seguintes objetivos:

- Demanda de fornecimento: definida como a quantidade mínima de bateladas de cada produto que deve ser enviada de uma refinaria, evitando assim a saturação dos seus tanques de armazenagem. Não é desejável paralisar a produção por falta de parque de armazenagem.
- Demanda de recebimento: definida como a quantidade de bateladas de cada produto que deve ser entregue aos pontos solicitantes (terminais).
- Minimização do tempo para atendimento às demandas, este tempo é denominado horizonte de programação.
- Minimização da fragmentação das bateladas nas conexões: evitar a alternância no envio das bateladas dos produtos, ou seja, tentar enviar sempre sequências de bateladas do mesmo produto. A alternância no tipo de produto a ser enviado pode provocar contaminações nas fronteiras do produto enviado e por isso a fragmentação de bateladas não é desejada.

5.2.1 REPRESENTAÇÃO DA REDE PARA ALGORITMOS EVOLUCIONÁRIOS

Para representar a população no modelo é necessário codificar a solução em uma matriz, onde cada linha representa um indivíduo. Cada indivíduo é composto por instantes de tempo, que por sua vez contém o valor da batelada (tipo de produto transportado) de todas as conexões naquele instante. A representação assume a discretização do tempo.

Por exemplo, a Tabela 8 mostra um indivíduo que representa o estado da rede em 10 instantes de tempo: os primeiros n elementos representam as bateladas enviadas através das n conexões no primeiro instante de tempo, os n elementos seguintes representam as

bateladas para o segundo instante de tempo e assim sucessivamente. Os valores estão no intervalo $[0; numerodeprodutos]$ onde 0 representa uma conexão vazia. (WESTPHAL, 2006; WESTPHAL; ARRUDA, 2007; WESTPHAL et al., 2011).

Tabela 8: Codificação da solução

Tempo	Instante 1	...	Instante 10
Conexão	1 2 3 4 5 6 7 8	...	1 2 3 4 5 6 7 8
Solução	1 0 1 2 2 3 3 0	...	2 4 3 0 2 2 2 3

As fontes N1 a N4 podem produzir tipos de produtos diferentes, assim para o modelo se aproximar da situação real, algumas conexões não podem assumir determinados valores. A Tabela 9 mostra um exemplo de compatibilidade de produtos (P1 a P8) por conexão, respeitando as produções nas refinarias, onde ○ indica que o produto pode passar na conexão e *x* indica que o produto não pode passar na conexão.

Tabela 9: Matriz de compatibilidade de produtos por conexão

Conexão	P1	P2	P3	P4	P5	P6	P7	P8
1	○	*x*	*x*	○	○	*x*	*x*	○
2	*x*	○	○	*x*	*x*	○	○	*x*
3	○	*x*	*x*	○	○	*x*	*x*	○
4	*x*	○	○	*x*	*x*	○	○	*x*
5	○	*x*	*x*	○	○	*x*	*x*	○
6	○	○	○	○	○	○	○	○
7	*x*	○	○	*x*	*x*	○	○	*x*
8	○	○	○	○	○	○	○	○

5.3 OTIMIZAÇÃO MULTIOBJETIVO

Para detalhar e facilitar o entendimento da otimização multiobjetivo empregada neste trabalho, foi utilizada a seguinte notação:

- D_{ij} : quantidade de bateladas demandada do produto j pelo destino i;
- R_{ij} : quantidade de bateladas recebida do produto j no destino i;
- P_{ij} : quantidade mínima de bateladas do produto j que deve ser enviado pela fonte i;
- C_{ij} : quantidade de bateladas do produto j armazenado no tanque i;
- E_{ij} : quantidade de bateladas do produto j enviado pela fonte i;
- $Tchegada_{ij}$: tempo de chegada da última batelada do produto i no destino j;

- $Tmax$: horizonte de tempo do modelo;
- $Tmin$: menor tempo para receber uma batelada no destino i;
- NC : número de colisões na conexão bidirecional;
- LCm_{ij} : limite inferior da quantidade de bateladas do produto j no nó i;
- LCM_{ij} : limite superior da quantidade de bateladas do produto j no nó i;
- N_{conex}: número de conexões;
- $Frag_i$: quantidade de fragmentações na conexão i;
- Nt_i : quantidade de tanques no nó i;
- N_f : quantidade de fontes;
- N_d : quantidade de destinos;
- $Mask[i]$: matriz contendo a quantidade de produto em cada conexão em um dado instante de tempo;

5.3.1 RESTRIÇÕES

O modelo multiobjetivo implementado está sujeito às seguintes restrições:

1. A demanda em cada destino deve ser cumprida e cada destino não deve receber mais do que o planejado:

$$R_{ij} = D_{ij} \tag{32}$$

$$\text{para } i = 1,...,N_d$$
$$\text{e } j = 1,...,Nt_i$$

Ressaltando que cada nó terminal ou intermediário possui a quantidade de tanques necessária para cada produto que ele possa receber.

2. Uma quantidade mínima de bateladas de cada produto deve ser enviada. Para evitar a paralisação da produção em uma refinaria e por não haver recursos (tanques disponíveis) para armazenagem dos produtos, uma quantidade mínima de produtos deve ser enviada.:

$$P_{ij} \leq E_{ij} \leq LCM_{ij} \tag{33}$$

$$\text{para } i = 1, ...N_f$$
$$\text{e } j = 1, ...Nt_i$$

3. O número de bateladas nos tanques não pode violar o limite mínimo e máximo para cada nó i e produto j.

$$LCm_{ij} \leq C_{ij} \leq LCM_{ij} \tag{34}$$

$$\text{para } i = 1...N_f$$
$$\text{e } j = 1...Nt_i$$

4. Um nó não pode enviar e receber ao mesmo tempo produtos de uma conexão bidirecional, portanto ou o nó está em um estado de envio ou de recebimento ou ocioso. Então não podem existir colisões na conexão bidirecional.

$$NC = 0 \tag{35}$$

As restrições 1 e 2 estão diretamente relacionadas com os critérios de demanda e fornecimento e serão consideradas como objetivos para o modelo de otimização. Para tratar as restrições 3 e 4 foi utilizada uma estratégia de reparação da solução que consiste em reparar um indivíduo infactível, transformando-o em uma solução factível (WESTPHAL, 2006). A reparação da restrição 4 considera que a conexão bidirecional é desdobrada em duas conexões, uma representando o envio em um sentido, e a outra representando o envio no sentido oposto. Dessa maneira, toda vez que duas bateladas existirem na conexão ao mesmo tempo, a batelada que iniciar primeiro será mantida, e a outra será retirada. Se as bateladas iniciaram ao mesmo tempo o envio, então uma delas é retirada aleatoriamente. Para reparar a restrição 3, usou-se uma função recursiva que faz a contagem das bateladas e atualiza o estado de cada tanque, como proposto por (WESTPHAL; ARRUDA, 2007), (WESTPHAL et al., 2011).

Conforme discutido na seção 5.2, o modelo de escalonamento da rede desenvolvido deve atender a quatro objetivos definidos pela política de escalonamento da rede de dutos. A formulação matemática desses objetivos é dada pelas equações 36 para 40.

1. Receber a quantidade de bateladas demandada pelos terminais ($f_1(x)$):

$$min \frac{1}{N_d} \sum_{j=0}^{N_d} \frac{1}{Nt_j} \sum_{i=0}^{Nt_j} 1 - \frac{R_{ij}}{D_{ij}} \tag{36}$$

A equação 36 apresenta a minimização do atendimento à demanda de cada produto em cada destino, considerando todos os destinos conjuntamente. O valor mínimo é 0 indicando que todas as demandas dos produtos foram cumpridas durante o horizonte de programação.

2. Satisfazer a produção mínima ($f_2(x)$):

$$min\frac{1}{N_f}\sum_{j=0}^{N_f}\frac{1}{Nt_j}\sum_{i=0}^{Nt_j}EP(i,j) \tag{37}$$

onde:

$$EP(i,j) = \begin{pmatrix} 1-\frac{E_{ij}}{P_{ij}}, & E_{ij} \leq P_{ij} \\ 0, & E_{ij} > P_{ij} \end{pmatrix} \tag{38}$$

As equações 37 e 38 consideram a minimização do envio de uma produção mínima por produto e por fonte, considerando todas as fontes.

3. Minimizar o tempo de entrega das bateladas ($f_3(x)$). O termo $(D_{ij} - R_{ij})$ é uma penalidade caso a demanda não seja cumprida.

$$min\frac{1}{N_d}\sum_{j=0}^{N_d}\frac{1}{Nt_j}\sum_{i=0}^{Nt_j}\frac{\frac{Tarrival_{ij}-Tmin_j}{Tmax-Tmin_j}+D_{ij}-R_{ij}}{D_{ij}-R_{ij}+1} \tag{39}$$

A equação 39 minimiza o tempo de atendimento à demanda por produto em cada destino, considerando todos os destinos conjuntamente.

4. Minimizar a fragmentação das bateladas ($f_4(x)$):

$$min(\frac{\sum_{i=0}^{N_{conex}}FM(i)}{N_{conex}}) \tag{40}$$

onde:

$$FM(i) = \begin{pmatrix} \frac{Frag_i-Mask[i]-1}{Mask[i]}, & Frag_i \leq Mask[i] \\ 0, & Frag_i > Mask[i] \end{pmatrix} \tag{41}$$

As equações 40 e 41 minimizam a fragmentação no envio de bateladas em todas as conexões do modelo.

Além dos objetivos serem conflitantes entre si para conseguir alcançar um bom desempenho nas operações das redes de dutos, trabalha-se com objetivos com diferentes

unidades de medidas. Enquanto o objetivo 1 e 2 são valores em volume (m^3), o objetivo 3 trabalha com unidades de tempo discreto e o objetivo 4 com valores inteiros.

A metodologia aplicada para calcular a atribuição do *fitness* e para classificar os indivíduos da população baseia-se no método da soma ponderada (WESTPHAL; ARRUDA, 2007; RIBAS et al., 2013). Nesta abordagem, o conjunto de objetivos são agregados em um único objetivo, pré-multiplicando cada objetivo com um peso fornecido pelo usuário (DEB; KALYANMOY, 2001). No entanto, os diferentes objetivos dados pelas Equações 36 - 40 assumem diferentes ordens de magnitude. Por isso, de modo a compor a função de *fitness*, os valores dos objetivos são normalizados no intervalo de [0,1]:

$$ft_i = \frac{f_i(x)}{f_{max_i}} \tag{42}$$

onde f_{max_i} é o valor máximo (pior caso) aceito pelo critério $f_i(x)$. Estes valores dependem do número de bateladas que serão movimentadas através da rede. Depois que os objetivos são normalizados, a função de *fitness* pode ser calculado somando-se as funções objetivo normalizadas ponderadas:

$$F(x) = \frac{\sum_{i=1}^{4} w_i \cdot ft_i(x)}{\sum_{i=1}^{4} \cdot w_i} \tag{43}$$

onde w_i é o peso para cada objetivo i, e f_i é a função de minimização para cada objetivo i. Os resultados, para cada objetivo, ficam dentro da faixa $[0;1]$ de tal forma que 0 é o valor ótimo, ou seja, todos os critérios foram cumpridos. Qualquer valor entre 0 e 1 mostra que o critério foi atendido parcialmente.

5.4 METODOLOGIA

A metodologia desenvolvida é mostrada na Figura 16 e foi implementada em linguagem C++ utilizando a ferramenta C++Builder 2007 versão 11.

Etapa 1: Uma população é inicializada com valores aleatórios;

Etapa 2: As funções reparadoras corrigem as soluções infactíveis desta população de acordo com a violação de cada restrição. As violações referem-se aos limites máximo e mínimo dos tanques, a sobre produção e a existência de colisões de bateladas na conexão bidirecional;

Etapa 3: A população é em seguida avaliada e ranqueada de acordo com a função objetivo

na equação 43. Para isto são calculados os quatro objetivos referentes à produção mínima, ao atendimento à demanda (restrições modeladas como objetivos), ao tempo deste atendimento e por fim à fragmentação (objetivos propriamente ditos);

Etapa 4: Define-se o AE (algoritmo evolucionário) que será utilizado;

Etapa 5: Resumidamente, após o ranqueamento da população:

- O algoritmo NSGA-II (discutido na seção 3.1.5) classifica toda a população N em níveis de dominância, depois utiliza um mecanismo de seleção por torneio, baseado na técnica de *crowding distance*, em seguida aplica-se os operadores genéticos cruzamento e mutação e gera uma população $2N$ (DEB et al., 2002a).
- Para o algoritmo μAG (discutido na seção 3.1.2) as castas elitistas são formadas e alguns indivíduos são selecionados para a aplicação dos operadores genéticos de recombinação e mutação (WESTPHAL et al., 2011).
- No algoritmo MSFLPA (discutido na seção 4.1), a população é então dividida em *memeplexes* para ser realizada a busca local.
  Para todos os algoritmos, a nova população é então formada e está sujeita novamente a função reparadora, pois soluções infactíveis podem surgir em qualquer ocasião após as operações genéticas e a evolução memética.

No próximo capítulo, serão apresentados os resultados obtidos com os modelos multiobjetivos implementados utilizando os algoritmos MSFLPA, NSGA-II e μAG.

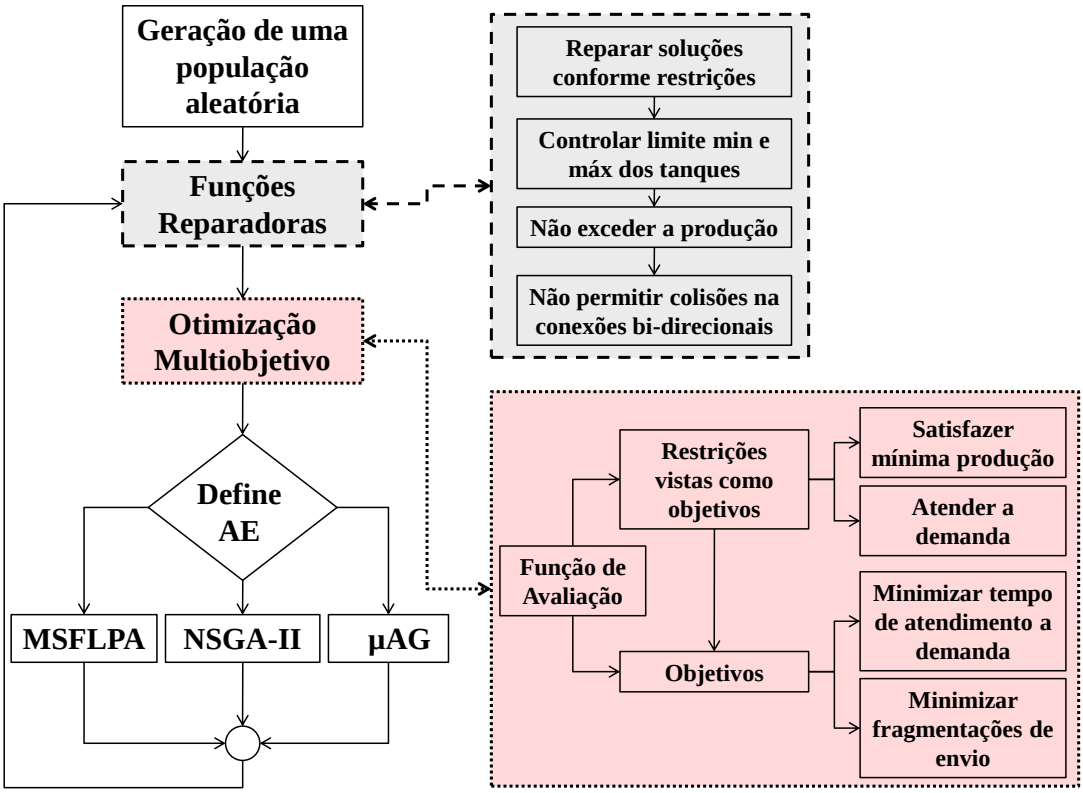


Figura 16: Metodologia para otimização multiobjetivo do transporte de produtos em rede de dutos

6 ESTUDO DE CASO

Primeiramente este capítulo mostra os resultados obtidos e apresentados em Lamboia et al. (2012) e Lamboia et al. (2014), executando simulações com o modelo multiobjetivo com configurações básicas e comparando os resultados entre o algoritmo μAG e o SFLA modificado. Em seguida, dois estudos de caso são apresentados, modificando alguns parâmetros no modelo e por fim, os resultados obtidos para dois cenários com 4 e 8 produtos com a nova abordagem: o algoritmo MSFLPA.

6.1 MODELO BÁSICO

Para executar o modelo, é necessário configurar o limite superior e inferior dos tanques, a distância em termos de unidades discretas de tempo de cada conexão (Tabela 7) e o horizonte de programação. Esta configuração é uma decisão de projeto.

O horizonte de programação das simulações a serem apresentadas é de 48 unidades de tempo, ou seja, o último recebimento de produto possível será na unidade de tempo 48 e nenhum envio poderá ocorrer após esse tempo.

Os objetivos de otimização para o modelo desenvolvido são:

1. Atender a demanda de recebimento no horizonte programado;
2. Atender a demanda de fornecimento de produtos;
3. Minimizar o tempo de atendimento da demanda de recebimento;
4. Minimizar a quantidade de fragmentação no envio das bateladas.

Os parâmetros de configuração do algoritmo μAG são:

✓ P (população) = 20;

✓ taxa de mutação = 0.1;

✓ taxa de *crossover* = 0.8.

Os parâmetros de configuração do algoritmo SFLA são:

✓ F (população) = 20;

✓ m (*memeplexes*) = 4;

✓ N (evoluções) = 5;

✓ C (fator de aceleração) = 1.7.

Os produtos produzidos pelas refinarias podem ser configurados conforme desejado, para estes primeiros resultados apresentados, são utilizados 4 produtos e as designações dos produtos produzidos por cada refinaria estão apresentados na Tabela 10.

Tabela 10: Produtos produzidos por cada refinaria

Refinaria/Nó	Produtos
N1	1,2
N2	1,3
N3	2,4
N4	3,4

As demandas de fornecimento são configuradas através de uma quantidade mínima de bateladas de cada produto que deve ser enviada a partir de cada refinaria. Assim como as demandas de recebimento são configuradas através da quantidade de bateladas de cada produto que um terminal deverá receber. Para a rede utilizada no modelo, o nó N8 é único terminal, e o objetivo geral da rede é atender a demanda desse nó. A Tabela 11 apresenta as configurações das demandas.

Tabela 11: Demanda de Fornecimento e Recebimento

Demanda	**Nó**	P1	P2	P3	P4
Fornecimento	N1	2	2	-	-
	N2	2	-	2	-
	N3	-	2	-	2
	N4	-	-	2	2
Recebimento	N8	4	4	4	4

A otimização dos objetivos levará em consideração apenas o resultado global de cada objetivo (WESTPHAL, 2006). A Tabela 12 apresenta os pesos associados a cada objetivo.

O objetivo de minimização da fragmentação no envio das bateladas possui um peso associado menor, pois ele pode prejudicar o desempenho de otimização dos outros objetivos.

Tabela 12: Ponderação para os objetivos globais

Critério	Peso
Demanda de recebimento	3
Demanda de fornecimento	3
Minimização do tempo	3
Minimização da Fragmentação	1

Contudo, evitar a fragmentação das bateladas é um critério muito importante, mas não é prioridade nas designações das atividades operacionais da rede de dutos.

Os resultados, para cada objetivo, ficam dentro da faixa $[0;1]$ de tal forma que 0 é o valor ótimo, ou seja, todos os critérios foram cumpridos. Qualquer valor entre 0 e 1 mostra que o critério foi atendido parcialmente.

Tabela 13: Resultados para um conjunto de 20 simulações para o algoritmo μAG

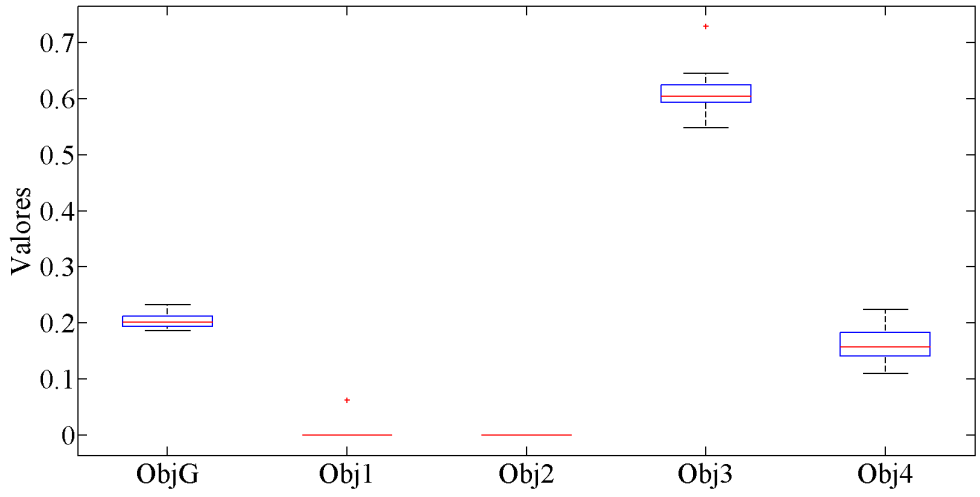
Objetivo	Mínimo	Máximo	Média	Mediana	Desvio Padrão
1	0	0	0	0	0
2	0	0.0625	0.0125	0	0.0256
3	0.5486	0.7292	0.6130	0.6042	0.0366
4	0.1094	0.2240	0.1617	0.1576	0.0329
Global	0.1859	0.2321	0.2037	0.2013	0.0123

Tabela 14: Resultados para um conjunto de 20 simulações para o algoritmo SFLA

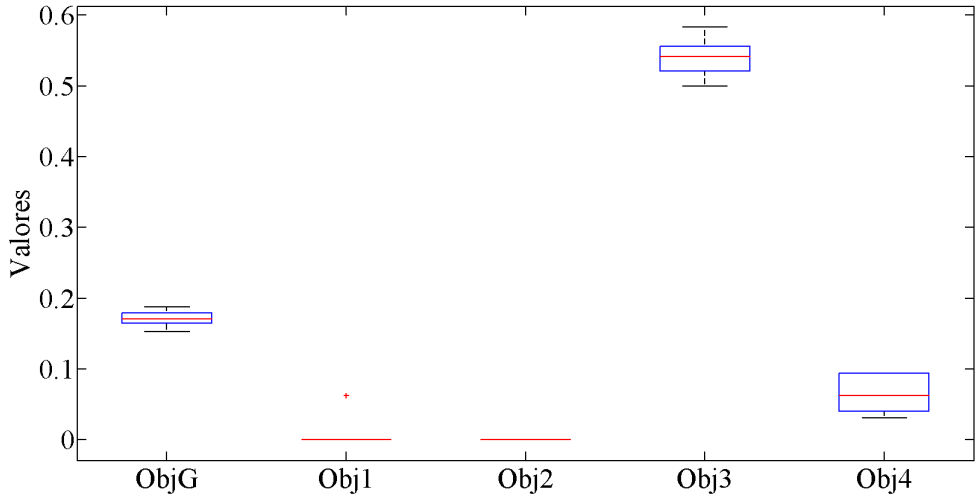
Objetivo	Mínimo	Máximo	Média	Mediana	Desvio Padrão
1	0	0	0	0	0
2	0	0.0625	0.0094	0	0.0229
3	0.5000	0.5833	0.5405	0.5417	0.0231
4	0.0313	0.0938	0.0641	0.0625	0.0250
Global	0.1531	0.1875	0.1714	0.1703	0.0095

Os dois algoritmos são executados utilizando o mesmo número de gerações: 1000. Todas as simulações foram realizadas em um computador Intel Core2 Quad 2.83GHz, 4GB de memória RAM, sistema operacional Windows de 64 bits. Foram executados 20 experimentos de 1000 gerações, para cada um dos dois algoritmos. Os resultados estatísticos em termos de média, mediana, desvio padrão, máximo e mínimo de cada um dos objetivos (equações 36 a 41) e da função de avaliação (equação 43) estão mostrados nas Tabelas 13 e 14. Estes resultados também são apresentados no boxplot da Figura 17.

A Figura 18 mostra a evolução do objetivo global do modelo, o *fitness* do melhor indivíduo para o experimento 1 (primeira simulação com 1000 gerações), para ambos algoritmos durante as gerações.



(a) μAG



(b) SFLA

Figura 17: Resultados estatísticos de 20 simulações

A Tabela 15 apresenta o desempenho por objetivo do modelo, para o melhor indivíduo do experimento 1, para o algoritmo μAG e o algoritmo SFLA modificado.

Tabela 15: Valor do *Fitness* para todos os objetivos

Algoritmo/Objetivo	Global	1	2	3	4
μAG	0.20	0	0	0.60	0.21
SFLA	0.16	0	0	0.54	0.03

Os dois algoritmos utilizados apresentaram uma rápida convergência na solução do modelo. Pode se notar que para os objetivos 1 e 2, foi encontrada uma solução ótima, isto é, as demandas de recebimento e de fornecimento foram atendidas pelos dois algoritmos. Atender as demandas é muito importante na solução do problema de redes de distribuição, pois é indesejável paralisar uma produção por falta de parque de armazenagem ou deixar faltar produtos nos pontos solicitantes (terminais).

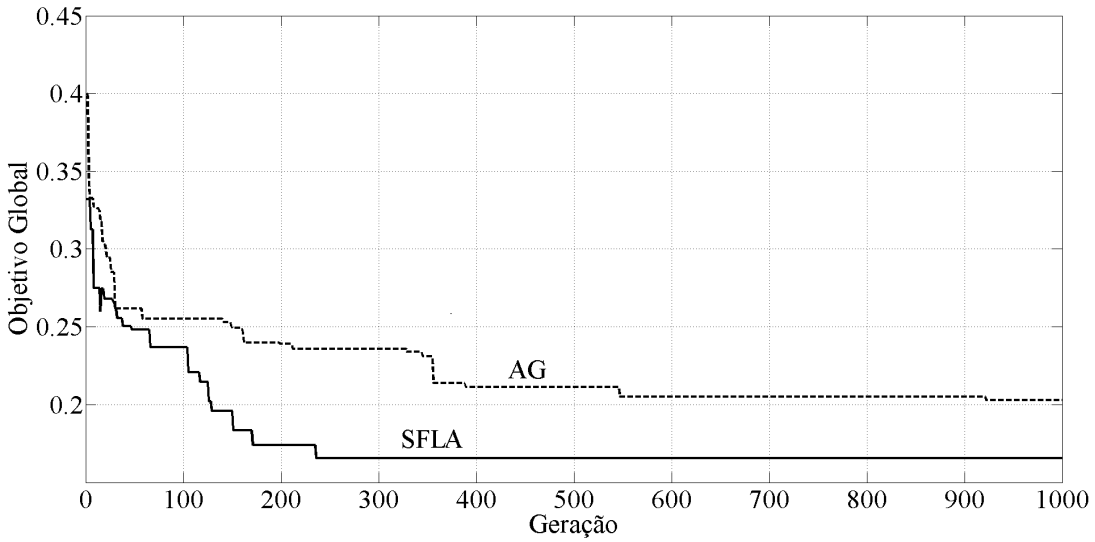


Figura 18: Evolução do *fitness* para o objetivo global

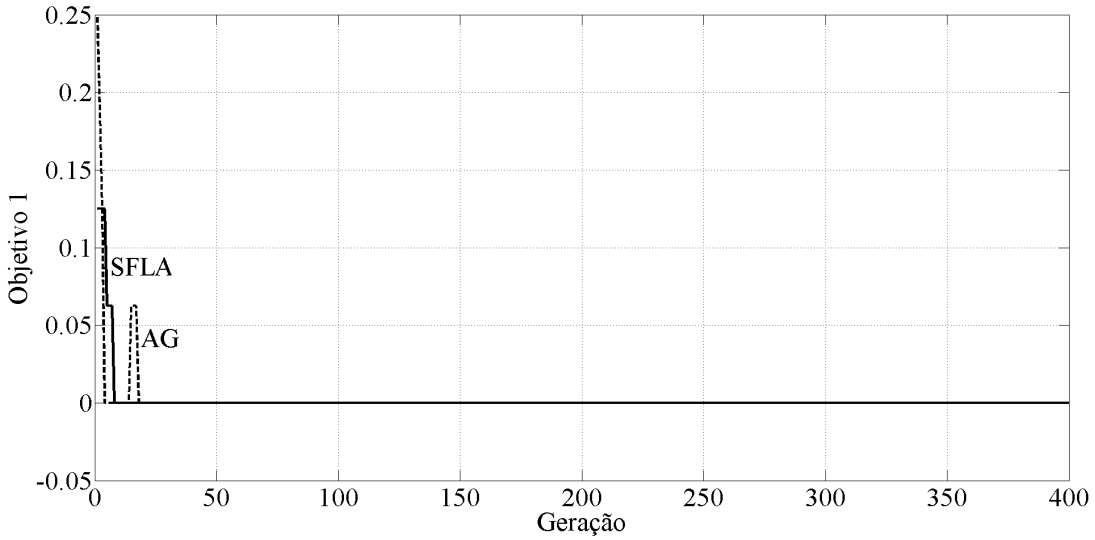


Figura 19: Evolução do *fitness* para o objetivo 1

Atingir o valor zero para o objetivo 3, a minimização do tempo, não é possível, pois fisicamente é inviável que todos os produtos de um terminal sejam recebidos ao mesmo tempo com apenas uma conexão. Já para o objetivo 4, a minimização da fragmentação das bateladas, o algoritmo SFLA chegou próximo de uma solução ótima.

O algoritmo SFLA apresentou uma melhor convergência em todos os objetivos do modelo, alcançando melhores resultados na otimização do problema em questão. Esta rápida convergência pode ser associada à busca local utilizada por este algoritmo.

Como mencionado anteriormente, o SFLA usa a evolução memética sob a forma de infecção de idéias entre os indivíduos na busca local, e uma estratégia de embaralhamento que permite a troca de informações entre as buscas locais para se mover em direção a um ótimo

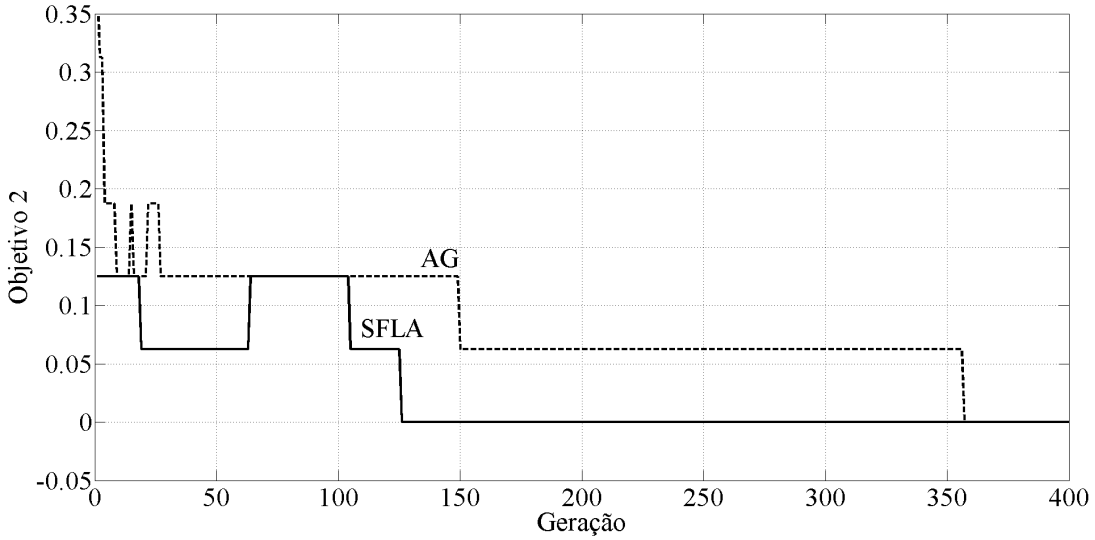


Figura 20: Evolução do *fitness* para o objetivo 2

global.

No μAG, a transmissão dos genes normalmente ocorre entre as gerações. Na evolução memética, se uma ideia melhor é encontrada, ela pode ser incorporada em outros memes imediatamente, em vez de esperar para uma geração completa de genes a serem replicados. Além disso, a replicação do gene é limitada pelo número relativamente pequeno de descendentes que um único pai pode ter, ao passo que o número de indivíduos que podem assumir um meme a partir de um único indivíduo é quase ilimitado. Outra diferença entre memes e genes é que os memes são processados e possivelmente melhorados pelo indivíduo que os detém, algo que não pode acontecer com os genes (EUSUFF et al., 2006).

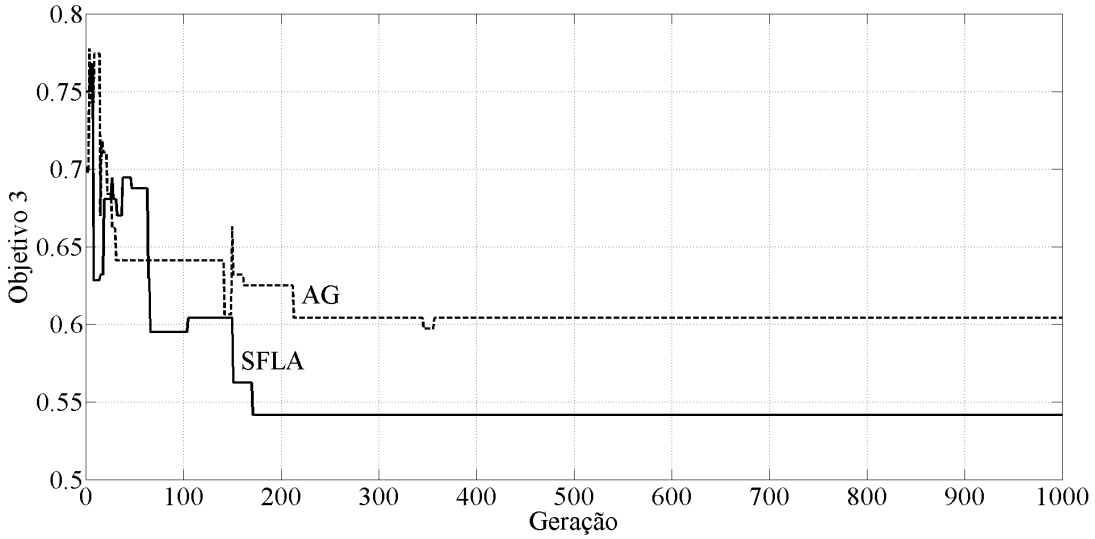


Figura 21: Evolução do *fitness* para o objetivo 3

As Figuras 19, 20, 21 e 22 mostram a evolução de cada objetivo considerado no

modelo, o valor do *fitness* do melhor indivíduo do experimento 1, para ambos algoritmos durantes as gerações.

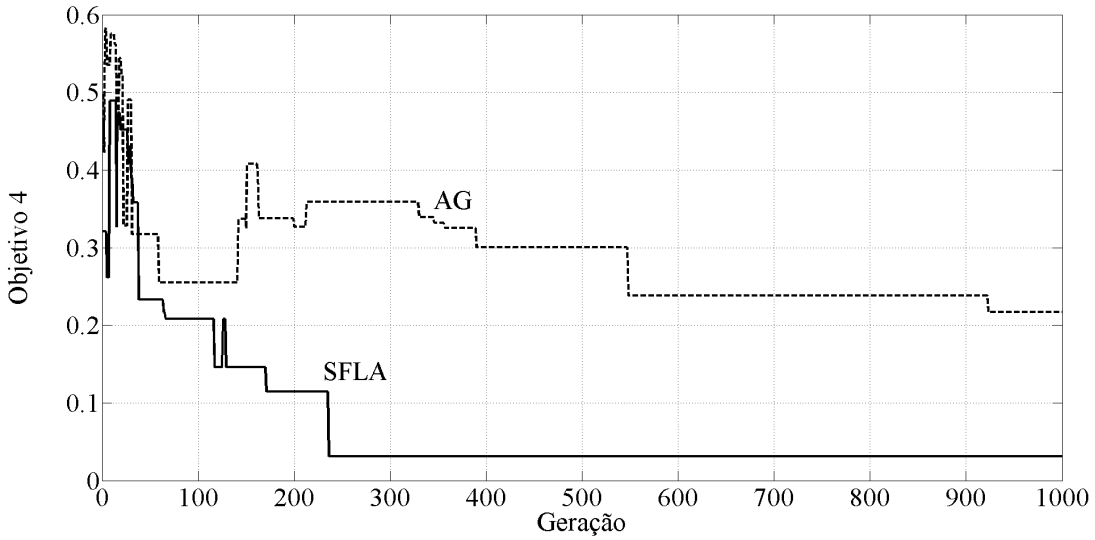


Figura 22: Evolução do *fitness* para o objetivo 4

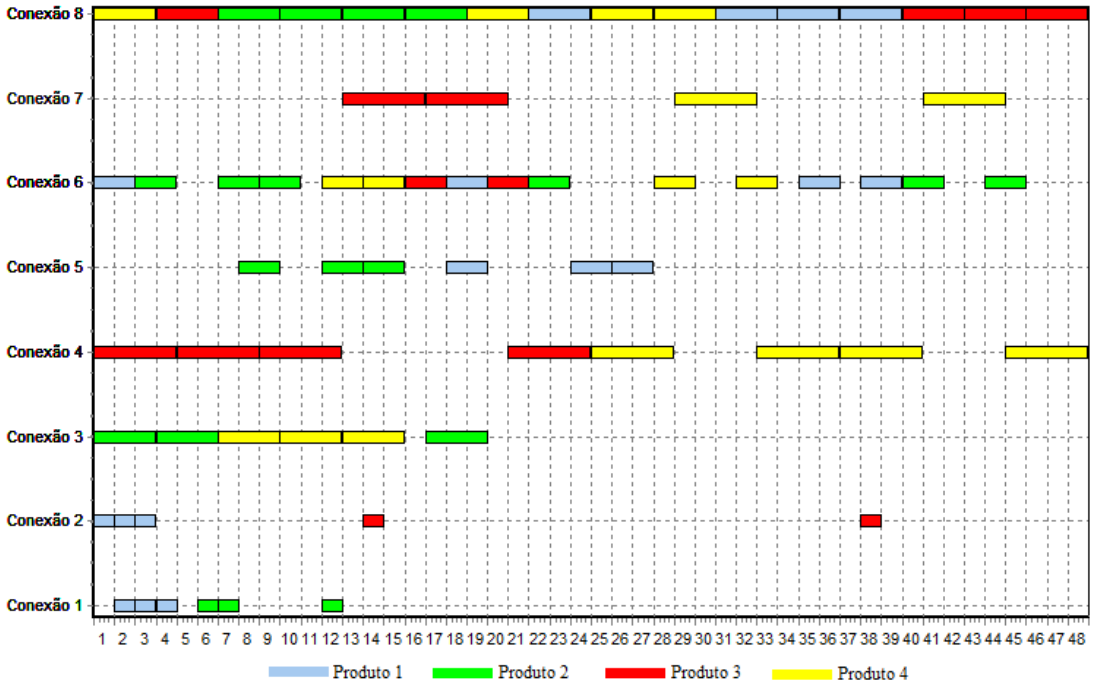
Para fins de comparação, a Figura 23 apresenta a carta de Gantt da solução do melhor indivíduo do experimento 1 ao longo do horizonte de programação utilizando o algoritmo μAG e o algoritmo SFLA, respectivamente. Nota-se que a conexão 8 é a mais ocupada durante todo o horizonte de tempo, pois para rede utilizada no modelo, o principal problema é atender a demanda no nó 8, que é o porto de Santos.

A minimização da fragmentação das bateladas alcançada pelo algoritmo SFLA também pode ser notada na Figura 23(b). A fragmentação ocorre quando em uma sequencia de bateladas diferentes produtos são enviados alternadamente, ao invés de ocorrer o envio de um mesmo produto sequencialmente. A alternância no tipo de produto a ser enviado pode provocar contaminações nas fronteiras, e por isso a fragmentação de bateladas em determinadas situações não é desejada.

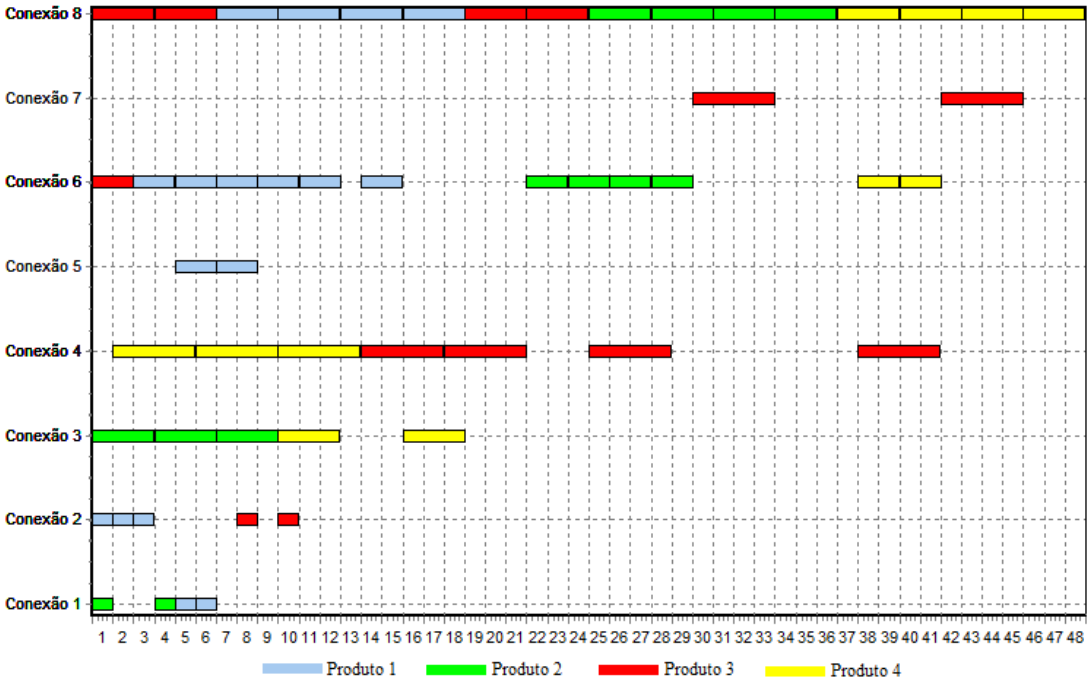
O tempo de execução de ambos algoritmos para encontrar a solução do modelo utilizado nestas simulações, não foi apresentado devido a resposta da solução ser encontrada em poucos segundos.

6.2 CENÁRIO 1

Para apresentar resultados com o modelo mais próximo da rede real, a quantidade de produtos foi dobrada, passando a ser 8 produtos. Para este novo conjunto de simulações, os produtos produzidos pela refinarias são configurados conforme os valores apresentados na



(a) μAG



(b) SFLA

Figura 23: Distribuição dos produtos ao longo do tempo

Tabela 16.

A Tabela 17 apresenta as configurações das demandas de fornecimento e recebimento de cada produto para os nós fontes e para o nó terminal.

O número de gerações também foi dobrada, assim foram executados 20 experimentos

Tabela 16: Produtos produzidos por cada refinaria

Refinaria/Nó	Produtos
N1	1,4,5,8
N2	2,3,6,7
N3	1,4,5,8
N4	2,3,6,7

Tabela 17: Demanda de Fornecimento e Recebimento

Demanda	**Nó**	P1	P2	P3	P4	P5	P6	P7	P8
Fornecimento	N1	2	-	-	2	2	-	-	2
	N2	-	2	2	-	-	2	2	-
	N3	2	-	-	2	2	-	-	2
	N4	-	2	2	-	-	2	2	-
Recebimento	N8	2	1	1	2	2	2	1	1

de 2000 gerações, para cada um dos dois algoritmos.

O horizonte de programação das simulações a serem apresentadas também é de 48 unidades de tempo, ou seja, o último recebimento de produto possível será na unidade de tempo 48 e nenhum envio poderá ocorrer após esse tempo.

Ressaltando que os resultados, para cada objetivo, ficam dentro da faixa $[0;1]$ de tal forma que 0 é o valor ótimo, ou seja, todos os critérios foram cumpridos. Qualquer valor entre 0 e 1 mostra que o critério foi atendido parcialmente.

Os resultados estatísticos em termos de média, mediana, desvio padrão, máximo e mínimo de cada um dos objetivos (equações 36 a 41) e da função de avaliação (equação 43) são apontados nas Tabelas 18 e 19. Estes resultados também são apresentados no *boxplot* da Figura 24.

Tabela 18: Resultados para um conjunto de 20 simulações para o algoritmo μAG

Objetivo	Mínimo	Máximo	Média	Mediana	Desvio Padrão
1	0	0	0	0	0
2	0	0	0	0	0
3	0.3119	0.3714	0.3359	0.3359	0.0163
4	0.2630	0.4037	0.3453	0.3415	0.0344
Global	0.1234	0.1484	0.1353	0.1349	0.0063

Para este estudo de caso, ambos algoritmos utilizados novamente apresentaram uma boa convergência na solução do modelo. Foram encontradas soluções ótimas para os objetivos 1 e 2, isto é, as demandas de recebimento e de fornecimento foram atendidas pelos dois algoritmos. Como já mencionado, atingir o valor zero para o objetivo 3, a minimização do tempo, não é possível, pois fisicamente é inviável que todos os produtos de um terminal sejam

Tabela 19: Resultados para um conjunto de 20 simulações para o algoritmo SFLA

Objetivo	Mínimo	Máximo	Média	Mediana	Desvio Padrão
1	0	0	0	0	0
2	0	0	0	0	0
3	0.3083	0.3641	0.3348	0.3368	0.0150
4	0.0714	0.2152	0.1428	0.1420	0.0466
Global	0.1055	0.1273	0.1147	0.1131	0.0072

recebidos ao mesmo tempo com apenas uma conexão.

Como pode ser visto nas Tabelas 18 e 19, o algoritmo SFLA novamente encontrou melhores resultados nas soluções dos objetivos quando comparado ao algoritmo μAG, minimização do tempo e a minimização da fragmentação das bateladas.

Para este conjunto de simulações a convergência de ambos algoritmos foi semelhante, como pode ser visto na Figura 25, que mostra a evolução do objetivo global do modelo, o *fitness* do melhor indivíduo para o experimento 1 (primeira simulação com 2000 gerações), para ambos algoritmos durante as gerações.

A Tabela 20 apresenta o desempenho por objetivo do modelo, para o melhor indivíduo do experimento 1, para o algoritmo μAG e o algoritmo SFLA modificado.

Tabela 20: Valor do *Fitness* para todos os objetivos

Algoritmo/Objetivo	Global	1	2	3	4
μAG	0.1317	0	0	0.337	0.306
SFLA	0.1058	0	0	0.321	0.093

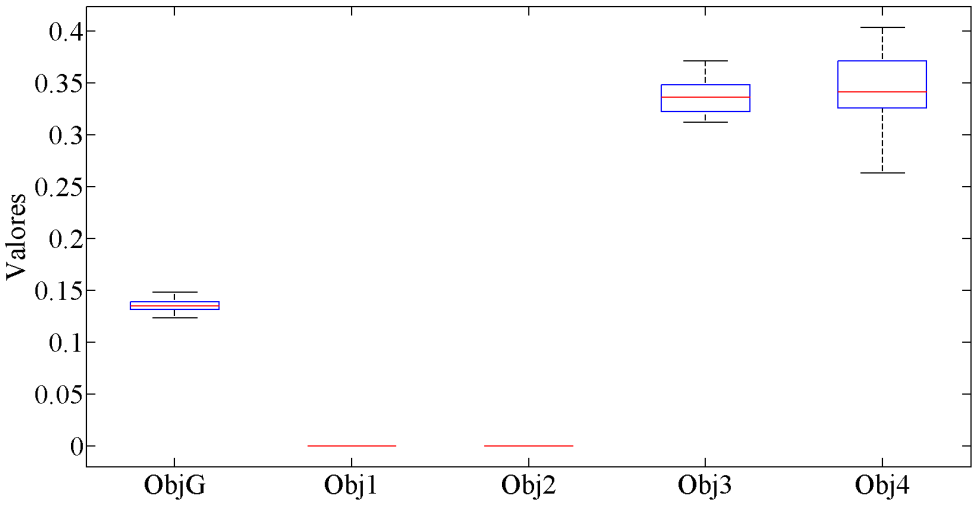
A Figura 26 apresenta a carta de Gantt da solução do melhor indivíduo do experimento 1 ao longo do horizonte de programação utilizando o algoritmo SFLA e o algoritmo μAG, respectivamente.

A media de tempo de execução de ambos algoritmos para o conjunto de 20 simulações estão mostrados na Tabela 21, assim como a quantidade de soluções não dominadas encontradas nas 20 execuções dos algoritmos.

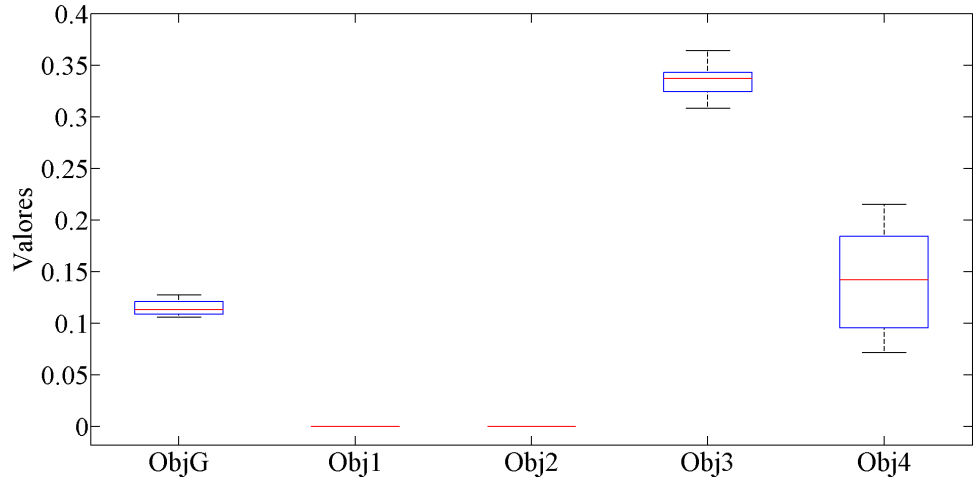
Tabela 21: Tempo de execução e quantidade de soluções de pareto

Algoritmo	Tempo médio de execução	Soluções de Pareto
μAG	2.80s	34
SFLA	24.87s	25

O tempo médio de execução do algoritmo SFLA é pior que o tempo médio do algoritmo μAG, pois o algoritmo SFLA possui uma complexidade maior que o algoritmo μAG.



(a) μAG



(b) SFLA

Figura 24: Resultados estatísticos de 20 simulações

O SFLA utiliza uma busca local conforme mencionado anteriormente, o que faz seu tempo de execução ficar maior em relação ao μAG. Porém, o SFLA encontrou resultados melhores em todos os objetivos para os dois conjuntos de simulações apresentados, tanto para o modelo com 4 produtos quanto para o modelo com 8 produtos.

Todas as soluções obtidas em cada simulação para cada um dos modelos são armazenadas em um conjunto de soluções viáveis. Estes conjuntos de soluções finais viáveis calculados para 20 simulações de ambos os modelos são mostrados na Figura 27. O modelo μAG encontrou 34 soluções diferentes e o modelo SFLA encontrou 25 soluções diferentes.

Embora a cada simulação ambos algoritmos calculem as soluções não-dominadas, o conjunto final resultante das 20 simulações podem conter soluções dominadas. Assim, uma análise entre as soluções utilizando o conceito de não-dominância é considerada a fim de formar

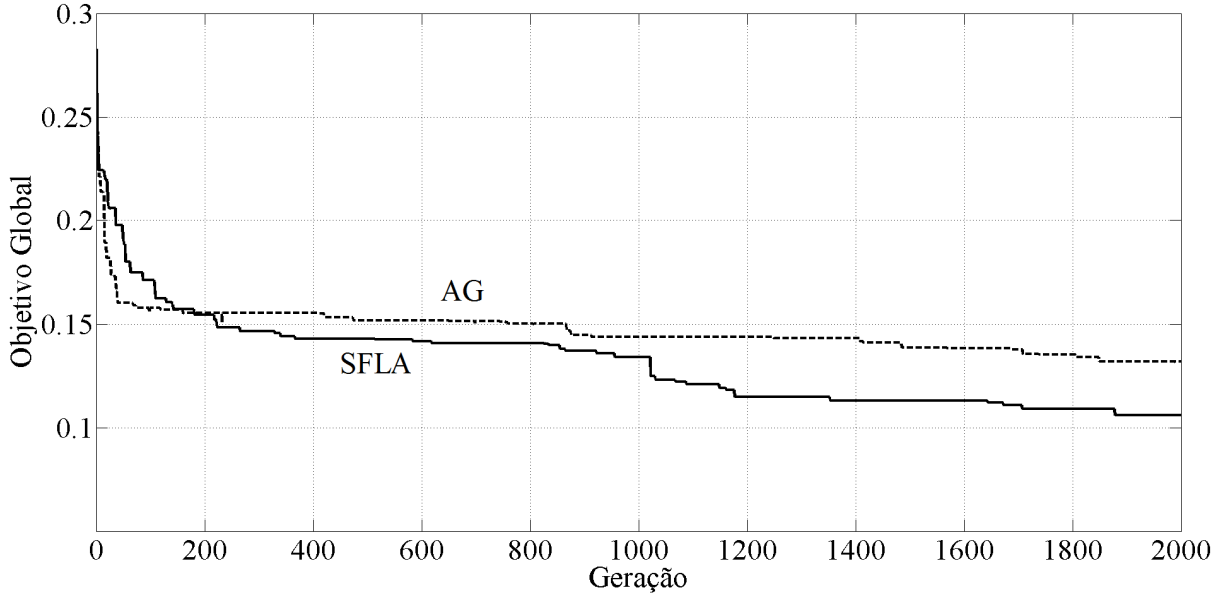


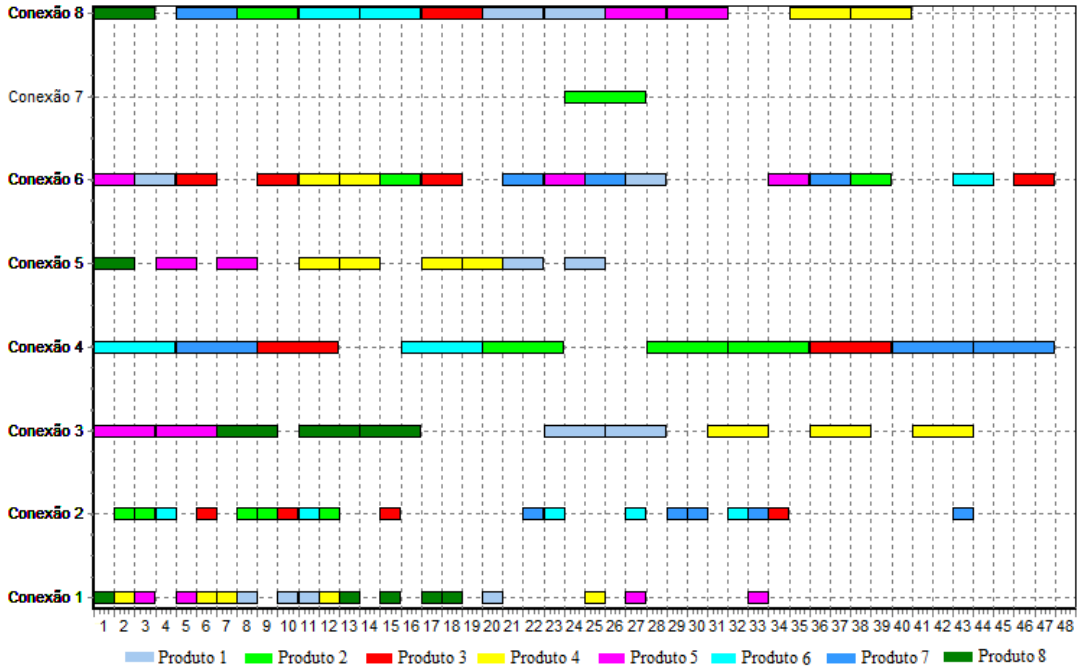
Figura 25: Evolução do *fitness* para o objetivo global

um conjunto ótimo de Pareto e permitir a construção da fronteira de Pareto para o problema. Para o modelo μAG, apenas 2 soluções não-dominadas são encontrados entre as 34 soluções do conjunto. Isto é, o modelo μAG explorou a mesma região do espaço de busca problema, e ele sempre alcança soluções semelhantes ao longo de 20 simulações. Para o modelo SFLA, 5 soluções não-dominadas são encontrados entre as 25 soluções viáveis. Na Tabela 22 são apresentadas estas soluções para ambos algoritmos.

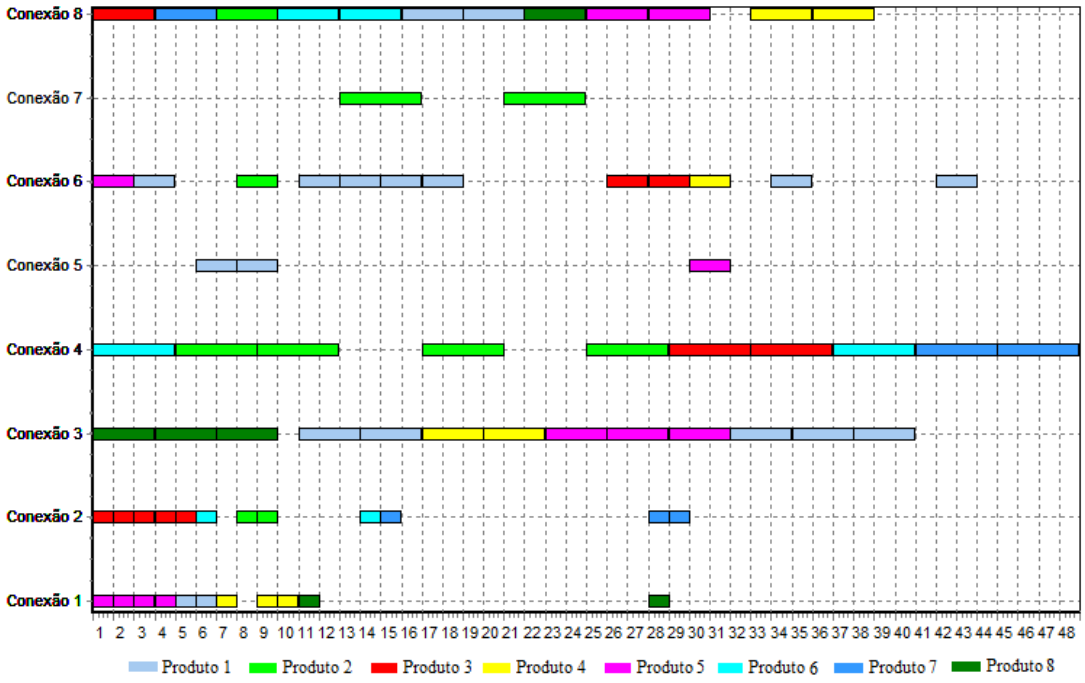
Tabela 22: Conjunto de solução não dominadas

Soluções	Algoritmo	Demanda de recebimento	Demanda de fornecimento	Minimização do tempo	Minimização da fragmentação
1	μAG	0	0	0.312	0.3121
2		0	0	0.3238	0.2630
1	SFLA	0	0	0.3083	0.1593
2		0	0	0.3089	0.1281
3		0	0	0.3214	0.0937
4		0	0	0.334	0.0870
5		0	0	0.3416	0.0714

A fronteira de Pareto é formada pelos pontos no espaço das funções objetivo que corresponde ao conjunto Pareto-ótimo. Para representar e visualizar a fronteira de Pareto do conjunto das soluções apresentadas na Tabela 22, são desconsiderados os dois primeiros objetivos que são todos zeros e plotados no espaço apenas os objetivos 3 e 4. Esta representação é mostrada na Figura 28, e como pode ser observado nesta figura as soluções do μAG são todas dominadas pelas soluções do SFLA.



(a) μAG

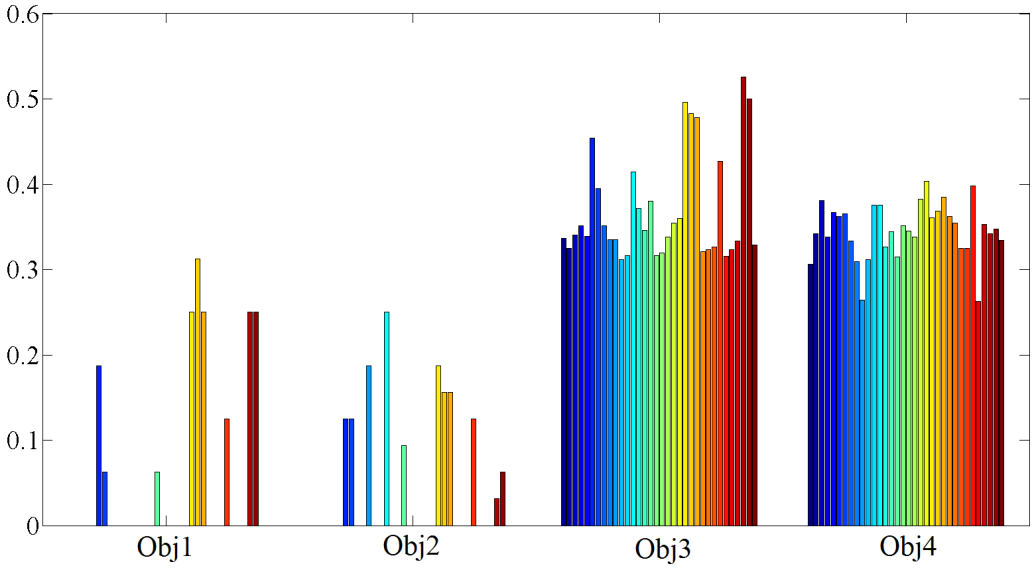


(b) SFLA

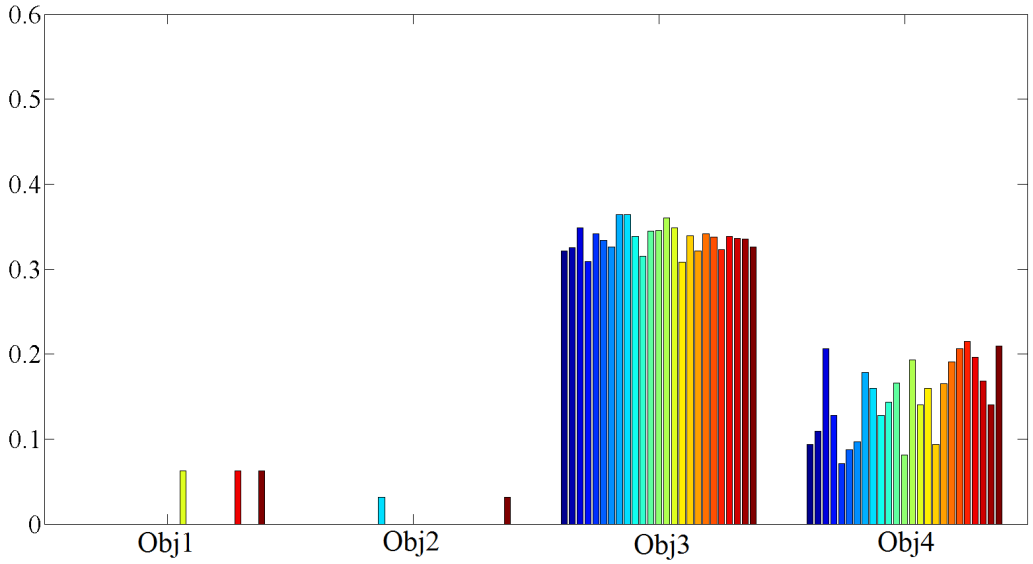
Figura 26: Distribuição dos produtos ao longo do tempo

Como o foco deste trabalho é a solução de um problema de otimização multiobjetivo, obter um conjunto de soluções Pareto-ótimas que construa uma fronteira de Pareto que possa representar razoavelmente o problema no espaço das funções objetivos se torna essencial.

Dessa forma, conforme visto na Figura 28, tanto o SFLA quanto o μAG apresentaram



(a) μAG



(b) SFLA

Figura 27: Soluções não dominadas para as 20 simulações.

um conjunto pequeno de soluções não-dominadas para construir e representar a fronteira de Pareto, surgindo a necessidade de alternativas para encontrar um conjunto maior e representativo para a fronteira de Pareto. Assim, este trabalho propôs uma nova abordagem utilizando o algoritmo SFLA através do conceito de otimalidade de Pareto, chamado de *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA). O principal objetivo desse novo algoritmo é representar e recuperar uma grande parte da fronteira de Pareto para o problema modelado. Na próxima seção os resultados preliminares obtidos com esta nova abordagem.

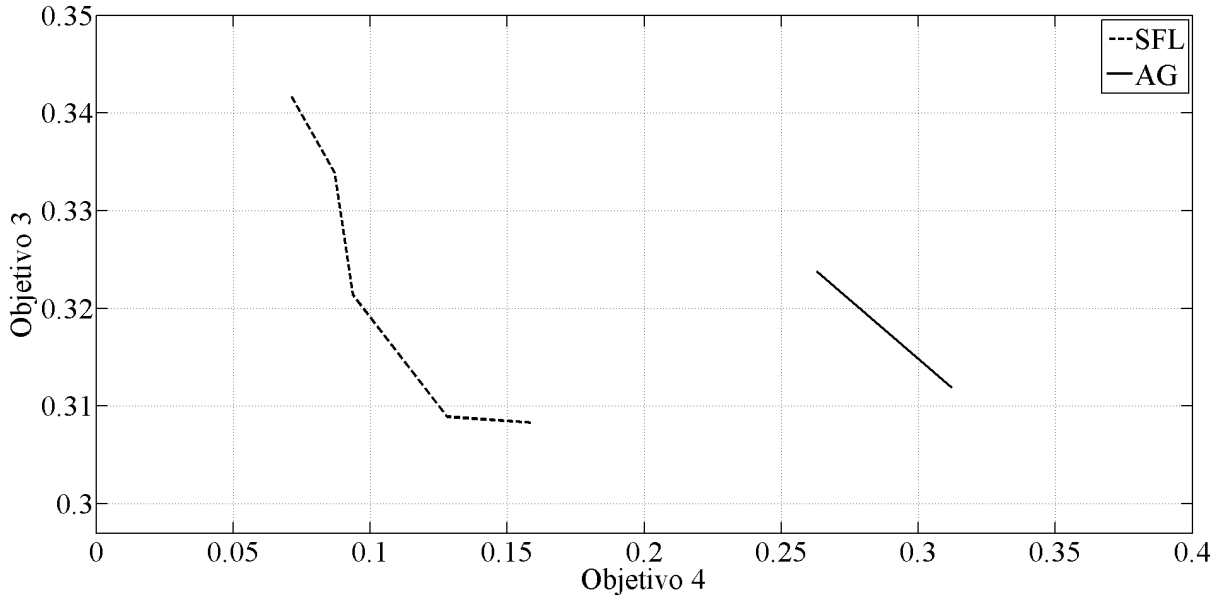


Figura 28: Representação da fronteira de Pareto

6.3 RESULTADOS COM MSFLPA

Conforme descrito na Seção 4.1, o MSFLPA utiliza algumas estratégias para favorecer a busca de diferentes pontos na fronteira de Pareto ao longo das gerações.

O algoritmo utiliza duas memórias auxiliares: i) a primeira é denominada conjunto temporário de soluções não dominadas e é usada para guardar as soluções não dominadas de um conjunto de soluções da evolução de um determinado número de gerações; e ii) a segunda é uma memória externa, denominada conjunto final de soluções não dominadas, que é utilizada para guardar o conjunto final de soluções de Pareto encontradas ao longo de todas as evoluções a partir da reinicialização da população.

Na segunda memória externa, as soluções armazenadas são o conjunto de todas as soluções, cujos vetores correspondentes são não-dominados quando comparados a todos os outros vetores antes armazenados. E esses vetores não-dominados quando plotados no espaço das funções objetivo formam a fronteira de Pareto.

Para o algoritmo MSFLPA, a reinicialização da população é feita por um parâmetro que verifica se não há mudança na evolução durante um determinado número de gerações. Se alcançar esse número sem haver nenhuma melhora ou mudança na evolução, a população é reinicializada.

Os resultados foram obtidos realizando 20 experimentos de 2000 gerações para ambos modelos apresentados, o modelo com 4 produtos e modelo com 8 produtos, que serão

representados como MSFLPA-4 e MSFLPA-8.

6.3.1 MODELO COM 4 PRODUTOS

Os resultados apresentados mantém as mesmas configurações de horizonte de programação, configuração de produtos, demanda de recebimento e fornecimento para modelo com 4 produtos apresentando anteriormente e os mesmos parâmetros do SFLA.

Destacando que os parâmetros de configuração do algoritmo MSFLPA para o modelo com 4 produtos são:

✓ F (população) = 20;

✓ m (*memeplexes*) = 4;

✓ N (evoluções) = 5;

✓ C (fator de aceleração) = 1.7.

As Figuras 29 e 30 mostram o conjunto de soluções Pareto-ótimas encontradas no experimento que obteve a maior quantidade de soluções não dominadas, com o algoritmo MSFLPA-4. Foram encontradas para este experimento um conjunto de 12 soluções não dominadas.

Os resultados apresentados na Tabela 23 mostram que o MSFLPA-4, além de encontrar um conjunto de soluções Pareto-ótima representativo, obteve bons resultados nas solução dos objetivos do modelo.

O MSFLPA-4 apresentou uma boa convergência na solução do modelo e também encontrou soluções ótimas para os objetivos 1 e 2, isto é, as demandas de recebimento e de fornecimento foram atendidas pelo algoritmo. Para os objetivos 3 e 4, o MSFLPA-4 também encontrou boas soluções se comparados aos algoritmos μAG e o SFLA anteriormente utilizado.

Como visualizar a fronteira de pareto no espaço de soluções com 4 objetivos fica impossível, uma alternativa é retirar as quatro primeiras soluções do conjunto da Tabela 23 e representar as outras soluções no espaço, utilizando apenas os objetivos 3 e 4, retirando ou outros dois objetivos que tem valor zero. Esta representação pode ser vista na Figura 31.

Tabela 23: Conjunto de solução não dominadas para o MSFLPA-4

Soluções	*Fitness* Global	Demanda de recebimento	Demanda de fornecimento	Minimização do tempo	Minimização da fragmentação
1	0.1965	0	0.125	0.4855	0.1343
2	0.1827	0.187	0	0.4217	0
3	0.1823	0.083	0	0.4797	0.1343
4	0.1821	0	0.062	0.5	0.1343
5	0.1729	0	0	0.5598	0.05
6	0.1700	0	0	0.5192	0.1428
7	0.1699	0	0	0.5320	0.1026
8	0.1683	0	0	0.5208	0.1214
9	0.1663	0	0	0.5336	0.0625
10	0.1560	0	0	0.4679	0.1562
11	0.1552	0	0	0.4487	0.2062
12	0.1541	0	0	0.4556	0.175

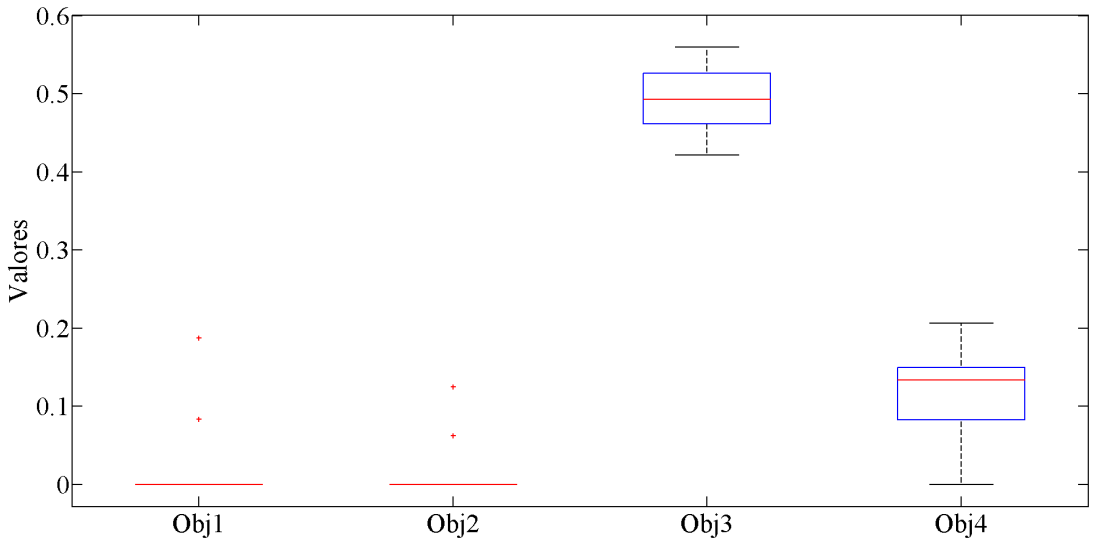


Figura 29: Boxplot para o conjunto de soluções não dominadas para o MSFLPA-4

6.3.2 MODELO COM 8 PRODUTOS

Os resultados apresentados mantém as mesmas configurações de horizonte de programação, configuração de produtos, demanda de recebimento e fornecimento para modelo com 8 produtos apresentando anteriormente e um parâmetro do SFLA foi alterado.

Os parâmetros de configuração do algoritmo MSFLPA para o modelo com 8 produtos são:

✓ F (população) = 20;

✓ m (*memeplexes*) = 4;

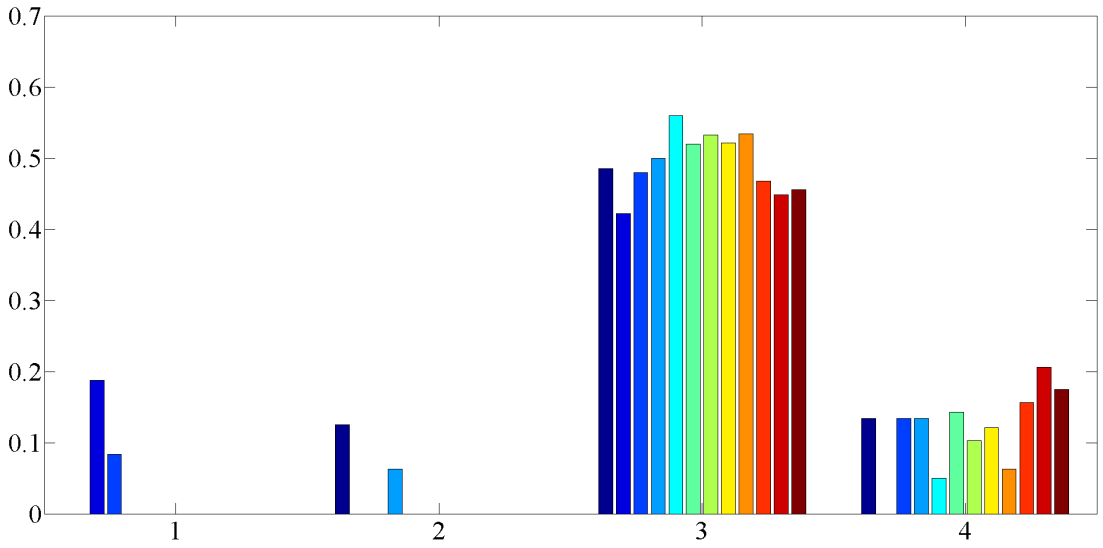


Figura 30: Soluções não dominadas para o MSFLPA-4

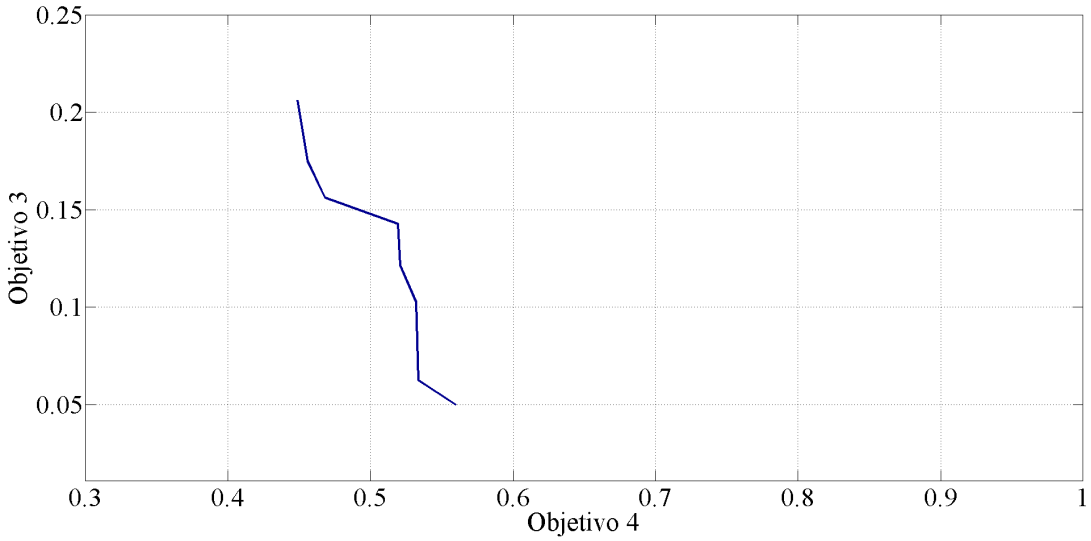


Figura 31: Fronteira de Pareto do conjunto de soluções não dominadas para o MSFLPA-4

✓ N (evoluções) = 15;

✓ C (fator de aceleração) = 1.7.

As Figuras 32 e 33 mostram o conjunto de soluções Pareto-ótimas encontradas no experimento que obteve a maior quantidade de soluções não dominadas, com o algoritmo MSFLPA-8. Foram encontradas para este experimento um conjunto de 15 soluções não dominadas.

Os resultados apresentados na Tabela 24 mostram que o MSFLPA-8, também para este modelo encontrou um conjunto de soluções Pareto-ótima representativo e obteve bons resultados nas solução dos objetivos do modelo.

Tabela 24: Conjunto de solução não dominadas para o MSFLPA-8

Soluções	*Fitness* Global	Demanda de recebimento	Demanda de fornecimento	Minimização do tempo	Minimização da fragmentação
1	0.1621	0.125	0	0.3550	0.1812
2	0.1620	0.125	0	0.3580	0.1715
3	0.1609	0.125	0	0.3595	0.1559
4	0.1503	0.062	0	0.3708	0.2031
5	0.1487	0.062	0	0.3616	0.2151
6	0.1482	0	0.031	0.3684	0.2831
7	0.1468	0	0.062	0.3089	0.3544
6	0.1439	0.062	0	0.3133	0.3117
9	0.1434	0	0.062	0.3178	0.2929
10	0.1433	0	0.062	0.3208	0.2832
11	0.1397	0	0.031	0.3327	0.3055
12	0.1396	0.062	0	0.3211	0.2458
13	0.1364	0	0.062	0.3297	0.1875
14	0.1361	0	0.031	0.3178	0.3144
15	0.1343	0	0	0.3392	0.3259

O MSFLPA-8 para este modelo também apresentou uma boa convergência na solução do modelo e encontrou soluções ótimas para os objetivos 1 e 2, isto é, as demandas de recebimento e de fornecimento foram atendidas. Assim como obteve bons resultados nos objetivos 3 e 4 se comparados aos algoritmos μAG e o SFLA anteriormente utilizado.

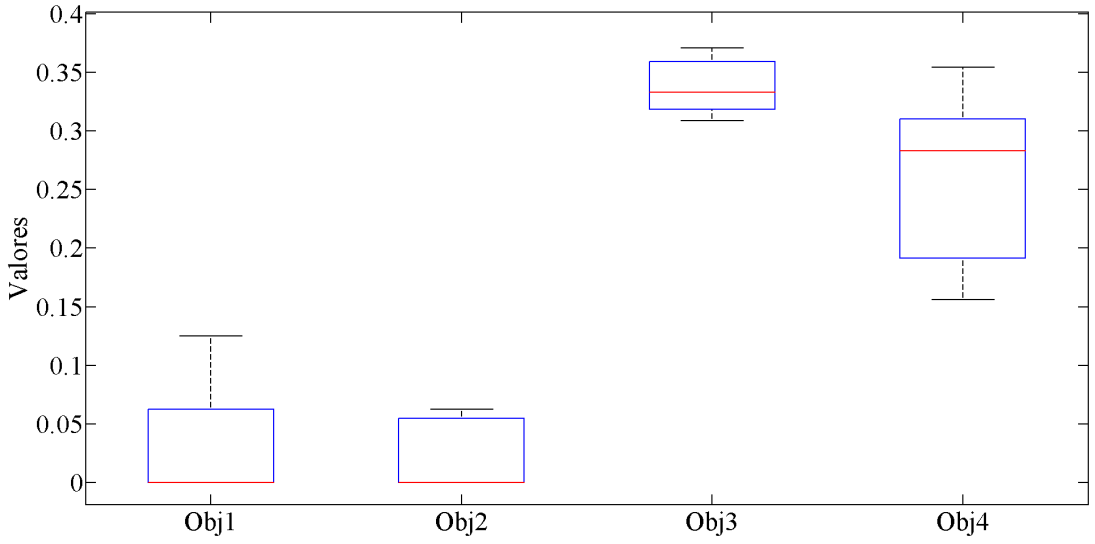


Figura 32: Boxplot para o conjunto de soluções não dominadas para o MSFLPA-8

A Tabela 25 apresenta os resultados obtidos do algoritmo MSFLPA para as 20 execuções, de 2000 gerações cada, de ambos os modelos. O MSFLPA encontrou uma média de aproximadamente 5 soluções Pareto-ótimas para o modelo com 4 produtos e uma média de 8 soluções Pareto-ótimas para o modelo de 8 produtos.

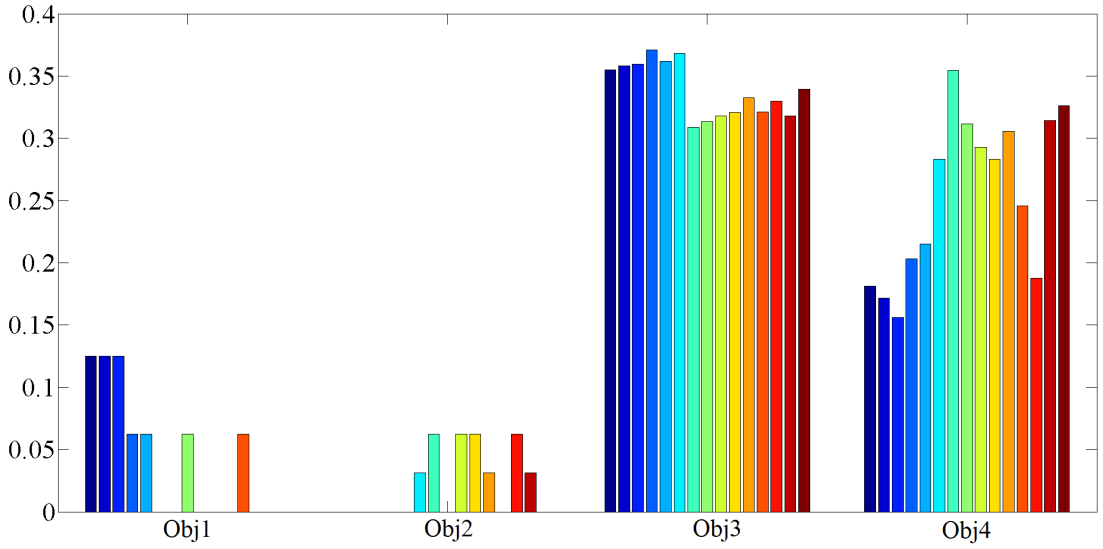


Figura 33: Soluções não dominadas para o MSFLPA-8

O tempo de execução médio apresentado na Tabela 25 mostra que o MSFLPA obteve bons resultados em um tempo consideravelmente bom para ambos modelos. O MSFLPA-4 teve uma média tempo de 18 segundos e o MSFLPA-8 obteve uma média de 65 segundos.

Pode-se observar na Tabela 25 a melhora do novo algoritmo em relação ao conjunto de soluções de Pareto encontrado a cada simulação quando comparado aos resultados encontrados na Seção 6.2, na Tabela 21. Enquanto o SFLA e o μAG no cenário com 8 produtos encontraram 25 soluções e 34 soluções de Pareto em 20 simulações, respectivamente o MSFLPA encontrou 165 no total das simulações.

Portanto, a partir desses resultados preliminares apresentados, pode-se concluir que a nova abordagem, o MSFLPA, obteve bons resultados em um tempo razoável. O algoritmo conseguiu, utilizando as estratégias propostas, favorecer a busca de diferentes pontos na fronteira de Pareto, aumentando a quantidade de soluções não-dominadas encontradas em apenas uma execução.

6.4 CONSIDERAÇÕES

Os resultados apresentados na Seção 6.2, mostram a eficiência do algoritmo SFLA quando comparado com o μAG. O algoritmo SFLA apresentou uma rápida convergência e uma busca local eficiente, obtendo melhores resultados na solução do cenário apresentado. Mas ambos algoritmos não obtiveram um conjunto de Pareto representativo para o problema modelado.

Tabela 25: Soluções de pareto para todas as simulações

Simulações	MSFLPA (4 produtos)	MSFLPA (8 produtos)
1	7	9
2	6	4
3	4	2
4	1	10
5	7	6
6	3	5
7	12	13
8	6	7
9	4	7
10	2	10
11	1	4
12	8	4
13	7	11
14	6	14
15	5	15
16	5	9
17	7	10
18	2	12
19	2	6
20	4	7
Total de soluções	99	165
Média de soluções	4.95	8.25
Média de tempo de execução	18.36s	65.17s

A partir disso, a nova abordagem proposta, o MSFLPA, que utiliza uma estratégia de arquivamento combinadas com as vantagens de uma rápida convergência de SFLA, encontrou conjuntos representativos de solução Pareto-ótimas para as execuções dos modelos e boas soluções para os objetivos na resolução do modelo multiobjetivo.

O próximo capítulo apresenta os resultados da comparação da nova abordagem MSFLPA com dois algoritmos utilizados para as mesmas classes de problemas estudada nesta tese: o NSGA-II e μAG.

7 RESULTADOS DA COMPARAÇÃO ENTRE OS ALGORITMOS MSFLPA, NSGA-II E μAG

Este capítulo apresenta os resultados da comparação de um cenário teste para a redes de dutos real entre os modelos multiobjetivos utilizando três algoritmos: a abordagem proposta MSFLPA, o NSGA-II e μAG.

7.1 APLICAÇÃO DA OTIMIZAÇÃO MULTIOBJETIVO PARA UMA REDE DE DUTOS REAL

Para realizar a comparação entre os algoritmos foi utilizado um cenário com oito produtos da rede de escuros do sistema dutoviário brasileiro. O desempenho do *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA) é comparado com o μAG algoritmo proposto por (RIBAS et al., 2013; ARRUDA et al., 2008) e o NSGA-II proposto por (DEB et al., 2002a). O objetivo desta comparação é validar a eficácia e adequação do MSFLPA quando aplicado para resolver problemas de escalonamento de curto prazo em redes de dutos, que são problemas de otimização complexos.

Os modelos evolucionários (MSFLPA, μAG e NSGA-II) adotam a codificação da solução descrito no Capítulo 5. O procedimento utilizado para reparar as soluções inviáveis também é o mesmo. Além disso, os modelos evolucionários estão procurando satisfazer simultaneamente os vários objetivos dadas pelas equações 36 para 41 do Capítulo 5 que correspondem as seguintes funções objetivos:

$f_1(x)$ Receber a quantidade de bateladas demandada pelos terminais;

$f_2(x)$ Satisfazer a produção mínima;

$f_3(x)$ Minimizar o tempo de entrega das bateladas;

$f_4(x)$ Minimizar a fragmentação das bateladas.

A otimização multiobjetivo leva em consideração apenas o valor global de cada objetivo, calculado pela equação 43. Os pesos associados a cada objetivo nesta equação são

dados na Tabela 26. O objetivo de minimização da fragmentação no envio das bateladas possui um peso associado menor, pois ele pode prejudicar o desempenho de otimização dos outros objetivos. Contudo, evitar a fragmentação das bateladas é um critério muito importante, mas não é prioridade nas designações das atividades operacionais da rede de dutos conforme os especialistas.

Tabela 26: Pesos utilizados na função objetivo

Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Peso	3	3	3	1

O horizonte de programação das simulações a serem apresentadas é de 48 unidades de tempo, ou seja, o último recebimento de produto possível será na unidade de tempo 48 e nenhum envio poderá ocorrer após esse tempo. O tempo tem representação discreta.

Para executar as simulações, o volume máximo e mínimo de cada tanque em cada área deve ser definida, e também a duração de cada conexão deve ser configurado em termos de intervalos de tempo. Estes parâmetros de configuração da rede são indicados na Tabela 7.

Os produtos produzidos pela refinarias podem ser configurados conforme desejado, as designações dos produtos produzidos por cada refinaria estão apresentados na Tabela 27.

Tabela 27: Produtos produzidos por cada refinaria

Refinaria/Nó	Produtos
N1	1,4,5,8
N2	2,3,6,7
N3	1,4,5,8
N4	2,3,6,7

As demandas de fornecimento são configuradas através de uma quantidade mínima de bateladas de cada produto que deve ser enviada a partir de cada refinaria. Assim como as demandas de recebimento são configuradas através da quantidade de bateladas de cada produto que um terminal deverá receber. Para a rede utilizada no modelo, o nó N8 é único terminal, e o objetivo geral da rede é atender a demanda desse nó. As Tabelas 28 e 29 apresentam as configurações das demandas.

Tabela 28: Produtos produzidos por cada refinaria

Nós	P1	P2	P3	P4	P5	P6	P7	P8
N1	2	-	-	2	2	-	-	2
N2	-	2	2	-	-	2	2	-
N3	2	-	-	2	2	-	-	2
N4	-	2	2	-	-	-	2	-

Tabela 29: Demanda no nó terminal (porto)

Produtos	P1	P2	P3	P4	P5	P6	P7	P8
N8	2	1	1	2	2	2	1	1

Todos os parâmetros de modelos evolucionários foram escolhidos após vários experimentos utilizando as diretrizes do (GREFENSTETTE, 1986). Estes parâmetros são mostrados para os três algoritmos na Tabela 30. Para o NSGA-II, foi utilizado o *simulated binary crossover* (SBX) e mutação polinomial (DEB; AGRAWAL, 1995) e os índices de distribuição para operadores de cruzamento e mutação como $N_c = 20$ e $N_m = 20$, respectivamente. Todos os modelos foram executados em um processador Intel Core 2 Quad, 2,83 GHz, 4 GB de RAM, com 64 bits plataforma Windows. Todos os modelos são codificadas em linguagem de programação C++.

Tabela 30: Parâmetros do MSFLPA, μAG e NSGA-II

Parâmetros	MSFLPA	μAG	NSGA-II
Tamanho da População	20	20	20
Número de gerações	2000	2000	2000
Memeplexes	4	-	-
Iterações evolucionária	15	-	-
Fator de Aceleração	1.7	-	-
Taxa de *Crossover*	-	0.8	0.9
Taxa de Mutação	-	0.1	0.01

Como as funções objetivos são normalizadas no intervalo $[0;1]$, o ótimo global da função de avaliação na equação 43 é 0 que corresponde ao cumprimento de todos os critérios. Se o valor final da função de avaliação for superior a zero, um ou mais objetivos são cumpridos parcialmente.

Os três modelos evolucionários são simulados com 20 populações iniciais diferentes (20 execuções) sobre 2000 gerações. Os resultados estatísticos em termos de média, mediana, desvio padrão, valores máximo e mínimo para cada função objetivo (equações 36 a 41) são mostrados nas Tabelas 31 e 32 para o modelo μAG, na tabela 33 para o modelo NSGA-II, na Tabela 34 para o modelo MSFLPA e a função de avaliação (equação 43) para os três modelos na Tabela 35. Estes resultados também são ilustrados nos boxplots de Figura 34.

A partir destes resultados apresentados nas tabelas, pode-se concluir que os algoritmos MSFLPA e μAG obtiveram melhores desempenhos do que o algoritmo NSGA-II no aspecto de convergência para uma solução viável. Os algoritmos MSFLPA e μAG encontraram o ponto ótimo para os dois primeiros objetivos $f_1(x)$ e $f_2(x)$, ou seja, as demandas de recebimento e de fornecimento foram totalmente cumpridas por ambos modelos.

Tabela 31: Resultados estatísticos para 20 simulações do modelo μAG

Função Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Mínimo	0	0	0.3119	0.2630
Máximo	0	0	0.3714	0.4037
Média	0	0	0.3359	0.3453
Mediana	0	0	0.3359	0.3415
Desvio Padrão	0	0	0.0163	0.0344

Tabela 32: Resultados estatísticos para 20 simulações do modelo μAG2 para 65.2s.

Função Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Mínimo	0	0	0.3173	0.1909
Máximo	0	0	0.3560	0.2832
Média	0	0	0.3373	0.2365
Mediana	0	0	0.3402	0.2362
Desvio Padrão	0	0	0.0144	0.0252

Tabela 33: Resultados estatísticos para 20 simulações do modelo NSGA-II

Função Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Mínimo	0.1250	0.0625	0.3557	0.1214
Máximo	0.3125	0.2813	0.4548	0.2156
Média	0.1906	0.1812	0.4054	0.1676
Mediana	0.1875	0.1875	0.4035	0.1717
Desvio Padrão	0.0516	0.0579	0.0254	0.0293

Tabela 34: Resultados estatísticos para 20 simulações do modelo MSFLPA

Função Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Mínimo	0	0	0.3089	0.0938
Máximo	0	0	0.3554	0.2804
Média	0	0	0.3324	0.2262
Mediana	0	0	0.3315	0.2272
Desvio Padrão	0	0	0.0130	0.0416

Tabela 35: Resultados estatísticos para 20 simulações

	MSFLPA	μAG	NSGA-II
	$F(x)$	$F(x)$	$F(x)$
Mínimo	0.1087	0.1234	0.2042
Máximo	0.1269	0.1484	0.2984
Média	0.1223	0.1353	0.2499
Mediana	0.1237	0.1349	0.2469
Desvio Padrão	0.0047	0.0063	0.0273

O algoritmo NSGA-II obteve um melhor resultado apenas na função objetivo $f_4(x)$, apesar da fragmentação das bateladas ser considerada importante não é prioridade nas operações de uma rede de dutos. Vale observar que uma solução que atenda a todas as demandas é

considerada uma boa solução para o escalonamento de rede de dutos, pois a falta de produtos para atender a demanda de mercado para qualquer cliente final deve ser prevenida. Além disso, não é desejável parar a produção da refinaria devido ao excesso de produto na área de tancagem.

A fim de obter uma comparação justa em relação ao tempo de execução (descrito na próxima seção) entre MSFLPA, μAG e o NSGA-II, foram feitas simulações com o μAG e NSGA-II por aproximadamente 65.2s permitindo o reinício da população. Para ser capaz de atingir aproximadamente o mesmo tempo de execução de MSFLPA, o μAG foi executado durante 58.000 gerações e o NSGA-II durante 25.000 gerações. Para o NSGA-II, os resultados foram muito próximos (quase iguais) aos apresentados na Tabela 33. Neste caso, os resultados não foram apresentados aqui. Para o μAG, a simulação será chamada de μAG2 para as referências em tabelas e figuras. Os resultados estatísticos para 20 execuções diferentes do μAG com 58 mil gerações podem ser visto na tabela 32. Os resultados nesta tabela mostram que, mesmo aumentando o número de gerações (média de tempo) do μAG os resultados obtidos pelo MSFLPA ainda são melhores e/ou estaticamente equivalentes em todos os objetivos.

Devido a limitações físicas, o valor mínimo de 0 não pode ser alcançado para a função objetivo $f_3(x)$ relacionado à programação do horizonte. É impossível receber todos os produtos em cada área, ao mesmo tempo. No que diz respeito a fragmentação das bateladas, função objetivo $f_4(x)$, os resultados do MSFLPA e NSGA-II estão mais perto da solução ótima dos que os resultados do μAG. De fato, baseados nos valores apresentados, os resultados do algoritmo MSFLPA são melhores em praticamente quase todos os objetivos do que os resultados dos algoritmos μAG e NSGA-II. Esse melhor desempenho pode ser atribuído ao esquema de busca local implementada pelo algoritmo SFLA, e pelas novas estratégias do MSFLPA.

O algoritmo NSGA-II obteve os piores resultados em termos de solução dos objetivos, isto pode ser visto na Tabela 33. Além disso, a convergência de NSGA-II para o problema apresentado foi lenta quando comparada com os outros dois algoritmos. Como mencionado anteriormente, o algoritmo NSGA-II apresenta algumas dificuldades para problemas de grande porte com mais do que dois objetivos, como é o caso do problema de escalonamento em redes de dutos.

O desempenho dos modelos para cada objetivo, considerando o melhor indivíduo dos experimentos, é mostrado na Tabela 36. A Figura 35 mostra para todos os modelos, os gráficos de Gantt da melhor solução (individual) ao longo do horizonte de programação (48 intervalos de tempo) para uma simulação, onde P1 a P8 representam as oito diferentes produtos transportados pela rede. Em todos os gráficos, a conexão 8 que transporta produtos para o porto, é a mais utilizada em todo o horizonte de programação. A partir desta figura, pode-se ver que

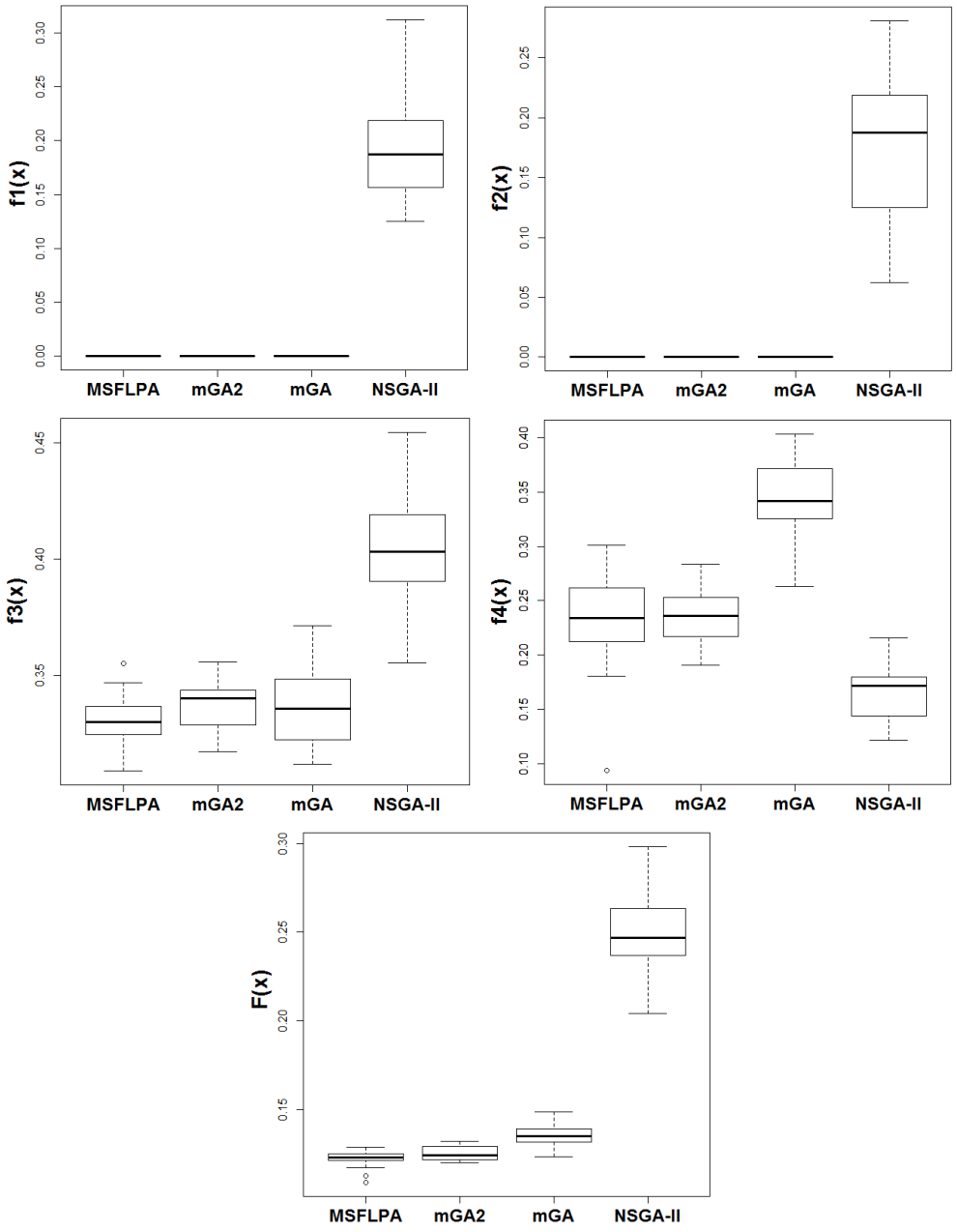


Figura 34: Boxplots das funções objetivos para 20 simulações

as bateladas são mais fragmentadas (função objetivo $f_4(x)$) no Gantt do μAG, especialmente para as conexões 1 e 2 e no Gantt do NSGA-II, especialmente para as conexões 6 e 8.

Tabela 36: *Fitness* para todos os objetivos

Objetivo	$F(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
MSFLPA	0.1086	0	0	0.330	0.093
μAG	0.1317	0	0	0.337	0.306
NSGAII	0.2042	0.1875	0.0625	0.390	0.121

Para garantir que os resultados obtidos pelo MSFLPA são satisfatórios para todos os objetivos, foram comparados os valores dos quatros objetivos e função de avaliação ($F(x)$) usando o teste t de *Student* (WALPOLE et al., 1998). Os resultados estatísticos obtidos por um teste t de *Student* bicaudal, com 38 graus de liberdade a um nível de significância de 0.05 é mostrado na Tabela 37. O resultado do Algoritmo-1 $\leftrightarrow$ Algoritmo-2 é mostrado como $+$ ou $\sim$ quando Algoritmo-1 é significativamente melhor do que ou estatisticamente equivalente ao Algoritmo-2, respectivamente.

Como esperado, os resultados na Tabela 37 confirmam que o MSFLPA obteve um melhor desempenho do que o NSGA-II para todas as funções objetivos e é igual a (função objetivo $f_1(x)$ e $f_2(x)$) e significativamente melhor do que ou estatisticamente equivalente a μAG (função objetivo $f_3(x)$ e $f_4(x)$).

Tabela 37: Resultados do teste t de Student para todos objetivos

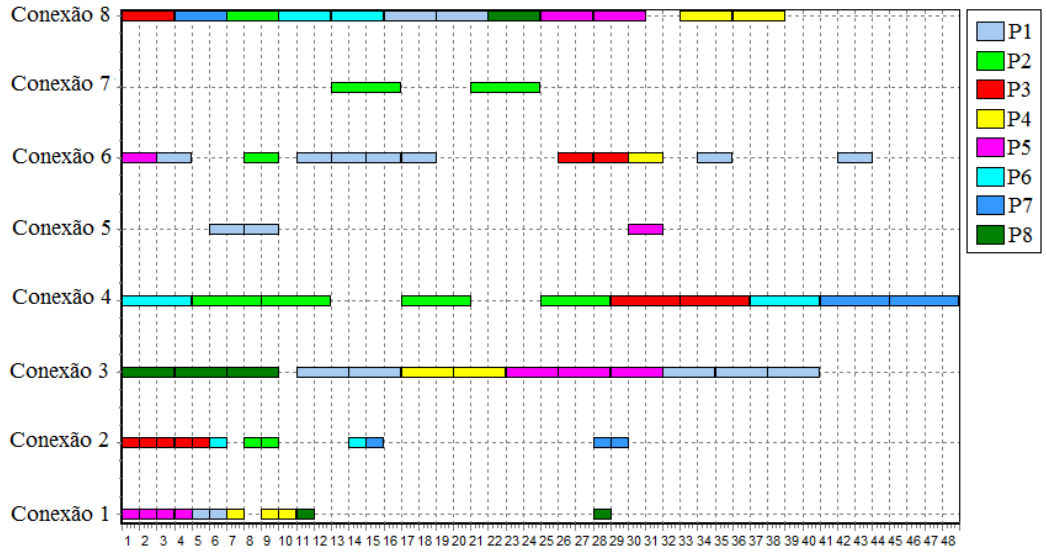
Algoritmo-1 $\leftrightarrow$ Algoritmo-2	$F(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
MSFLPA $\leftrightarrow$ NSGA-II	$+$	$+$	$+$	$+$	$\sim$
MSFLPA $\leftrightarrow$ μAG	$+$	$\sim$	$\sim$	$\sim$	$+$
MSFLPA $\leftrightarrow$ μAG2	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$

7.1.1 TEMPO E SOLUÇÕES DE PARETO

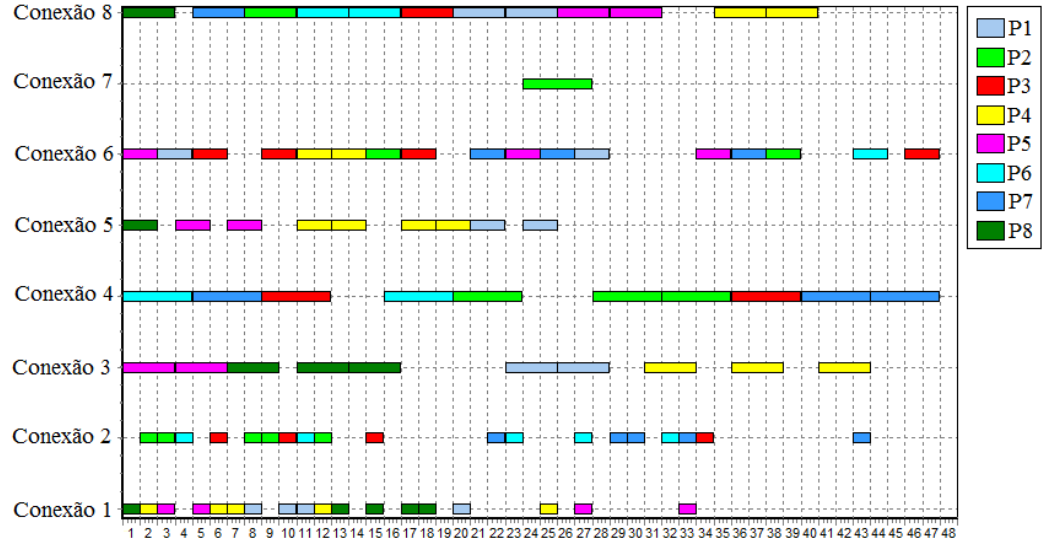
Tabela 38: Resultados para 20 simulações

Modelo	Média do tempo computacional
μAG	2.80s
NSGA-II	5.54s
MSFLPA	65.17s

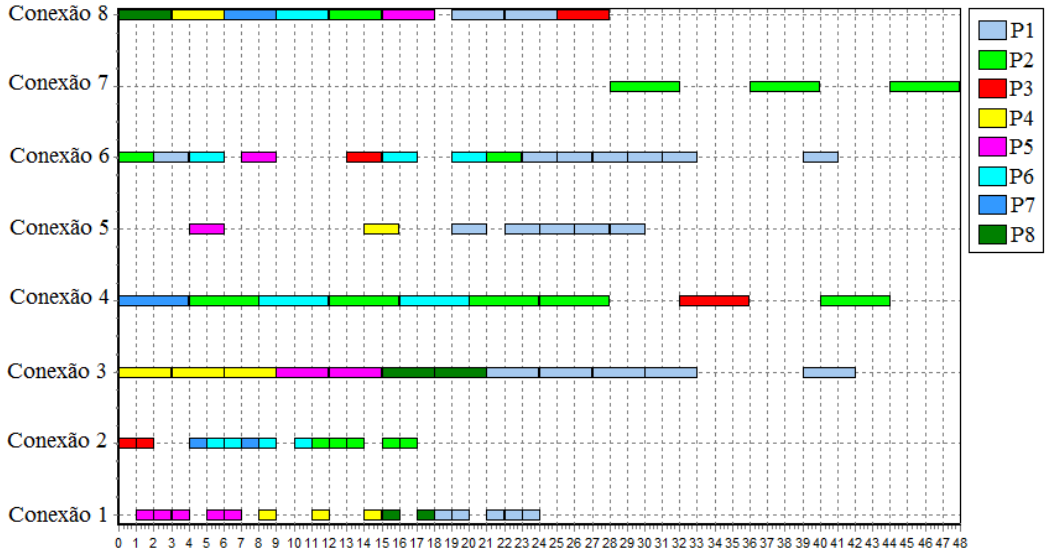
O tempo médio de execução para 20 simulações para os modelos é mostrada na Tabela 38. O tempo médio do modelo MSFLPA é pior, isso deve-se ao fato das novas estratégias incluídas para a busca de soluções de Pareto e principalmente pelo procedimento de busca



(a) MSFLPA



(b) μAG



(c) NSGAII

Figura 35: Gráficos Gantt da melhor solução

local. No entanto, como discutido acima, os resultados do MSFLPA são melhores para todos os objetivos.

Todas as soluções obtidas em cada simulação para cada modelo são armazenadas em um conjunto de soluções viáveis. Estes conjuntos de soluções finais viáveis calculados para 20 simulações de todos os modelos são mostrados na Figura 36. O modelo μAG encontrou 34 soluções diferentes, o μAG2 encontrou 27 soluções diferentes, o NSGA-II encontrou 68 soluções diferentes e o modelo MSFLPA encontrou 87 soluções diferentes.

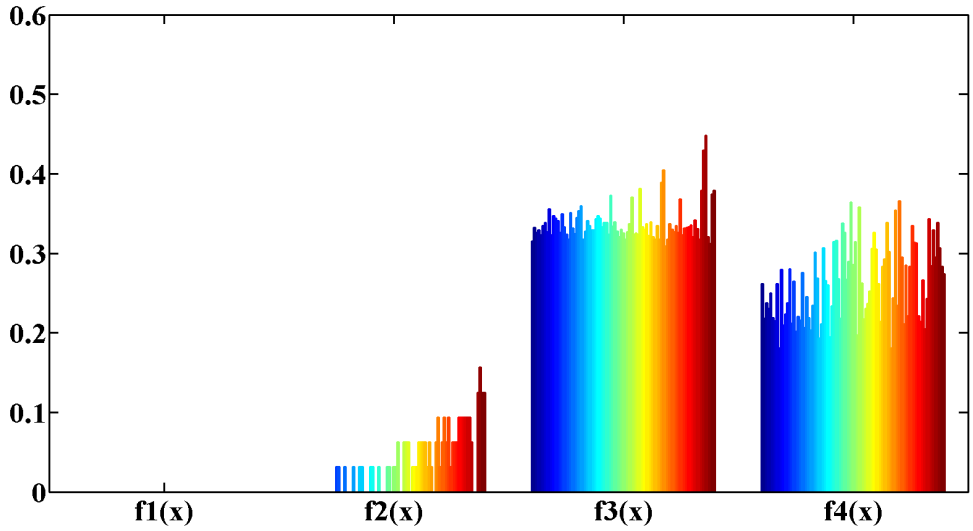
Embora a cada simulação todos os algoritmos calculem as soluções não-dominadas, o conjunto final resultante das 20 simulações podem conter soluções dominadas. Assim, uma análise entre as soluções utilizando o conceito de não-dominância é considerada a fim de formar um conjunto ótimo de Pareto e permitir a construção da fronteira de Pareto para o problema. Para o modelo μAG, apenas 2 soluções não-dominadas são encontrados entre as 34 soluções do conjunto. Isto é, o modelo μAG explorou a mesma região do espaço de busca do problema, e ele sempre alcança soluções semelhantes ao longo de 20 simulações. Para o μAG2, 7 soluções não-dominadas são encontrados entre as 27 soluções do conjunto. Para o modelo NSGA-II, 8 soluções não-dominadas são encontrados entre as 68 soluções viáveis e o modelo MSFLPA, 9 soluções não-dominadas são encontrados entre as 87 soluções viáveis.

Na Figura 36, se compararmos os conjuntos de soluções não dominadas encontrados para todos os algoritmos, pode-se constatar que as melhores soluções em termos de minimização dos objetivos são as soluções não-dominadas encontradas pelo MSFLPA.

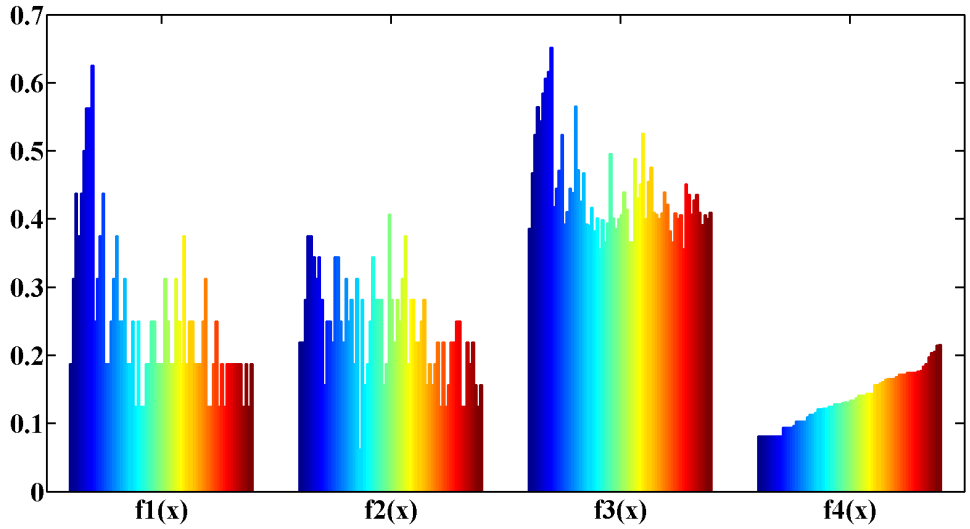
A fronteira de Pareto é composta por todos os pontos do conjunto Pareto-ótimo. Para construir essa fronteira, as duas soluções não-dominadas encontradas pelo μAG, as sete soluções não-dominadas encontradas pelo μAG2 e os nove soluções não-dominadas encontradas pelo MSFLPA são consideradas. O conjunto de soluções encontrado pelo NSGA-II não alcançou valores ótimos em qualquer dos objetivos, por isso não serão consideradas.

O três modelos (μAG, μAG2 e MSFLPA) atingiram os valores ótimos (0) para as demandas de recebimento e fornecimento ($f_1(x)$ e $f_2(x)$) para soluções não-dominadas encontradas, portanto a fronteira será construída apenas considerando os objetivos $f_3(x)$ e $f_4(x)$ na Figura 37. Ao analisar esta figura, pode-se notar que as soluções de modelo MSFLPA dominam todas as soluções do μAG e μAG2.

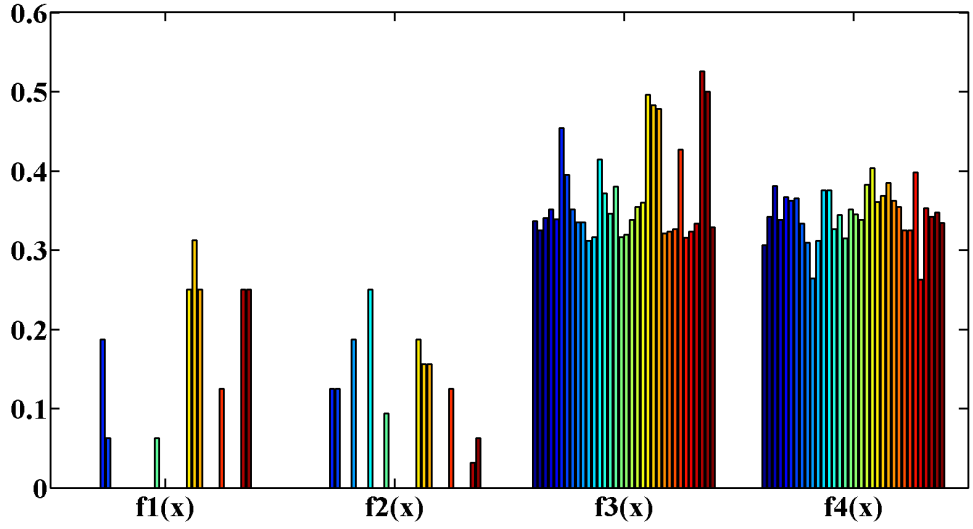
Além disso, as soluções do MSFLPA estão melhores “espalhadas”sobre o espaço de busca, resultando em um traçado da fronteira mais próximo da fronteira real e desconhecida de Pareto. Assim, pode-se concluir que a fronteira de Pareto calculada pelo MSFLPA está mais



(a) MSFLPA



(b) NSGA-II



(c) μAG

Figura 36: Soluções não-dominadas para 20 simulações.

perto de verdadeira fronteira de Pareto que as fronteiras calculadas pelos outros dois modelos.

A partir deste estudo comparativo, pode-se concluir que todos os modelos evolucionários são capazes de resolver o problema de escalonamento de rede de dutos. A carga computacional para μAG e NSGA-II é menor, no entanto, o conjunto de soluções calculado pela MSFLPA é muito melhor para todos os objetivos, especialmente para a minimização da fragmentação das bateladas($f_4(x)$), quando comparado com o modelo μAG e para as demandas de fornecimento e recebimento dos produtos ($f_1(x)$, $f_2(x)$) e a minimização do tempo de entrega ($f_3(x)$) quando comparado com o modelo NSGA-II. Além disso, a fronteira construída pelo modelo MSFLPA está mais perto da linha real da fronteira de Pareto do que o μAG e μAG2.

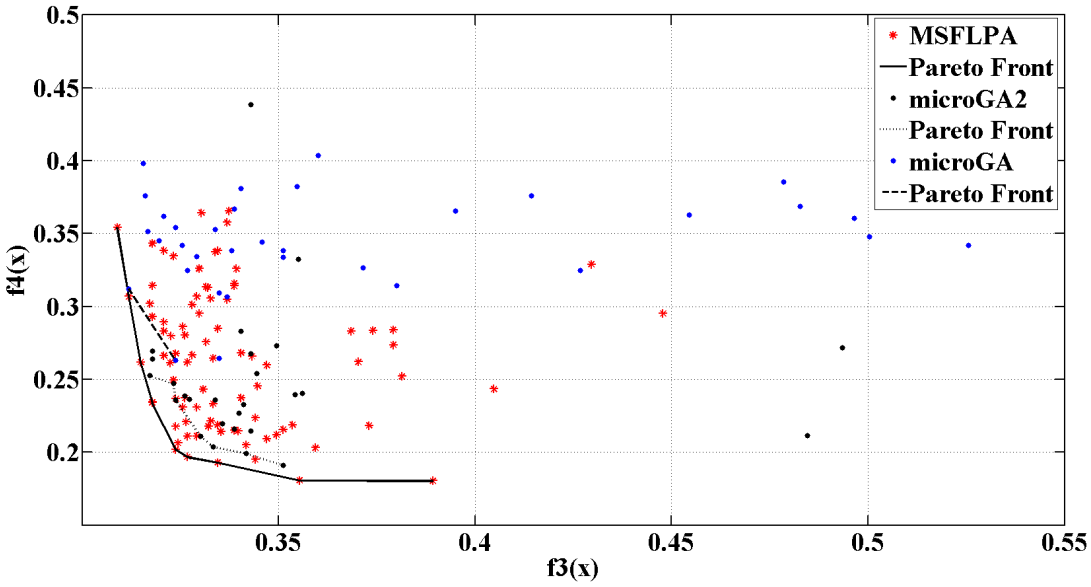


Figura 37: Fronteira de Pareto para somente dois objetivos

De uma forma geral, a abordagem proposta *Modified Shuffled Frog-Leaping Pareto Approach* apresentou um bom desempenho para calcular o conjunto Pareto-ótimo para o problema de escalonamento de rede de dutos. O tempo de processamento pode ser considerado razoável, cerca de 1 minuto para o cenário testado, embora possa crescer com o tamanho dos cenários (volume de produtos a serem transportados) e/ou tamanho da rede. Para o cenário apresentado aqui, o MSFLPA alcançou a minimização para todos os objetivos propostos no problema modelado. Assim, o processo de escalonamento de curto prazo calculadas pelo MSFLPA é razoável e viável para o estudo da rede de dutos deste trabalho.

Com base nos resultados deste capítulo, pode-se destacar algumas vantagens (*Vant*) e desvantagens (*Desv*) de MSFLPA:

Vant: o MSFLPA apresenta bons resultados (similar ou melhor do que NSGA-II e μAG) para

problemas com dois objetivos, caracterizando uma rápida convergência para um conjunto de solução de Pareto viável;

Vant: semelhante ao μAG, o procedimento de reinicialização da população assegura a diversidade da população e pode prevenir uma convergência prematura para super-indivíduos com população pequena;

Vant: como MSFLPA trabalha com a população dividida em grupos de indivíduos, um conjunto de soluções de Pareto pode ser obtido em apenas uma única simulação que pode abranger todas as restrições do problema e lidar com eles simultaneamente durante a evolução dos grupos dos indivíduos ao longo das gerações. Assim, MSFLPA é recomendado para resolver problemas complexos de escalonamento com restrições.

Desv: O tempo computacional de MSFLPA é afetado pela busca local que calcula *N* evoluções para cada grupo de indivíduos da população. Este tempo computacional também aumenta com a dimensão do problema e população maiores, no entanto, o número de evoluções tem o maior impacto no tempo e desempenho do algoritmo. De acordo com (EUSUFF et al., 2006), um *N* variando de 10 a 20 evoluções é uma boa escolha para várias aplicações.

8 RESULTADOS OBTIDOS PARA O MSFLPA APLICADO A DOIS CENÁRIOS REAIS

Neste capítulo são apresentados simulações com dois cenários reais da rede de escuros adaptados ao modelo de otimização multiobjetivo MSFLPA desenvolvido neste trabalho. O principal objetivo dessas simulações é validar o uso do algoritmo proposto em cenários reais e comprovar se os resultados obtidos estão próximos do esperado para um problema real.

8.1 OTIMIZAÇÃO MULTIOBJETIVO MSFLPA PARA DOIS CENÁRIO REAIS

Como já foi citado nos resultados obtidos dos capítulos anteriores, todos os parâmetros de modelos evolucionários foram escolhidos após vários experimentos utilizando as diretrizes do (GREFENSTETTE, 1986). Para as simulações apresentadas a seguir, os parâmetros do MSFLPA estão na tabela 39. Todas as simulações foram executados em um processador Intel Core 2 Quad, 2,83 GHz, 4 GB de RAM, com 64 bits plataforma Windows. O modelo é codificado em linguagem de programação C++.

Tabela 39: Parâmetros do MSFLPA

Parâmetros	MSFLPA
Tamanho da População	20
Número de gerações	2000
Memeplexes	4
Iterações evolucionária	10
Fator de Aceleração	2

Enfatizando que o modelo multiobjetivo evolucionário procura satisfazer simultaneamente os quatros objetivos dadas pelas equações 36 a 41 do capítulo 5, e correspondem as seguintes funções objetivos:

$f_1(x)$ Receber a quantidade de bateladas demandada pelos terminais;

$f_2(x)$ Satisfazer a produção mínima;

$f_3(x)$ Minimizar o tempo de entrega das bateladas;

$f_4(x)$ Minimizar a fragmentação das bateladas.

A otimização multiobjetivo leva em consideração apenas o valor global de cada objetivo, calculado pela equação 43. Os pesos associados a cada objetivo nesta equação são dadas na tabela 40.

Para as simulações, os objetivos $f_3(x)$ e $f_4(x)$ receberam um peso menor. No caso do objetivo $f_3(x)$, esse peso foi reduzido pois em um cenário real o tempo total de planejamento geralmente é cumprido, ou seja a minimização do tempo não é considerada prioridade em cenários reais. Já o objetivo $f_4(x)$ pode prejudicar o desempenho de otimização dos outros objetivos. Contudo, evitar a fragmentação das bateladas é um critério muito importante, mas também não é prioridade nas designações das atividades operacionais da rede de dutos conforme os especialistas.

Tabela 40: Pesos utilizados na função objetivo

Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Peso	3	3	1	1

Como o modelo desenvolvido tem representação discreta, algumas adaptações são necessárias para chegar mais próximo da representação dos cenários reais. Os cenários 1 e 2 (como serão referenciados no texto) utilizados, seguem o planejamento de alocação e sequenciamento das bateladas para um mês de 31 dias. Dessa forma, considerando uma unidade de tempo discreta do modelo como sendo 1 hora no planejamento real, o horizonte de programação das simulações será de 744 unidades de tempo. Assim, o último recebimento de produto possível será na unidade de tempo 744 e nenhum envio poderá ocorrer após esse tempo.

Tabela 41: Matriz de tempo das conexões na rede

Conexão	Nó origem	Nó destino	Tempo
1	1	5	8
2	2	6	6
3	3	6	9
4	4	7	3
5	5	6	8
6	6	7	6
7	7	4	3
8	7	8	6

Para executar as simulações, o volume máximo e mínimo de cada tanque em cada área deve ser definido, e também a duração de cada conexão deve ser configurada em termos de

intervalos de tempo. A representação desses intervalos das conexões adaptadas aos cenários reais está indicada na Tabela 41. Esses intervalos de tempo são quantas unidades de tempo uma batelada demora para percorrer a conexão do Nó origem ao Nó destino.

Os produtos produzidos pela refinarias podem ser configurados conforme exigido pelo cenário em questão, as designações dos produtos produzidos por cada refinaria para os dois cenários estão apresentados na Tabela 42.

Tabela 42: Produtos produzidos por cada refinaria

Refinaria/Nó	Cenário 1	Cenário 2
N1	1,2,3,4,5,6	2,3,4,6,7,8
N2	2,3,4,5,6,8	1,2,3,4,6
N3	3,4,5,6,7	2,4,5,7
N4	1,2,4,5,8	2,3,4,5,6,7,8

As demandas de fornecimento são configuradas através de uma quantidade mínima de bateladas de cada produto que deve ser enviada a partir de cada refinaria. Assim como as demandas de recebimento são configuradas através da quantidade de bateladas de cada produto que um terminal deverá receber. Para a rede utilizada no modelo, o nó N8 é único terminal, e o objetivo geral da rede é atender a demanda desse nó. As Tabelas 43 e 44 apresentam as configurações das demandas de ambos cenários.

Tabela 43: Produtos produzidos por cada refinaria

Cenário 1	P1	P2	P3	P4	P5	P6	P7	P8
N1	6	6	10	1	15	12	-	-
N2	-	8	7	11	10	15	-	3
N3	-	-	-	2	-	-	2	-
N4	3	2	-	6	7	-	-	-
Cenário 2	P1	P2	P3	P4	P5	P6	P7	P8
N1	-	9	6	4	-	4	-	4
N2	10	6	7	6	-	3	-	-
N3	-	2	-	2	3	-	1	-
N4	-	3	6	1	3	-	5	-

Tabela 44: Demanda no nó terminal (porto)

Cenário 1	P1	P2	P3	P4	P5	P6	P7	P8
N8	32	14	-	4	-	-	-	-
Cenário 2	P1	P2	P3	P4	P5	P6	P7	P8
N8	15	27	-	-	3	-	-	-

Lembrando que as funções objetivos são normalizadas no intervalo $[0;1]$, o ótimo global da função de avaliação na equação 43 é 0 que corresponde ao cumprimento de todos

os critérios. Se o valor final da função de avaliação for superior a zero, um ou mais objetivos são cumpridos parcialmente.

Os dois cenários são simulados com 20 populações iniciais diferentes (20 execuções) sobre 2000 gerações. Os resultados estatísticos em termos de média, mediana, desvio padrão, valores máximo e mínimo para cada função objetivo (equações 36 a 41) e para a função de avaliação (equação 43) são mostrados nas Tabelas 45, 46 e 47. Estes resultados também são ilustrados nos boxplots de Figura 38.

Tabela 45: Resultados estatísticos para 20 simulações para o cenário 1

Função Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Mínimo	0	0	0.5179	0.6471
Máximo	0.3571	0.2319	0.8982	0.7496
Média	0.0363	0.0167	0.6684	0.7231
Mediana	0.0208	0.0083	0.6639	0.7244
Desvio Padrão	0.0512	0.0295	0.0635	0.0126

Tabela 46: Resultados estatísticos para 20 simulações para o cenário 2

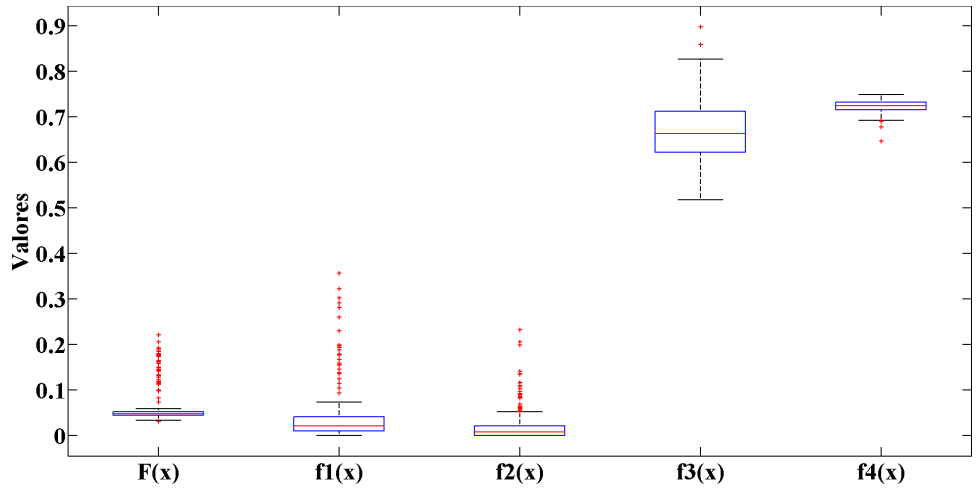
Função Objetivo	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Mínimo	0	0	0.4053	0.5531
Máximo	0.5259	0.1717	0.9490	0.7836
Média	0.0412	0.0248	0.6238	0.7133
Mediana	0.0222	0.0100	0.6335	0.7217
Desvio Padrão	0.0713	0.0342	0.0860	0.0367

Tabela 47: Resultados estatísticos para 20 simulações

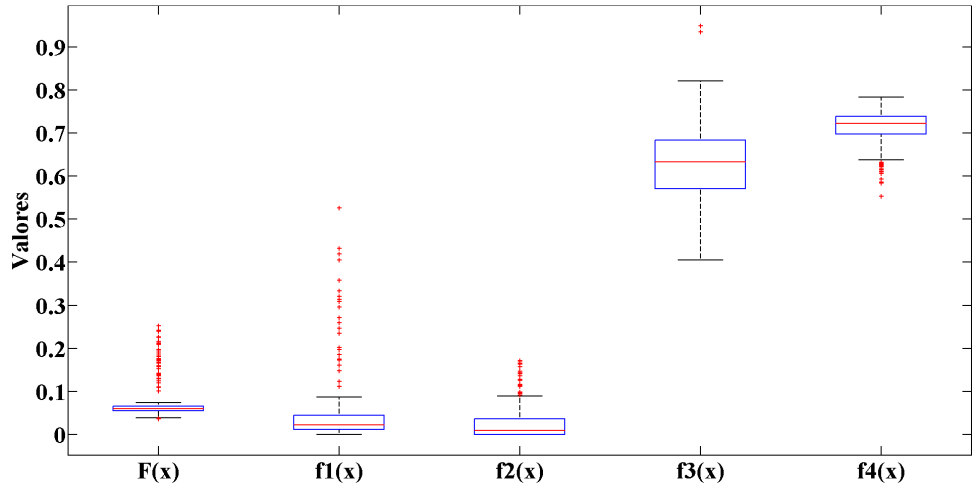
	Cenário 1 $F(x)$	Cenário 2 $F(x)$
Mínimo	0.0308	0.0362
Máximo	0.2217	0.2526
Média	0.0570	0.0675
Mediana	0.0484	0.0604
Desvio Padrão	0.0324	0.0341

Os resultados apresentados nas tabelas mostram que o MSFLPA encontrou o ponto ótimo para os dois primeiros objetivos $f_1(x)$ e $f_2(x)$, ou seja, as demandas de recebimento e de fornecimento foram totalmente cumpridas para ambos cenários. Atender as demandas é prioridade e são considerados objetivos muito importantes em um cenário real.

O desempenho de ambos cenários para cada objetivo, considerando o melhor indivíduo dos experimentos, é mostrada na Tabela 48. A Figura 39 mostra para os dois cenários, os gráficos de Gantt da melhor solução (individual) ao longo do horizonte de programação (48



(a) Cenário 1



(b) Cenário 2

Figura 38: Resultados estatísticos de 20 simulações

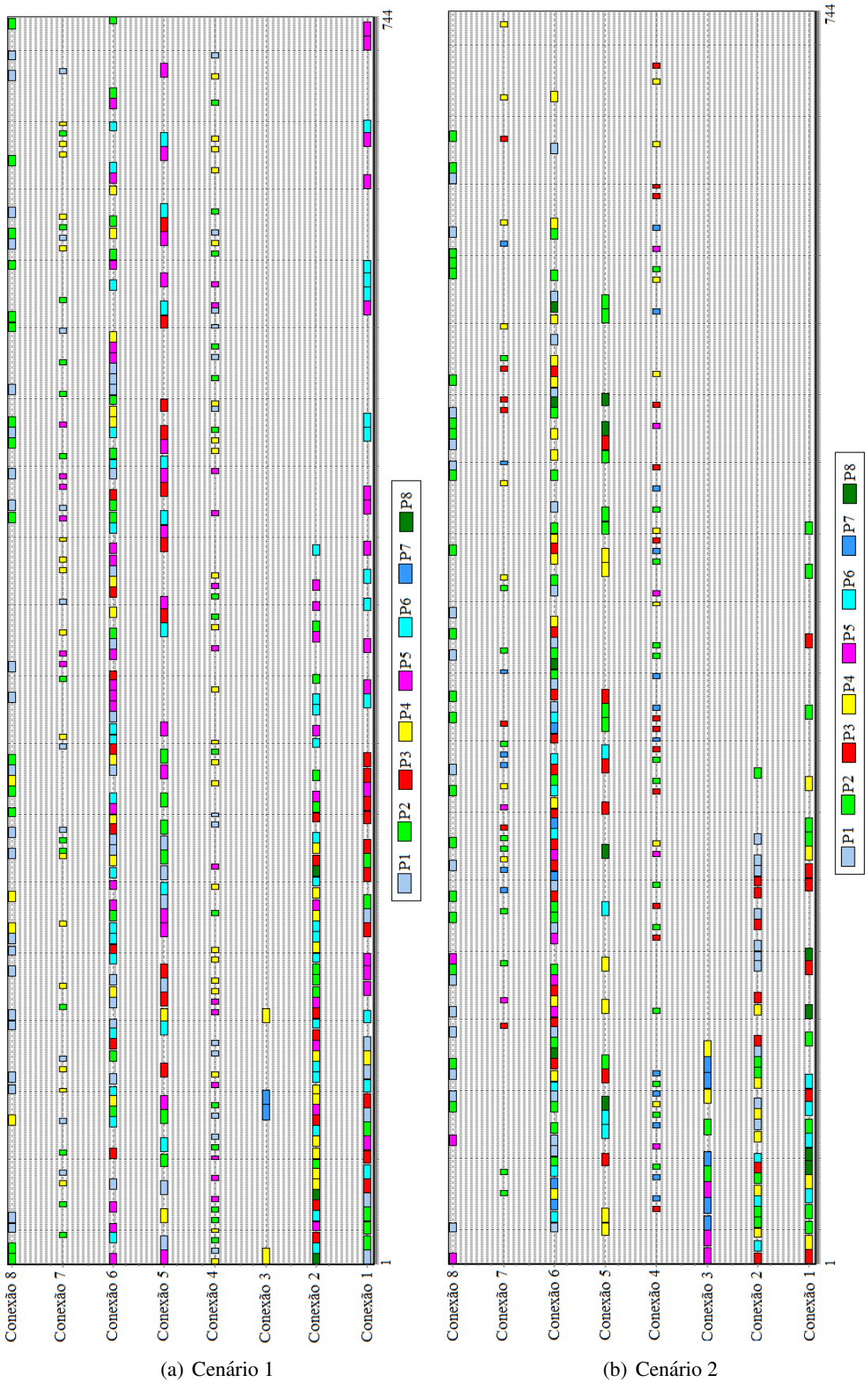
intervalos de tempo) para uma simulação, onde P1 a P8 representam as oito diferentes produtos transportados pela rede.

8.1.1 TEMPO E SOLUÇÕES DE PARETO

A Tabela 49 apresenta os resultados obtidos do algoritmo MSFLPA para as 20 execuções, de 2000 gerações cada, de ambos os cenários. O MSFLPA encontrou uma média de aproximadamente 21 soluções Pareto-ótimas para o cenário 1 e uma média de 24 soluções Pareto-ótimas para o cenário 2.

O tempo de execução médio apresentado na Tabela 49 mostra que o MSFLPA obteve bons resultados em um tempo consideravelmente razoável para ambos os cenários.

Todas as soluções obtidas em cada simulação para cada cenário são armazenadas em



(a) Cenário 1

(b) Cenário 2

Figura 39: Distribuição dos produtos ao longo do tempo

Tabela 48: *Fitness* para todos os objetivos

Objetivo	$F(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Cenário 1	0.0308	0	0	0.619	0.7342
Cenário 2	0.0362	0	0	0.511	0.7546

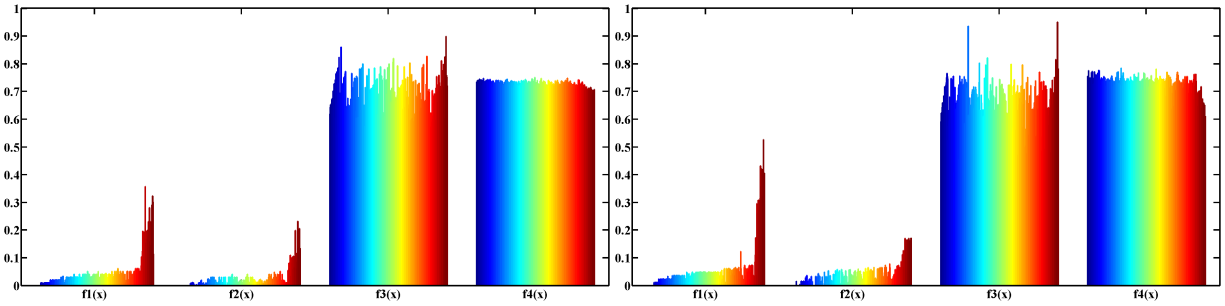
Tabela 49: Conjunto de soluções de pareto para cada simulação

Simulações	Cenário 1	Cenário 2
1	24	17
2	16	24
3	17	16
4	24	20
5	26	24
6	27	27
7	22	22
8	19	23
9	14	31
10	31	16
11	19	29
12	15	22
13	15	23
14	20	32
15	23	29
16	11	32
17	21	27
18	22	16
19	19	26
20	28	21
Total de soluções	413	477
Média de soluções	20.65	23.85
Média de tempo de execução	470.34s	444.07s

um conjunto de soluções viáveis. Estes conjuntos de soluções finais viáveis calculados para 20 simulações de ambos cenários são mostrados na Figura 40. O MSFLPA encontrou no total das simulações 413 soluções para o cenário 1 e 477 soluções para o cenário 2.

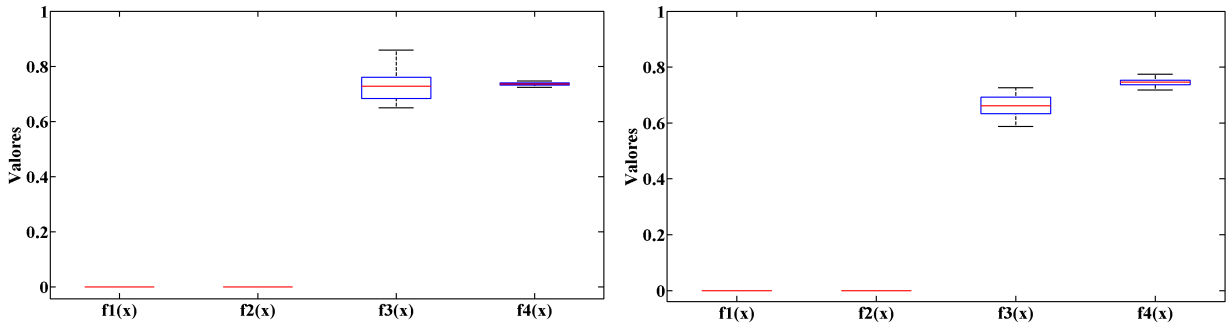
Embora a cada simulação o algoritmo calcule as soluções não-dominadas, o conjunto final resultante das 20 simulações podem conter soluções dominadas. Assim, uma análise entre as soluções utilizando o conceito de não-dominância é considerada a fim de formar um conjunto ótimo de Pareto e permitir a construção da fronteira de Pareto para o problema.

A fronteira de Pareto é composta por todos os pontos do conjunto Pareto-ótimo. Como o conjunto total de soluções encontrado pelo algoritmo para ambos cenários foi grande, e ressaltando que trata-se de quatro objetivos e representar esses objetivos no espaço dimensional



(a) Cenário 1 (b) Cenário 2

Figura 40: Conjunto de Soluções Viáveis para 20 simulações.



(a) Cenário 1 (b) Cenário 2

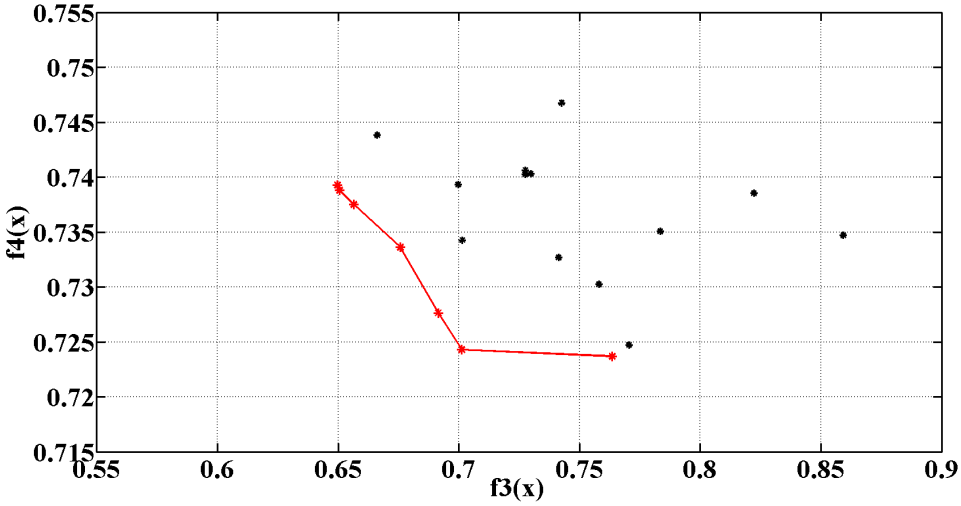
Figura 41: Conjunto de Pareto das 20 melhores soluções.

para visualizar a fronteira fica inviável, esse conjunto será divido em três classes de soluções para fazer uma análise de dominância em cada classe.

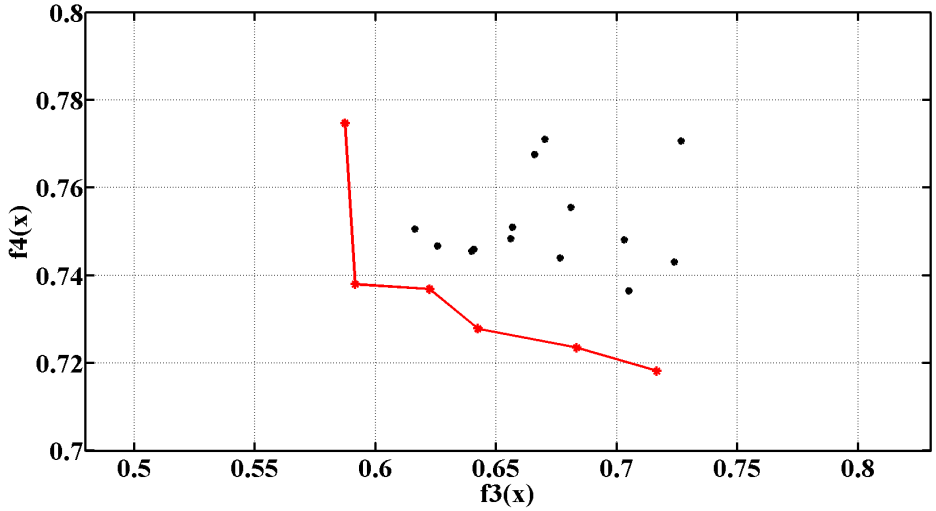
A primeira classe será das 20 melhores soluções, ou seja, as soluções que atingiram valores ótimos (0) para as demandas de recebimento e fornecimento ($f_1(x)$ e $f_2(x)$). A segunda classe será composta pelas soluções que atingiram valores ótimos (0) para a demanda de recebimento ($f_1(x)$) e a terceira classe será composta pelas soluções que atingiram valores ótimos (0) para a demanda de fornecimento ($f_2(x)$).

Inicialmente, para construir a fronteira utiliza-se a primeira classe de soluções (20 melhores soluções dentro do conjunto de soluções viáveis para cada cenário), essas soluções estão representadas na Figura 41. Como mencionado, esse conjunto de soluções atingiu os valores ótimos (0) para as demandas de recebimento e fornecimento ($f_1(x)$ e $f_2(x)$), portanto a fronteira será construída apenas considerando os objetivos $f_3(x)$ e $f_4(x)$ na Figura 42.

Apesar da primeira classe ser composta pelas melhores soluções, elas foram geradas por simulações diferentes e o conjunto final resultante pode conter soluções dominadas. Assim, uma análise entre as soluções utilizando o conceito de não-dominância foi realizada. Na Figura 42 pode-se observar que para o cenário 1, das 20 melhores soluções, foram encontradas 7 soluções não-dominadas e para o cenário 2, das 20 melhores soluções, foram encontradas 6



(a) Cenário 1



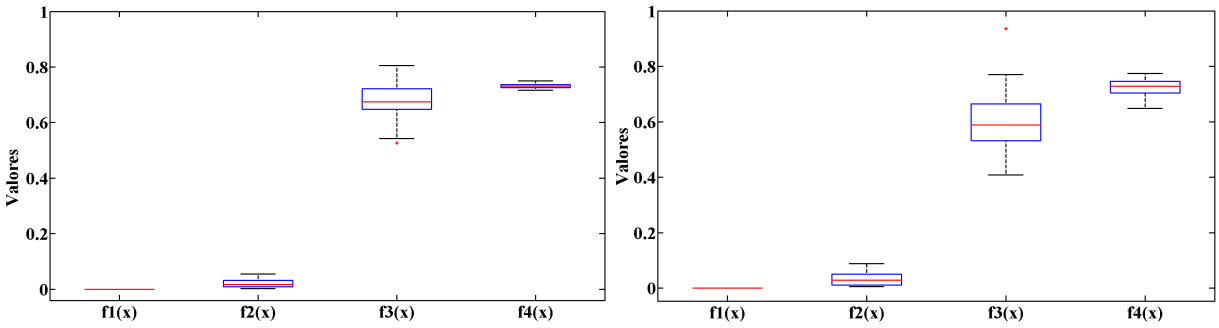
(b) Cenário 2

Figura 42: Fronteira de Pareto das 20 melhores soluções.

soluções não-dominadas. Dessa forma, pode-se observar que o algoritmo MSFLPA além de obter boas soluções, isto é atender todas as demandas que é prioridade em um cenário real, também obteve um conjunto representativo de soluções Pareto-ótima na solução do problema.

Para a segunda classe de soluções, considerou-se todas as soluções que obtiveram o valor ótimo (0) para a demanda de recebimento ($f_1(x)$). Para o cenário 1, do conjunto total de 413 soluções, 56 soluções conseguiram atingir o valor ótimo para objetivo $f_1(x)$. Para o cenário 2, do conjunto total de 477 soluções, 87 soluções conseguiram atingir o valor ótimo para objetivo $f_1(x)$. Essas soluções estão representadas nos boxplots das Figura 43.

A segunda classe também foi gerada por simulações diferentes e o conjunto final resultante pode conter soluções dominadas. Assim, uma análise entre as soluções utilizando



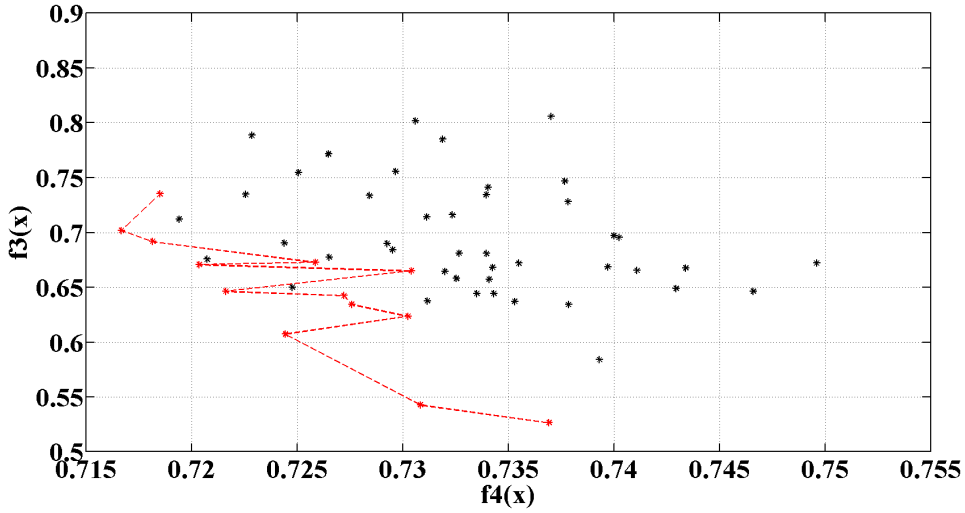
(a) Cenário 1 (b) Cenário 2

Figura 43: Conjunto de Soluções para $f1(x) = 0$

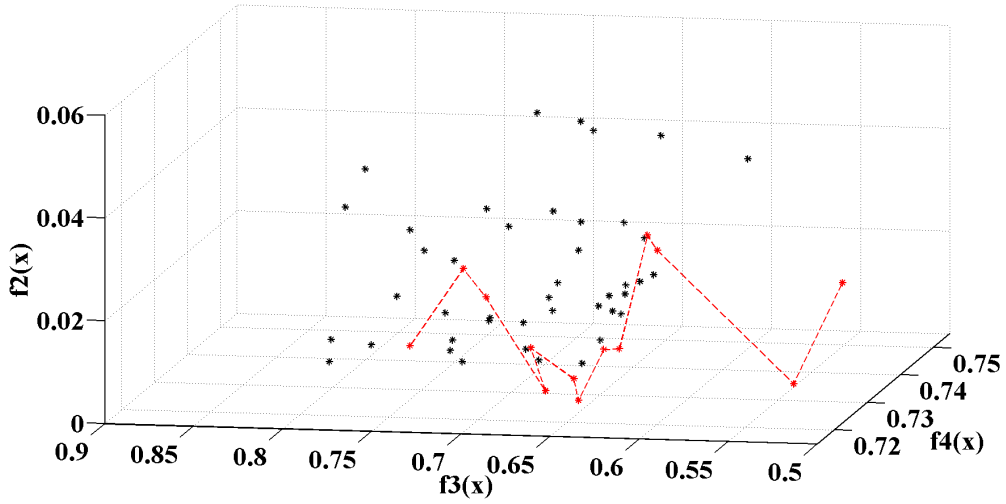
o conceito de não-dominância foi realizada. Para o cenário 1, das 56 soluções com $f1(x) = 0$, foram encontradas 13 soluções não-dominadas e para o cenário 2, das 87 soluções com $f1(x) = 0$, foram encontradas 16 soluções não-dominadas, essas soluções podem ser vistas nas figuras 44 e 45, para cenário 1 e 2, respectivamente. Para essas figuras, foram considerados 3 objetivos para representar a fronteira, tornando-se assim uma representação em três dimensões, como é desconhecido o formato da fronteira, fica difícil a projeção em 3D. Dessa forma, nas Figuras 44(a) e 45(a) são os pontos em duas dimensões para tentar visualizar a fronteira e nas Figuras 44(b) e 45(b) uma visualização aproximada da fronteira em três dimensões.

Para a terceira classe de soluções, considerou-se todas as soluções que obtiveram o valor ótimo (0) para a demanda de fornecimento ($f_2(x)$). Para o cenário 1, do conjunto total de 413 soluções, 130 soluções conseguiram atingir o valor ótimo para objetivo $f_2(x)$. Para o cenário 2, do conjunto total de 477 soluções, 143 soluções conseguiram atingir o valor ótimo para objetivo $f_2(x)$. Essas soluções estão representadas nos boxplots da Figura 46.

A terceira classe também foi gerada por simulações diferentes e o conjunto final resultante pode conter soluções dominadas. Assim, uma análise entre as soluções utilizando o conceito de não-dominância foi realizada. Para o cenário 1, das 130 soluções com $f2(x) = 0$, foram encontradas 17 soluções não-dominadas e para o cenário 2, das 143 soluções com $f2(x) = 0$, foram encontradas 29 soluções não-dominadas, essas soluções podem ser vistas nas figuras 47 e 48, para cenário 1 e 2, respectivamente. Para essas figuras, foram considerados 3 objetivos para representar a fronteira, tornando-se assim uma representação em três dimensões, como é desconhecido o formato da fronteira, fica difícil a projeção em 3D. Dessa forma, nas figuras 47(a) e 48(a) são os pontos em duas dimensões para tentar visualizar a fronteira e nas figuras 47(b) e 48(b) uma visualização aproximada da fronteira em três dimensões.

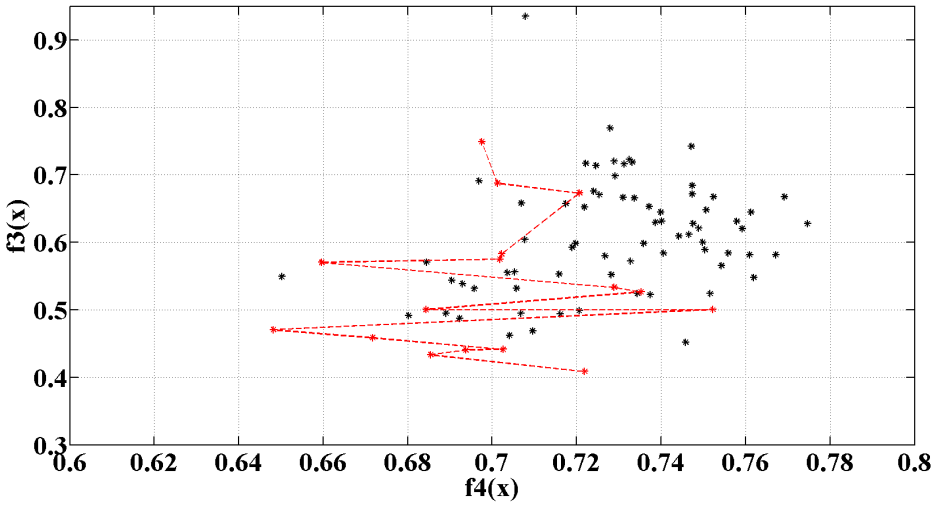


(a) $f1(x) = 0$

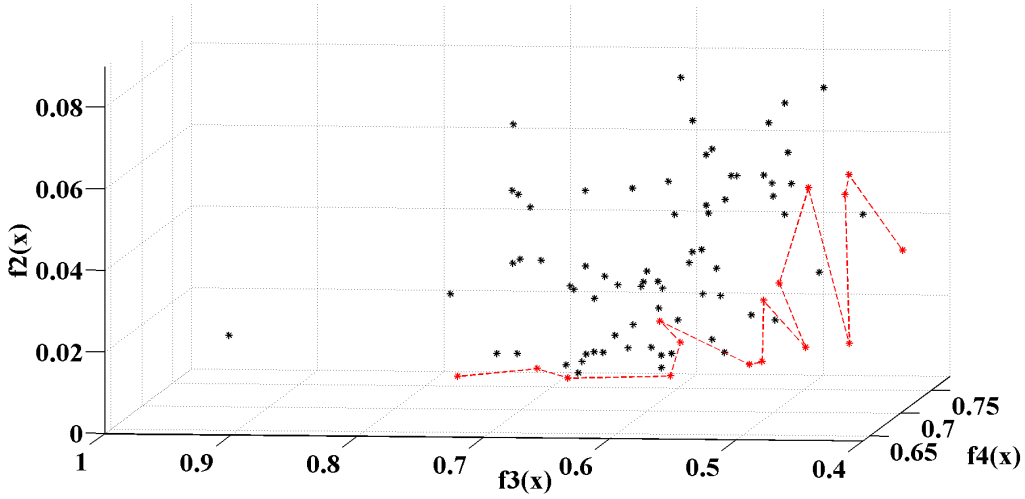


(b) $f1(x) = 0$

Figura 44: Fronteira de Pareto para três objetivos para o cenário 1.

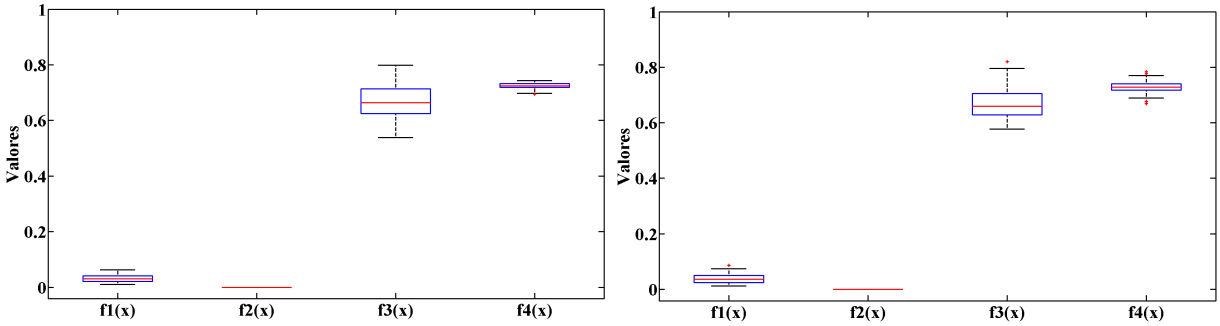


(a) $f1(x) = 0$



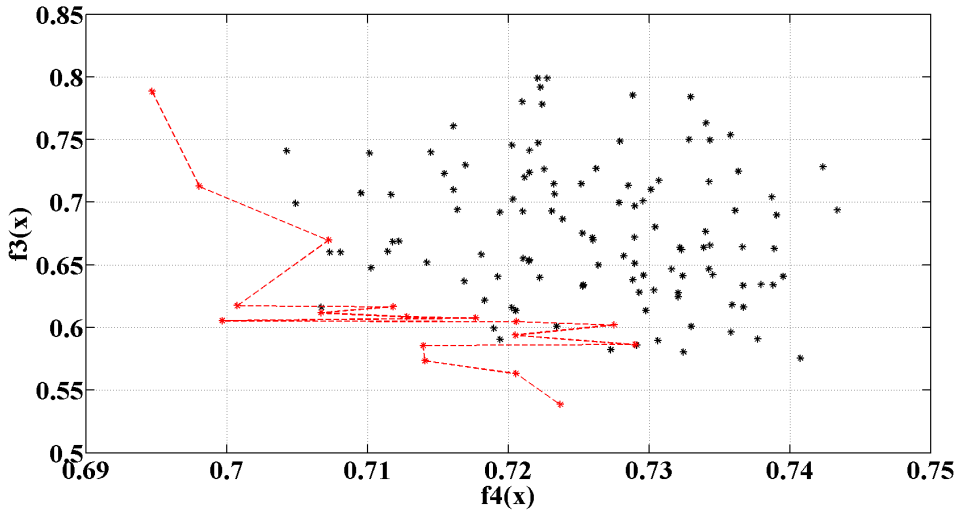
(b) $f1(x) = 0$

Figura 45: Fronteira de Pareto para três objetivos para o cenário 2.

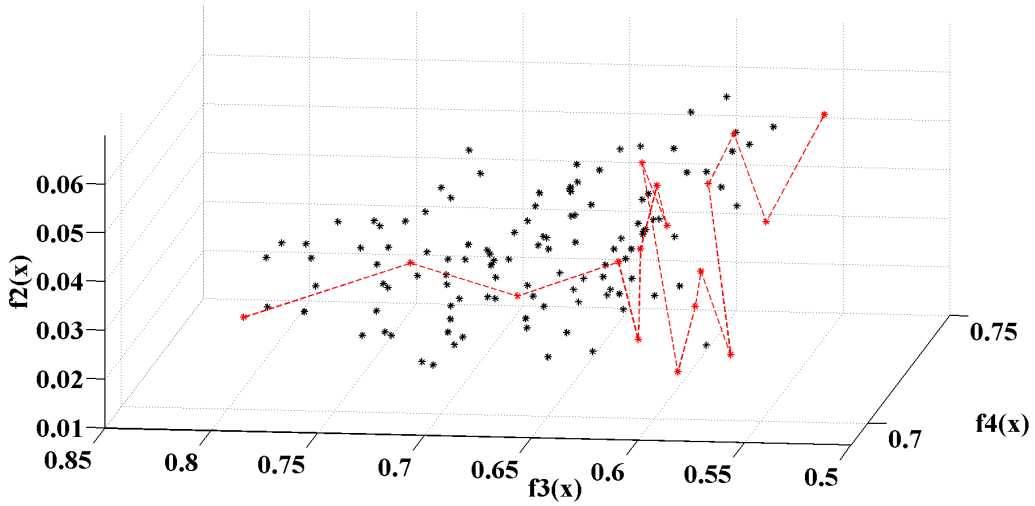


(a) Cenário 1 (b) Cenário 2

Figura 46: Conjunto de Soluções para $f2(x) = 0$.



(a) $f2(x) = 0$

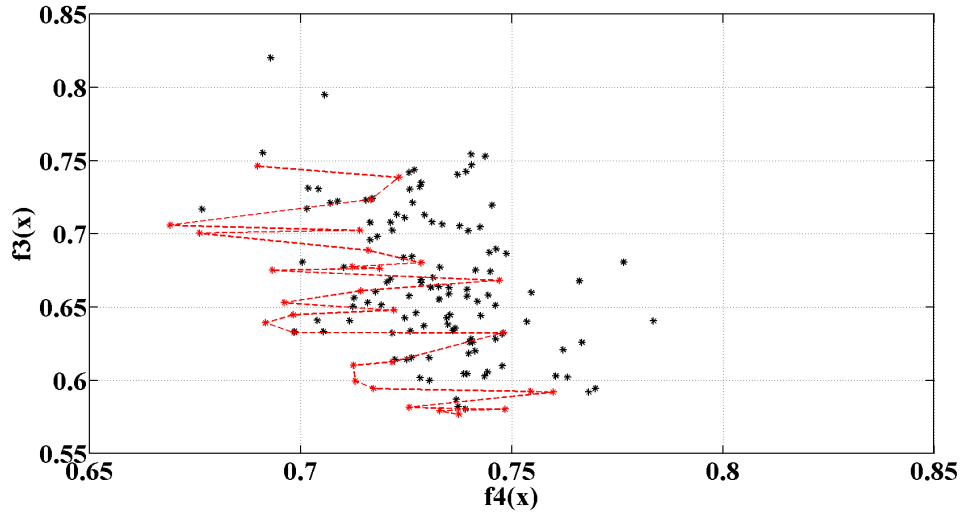


(b) $f2(x) = 0$

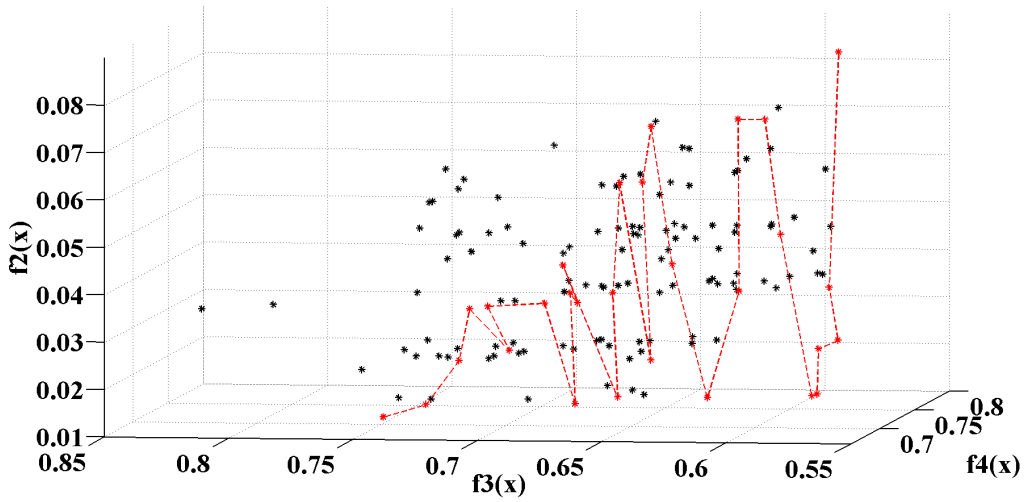
Figura 47: Fronteira de Pareto para três objetivos para o cenário 1.

8.2 CONSIDERAÇÕES

Os resultados apresentados nesse capítulo mostram o bom desempenho do algoritmo proposto *Modified Shuffled Frog-Leaping Pareto Approach* na solução de dois cenários reais da



(a) $f2(x) = 0$



(b) $f2(x) = 0$

Figura 48: Fronteira de Pareto para três objetivos para o cenário 2.

rede de escuros adaptados ao modelo desenvolvido nesta tese. O algoritmo conseguiu obter valores ótimos para os dois principais objetivos na solução da operação de rede de dutos: atendimento as demandas.

Em relação ao objetivo da minimização do tempo, também conseguiu obter bons resultados. Já para o objetivo da fragmentação das bateladas, os resultados são satisfatórios mas poderiam ser melhorados caso se alterasse o peso associado a este objetivo para o cálculo final do valor global. Esse pesos utilizados pelo modelo na equação 43, permitem a flexibilidade de escolha de quais objetivos priorizar para obter os melhores resultados em relação as soluções esperadas pelos especialistas do problema.

Além disso, o MSFLPA obteve um conjunto Pareto-ótimo representativo para o

problema de escalonamento de rede de dutos para ambos cenários, fornecendo um conjunto de soluções não-dominadas viáveis e capaz de representar e construir a fronteira de pareto para o problema apresentado. E o tempo de processamento pode ser considerado razoável, aproximadamente 8 minutos para cada cenário testado, se comparado com outras técnicas de soluções para esse tipo de problema.

9 CONCLUSÃO

A modelagem de sistemas envolvidos no gerenciamento das operações de uma rede de dutos é um problema de otimização que envolve complexas restrições operacionais. Uma abordagem de decomposição do problema deve ser utilizada para tornar viável a implementação de sistema. A subdivisão adotada nesta tese é baseada nos três elementos chave do escalonamento: a determinação e alocação dos recursos a serem utilizados (*assignment*), o sequenciamento de atividades (*sequencing*) e a temporização (*timing*) do uso dos recursos pelas atividades.

Nesse contexto, o presente trabalho teve como principal proposta a de aplicar técnicas metaheurísticas ao problema de escalonamento de operações em uma rede de dutos existente na indústria P & G, em específico nas atividades de alocação de recursos e sequenciamento. A necessidade de estruturas de otimização que possam auxiliar no procedimento de tomada de decisão operacional e o contexto do transporte dutoviário de derivados de petróleo são os elementos motivadores deste trabalho.

As restrições de sequenciamento são um grande complicador na geração de uma solução operacional ótima e a alocação de recursos além de ser um problema combinatório de difícil solução é em geral um problema multiobjetivo. Para resolver este tipo de problema, uma alternativa eficiente é o uso de metaheurísticas adaptadas para problemas multiobjetivos, como os algoritmos evolucionários multiobjetivos.

Dessa forma, este trabalho utilizou um algoritmo evolucionário mémetico, o SFLA, que conforme a literatura tem sido uma boa alternativa para resolver problemas de programação de grande porte com objetivos conflitantes. Uma versão modificada deste algoritmo, na qual é inserido um fator de aceleração nas equações do algoritmo e utiliza uma população reduzida, foi aplicado a rede de escuros, que é parte integrante do sistema dutoviário do estado de São Paulo que é operado pela Transpetro. A partir disso, foi feita uma análise dos resultados obtidos pelo SFLA observando os objetivos de otimização e comparando os resultados com o bem conhecido algoritmo genético (AG), em uma versão que utiliza população reduzida, denominado algoritmo

micro Genético (μAG).

Para essa comparação, foram utilizados dois modelos da rede com quantidade de produtos diferentes, um com 4 produtos distintos e outro com 8 produtos distintos. Os objetivos de otimização para os modelo desenvolvidos são quatro: minimizar o tempo de atendimento da demanda de recebimento; atender a demanda de recebimento no horizonte programado; atender a demanda de fornecimento de produtos e minimizar a quantidade de fragmentação no envio das bateladas.

Em uma visão preliminar, ambos algoritmos obtiveram uma boa convergência na solução do modelo. Mas pode-se notar que para ambos modelos, o SFLA apresentou melhores resultados na solução dos objetivos quando comparado ao algoritmo micro Genético. Foram encontradas soluções ótimas para os objetivos de demanda de recebimento e demanda de fornecimento, e o para os objetivos de minimização do tempo e de envio de bateladas, o SFLA apresentou melhores resultados que o algoritmo micro-genético.

O tempo médio de execução do algoritmo SFLA é maior que o tempo médio do algoritmo μAG, pois o SFLA utiliza um processo de busca local que faz seu tempo de execução ter um acréscimo na sua execução. Porém, o SFLA encontrou resultados melhores em todos os objetivos para os dois conjuntos de simulações apresentados, tanto para o modelo com 4 produtos quanto para o modelo com 8 produtos.

Para o conjunto de soluções não dominadas (cojunto de Pareto), tanto o SFLA quanto o μAG apresentaram um conjunto pequeno de soluções não-dominadas para construir e representar a fronteira de Pareto, surgindo a necessidade de alternativas para encontrar um conjunto maior e representativo para a fronteira. Assim, este trabalho propôs uma nova abordagem utilizando o algoritmo SFLA através do conceito de otimalidade de Pareto, chamado de *Modified Shuffled Frog-Leaping Pareto Approach* (MSFLPA). O MSFLPA utiliza algumas estratégias para favorecer a busca de diferentes pontos na fronteira de Pareto ao longo das gerações.

Essa nova abordagem tem uma população pequena e usa um mecanismo de reinicialização que mantém a diversidade da população e evita a convergência prematura para um super individuo. Além disso, uma estratégia de arquivamento baseado em duas memórias auxiliares é usada para manter as soluções não-dominadas e melhorar a classificação da população e procedimentos de reinicialização

A eficácia do MSFLPA para construir/recobrir a fronteira de Pareto foi demonstrado resolvendo cinco problemas multiobjetivo bem conhecidos da literatura. Esta eficácia é avaliada

por duas métricas e em comparação com dois famosos algoritmos de evolução multiobjetivo, NSGA-II (DEB et al., 2002a) e SPEA2 (ZITZLER et al., 2001). Os experimentos numéricos demonstraram que o MSFLPA é capaz de construir a fronteira de Pareto tão bem quanto os conhecido algoritmos NGSA-II e SPEA2. Além disso, o MSFLPA superou ambos algoritmos em relação as métricas de convergência e diversidade. Para ter um uso prático, o MSFLPA foi aplicado para resolver o problema de escalonamento de um cenário de uma rede de dutos. Várias simulações são analisadas e comparados com o NSGA-II e o algoritmo micro genético (μAG). Ambos os modelos (MSFLPA e μAG) combinam pequenas populações com um procedimento de reinicialização para manter a diversidade da população e para reduzir a carga computacional. Os resultados mostram que o tempo de execução médio do MSFLPA é maior do que o tempo de execução médio de μAG e NSGA-II devido à busca local implementada por MSFLPA. Assim, para uma comparação justa entre MSFLPA e μAG, foi feita uma simulação para o μAG durante aproximadamente 65.2s permitindo o reinício da população. No entanto, os valores finais das funções objetivos foram melhores (perto do ótimo) para MSFLPA do que NSGA-II, μAG e μAG2 ($\cong$ 65.2s). Além disso, o conjunto de soluções não-dominadas viáveis obtidas pelo MSFLPA era maior do que o NSGA-II, μAG e μAG2 ($\cong$ 65.2s), e estas soluções também estão melhores distribuídas ao longo e mais perto da fronteira de Pareto. Quando comparado com o NSGA-II, o conjunto de soluções não-dominadas encontrado por MSFLPA contém as melhores soluções em termos de minimização das funções objetivo para o problema apresentado.

Além disso, com o objetivo de validar o uso do MSFLPA na prática e comprovar se os resultados obtidos estão próximos do esperado para um problema real, dois cenários reais da rede de escuros adaptados ao modelo de otimização multiobjetivo MSFLPA foram utilizados. O MSFLPA alcançou bons resultados em um tempo razoavelmente bom. O algoritmo conseguiu, utilizando as estratégias propostas, favorecer a busca de diferentes pontos na fronteira de Pareto e encontrou conjuntos representativos de solução Pareto-ótimas e boas soluções em relação aos objetivos na resolução de ambos cenários.

Portanto, os resultados mostram que o MSFLPA pode fornecer um conjunto de soluções não-dominadas razoável, que é aproximadamente perto da fronteira de Pareto e uniformemente distribuído ao longo do fronteira obtida, e também pode obter rapidamente uma série de soluções em uma única execução, que é mais conveniente e eficiente para esse tipo de problema (problemas multiobjetivos de escalonamento com mais de dois objetivos conflitantes). Portanto, o algoritmo proposto pode ser usada como uma técnica alternativa eficiente para resolver processos de escalonamento de curto prazo do transporte do produto através de redes de dutos.

Com base em todas as simulações realizadas neste trabalho, pode-se concluir que o MSFLPA apresentou bons resultados (similar ou melhor do que NSGA-II e μAG) para problemas com dois objetivos, caracterizando uma rápida convergência para um conjunto de solução de Pareto viável; que semelhante ao μAG, o procedimento de reinicialização da população assegura a diversidade da população e pode prevenir uma convergência prematura para super-indivíduos com população pequena; que como MSFLPA trabalha com a população dividida em grupos de indivíduos, um conjunto de soluções de Pareto pode ser obtido em apenas uma única simulação que pode abranger todas as restrições do problema e lidar com elas simultaneamente durante a evolução dos grupos dos indivíduos ao longo das gerações.

Dessa forma, este trabalho teve como principal contribuição o desenvolvimento de um novo algoritmo, o MSFLPA, que obteve bons resultados em todas as simulações realizadas demonstrando ser um algoritmo eficiente e competitivo para resolver problemas de escalonamento multiobjetivo com mais de dois objetivos conflitantes. Em específico, mostrou-se que o MSFLPA é uma alternativa boa e eficaz na solução de problemas de alocação de recursos e de transporte do produtos por meio de redes de dutos.

Além disso, conclui-se que o novo algoritmo pode ser utilizado em outros problemas complexos de escalonamento com restrições, e também pode ser aplicado a tipos de classes de problemas que podem ser solucionados por algoritmos genéticos e algoritmos evolucionários multiobjetivo (AEMOs).

9.1 TRABALHOS FUTUROS

A seguir são destacados alguns pontos a serem considerados para a continuidade da pesquisa:

- O modelo considera que os produtos são entregues na forma de bateladas discretas, ou seja o volume da batelada é fixo em valor pré-determinado. Uma mudança que poderia melhorar os resultados seria utilizar um valor variável de volume das bateladas. Isto poderia ser inserido como uma variável/restrição a mais no modelo de otimização ou apenas uma mudança na implementação do modelo para permitir variações nos volumes das bateladas, como realmente funciona em cenários reais;

- O modelo assume a discretização do tempo. A evolução do modelo para tempo contínuo, seria um item muito importante a ser considerado, para resultados mais próximos dos esperados em operações reais. Além disso, acrescentar o conceito de janelas de

tempo (FELIZARI, 2009) para melhorar o desempenho do modelo, permitindo assim um melhor fluxo das batelas nas conexões, melhoria o tempo e o atendimento as demandas.

- O algoritmo proposto MSFLPA, tem como principal desvantagem o tempo de execução, devidas as estratégias desenvolvidas. Um estudo na mudanças de alguns parâmetros dos algoritmo e uma otimização da busca local poderiam ser realizados para tentar alcançar uma melhora no tempo. Uma segunda versão do algoritmo poderia ser desenvolvida, considerando essas alterações, e também considerando algumas outras estratégias de melhorias para encontrar o conjunto Pareto-ótimo já propostas na literatura para esses tipos de algoritmos.

REFERÊNCIAS

ALLE, A. **Técnicas de programação mista-inteira aplicadas ao scheduling de plantas químicas contínuas**. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2003.

ALMEIDA, M. R. **Programação automática da produção em refinarias de petróleo utilizando algoritmos genéticos**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro., 2001.

ALVES, V. R. F. M. **Programação de transferência de derivados de petróleo em rede dutoviária usando algoritmo genético**. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Coordenação dos Programas de Pós-Gradução de Engenharia da Universidade Federal do Rio de Janeiro - COPPE/UFRJ, 2007.

AMIRI, B.; FATHIAN, M.; MAROOSI, A. Application of shuffled frog-leaping algorithm on clustering. **The International Journal of Advanced Manufacturing Technology**, Springer-Verlag, v. 45, n. 1-2, p. 199–209, 2009. ISSN 0268-3768.

ARRUDA, L.; NEVES-JR, F.; YAMAMOTO, L. Using moga to order batches in a real world pipeline network. In: GARCíA-PEDRAJAS, N. et al. (Ed.). **Trends in Applied Intelligent Systems**. [S.l.]: Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6098). p. 546–555.

ARRUDA, L. V. R. et al. Um método evolucionário para sintonia de controladores PI/PID em processos multivariáveis. **Sba: Controle e Automação Sociedade Brasileira de Automatica**, scielo, v. 19, p. 1 – 17, 03 2008.

BARBOZA, A. O. **Simulação e técnicas da computação evolucionária aplicadas a problemas de programação linear inteira mista**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2005.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, 2003.

BODINGTON, C. E.; SHOBRYS, D. E. **Planning, scheduling and control integration in the process industries**. [S.l.]: McGraw-Hill, New York, USA, 1995.

BOSCHETTO, S. N. **Otimização das operações de transferência e estocagem em redes de dutos**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2011.

BOSCHETTO, S. N. et al. An operational scheduling model to product distribution through a pipeline network. **Industrial and Engineering Chemistry Research**, v. 49, n. 12, p. 5661–5682, 2010.

CAFARO, D. C.; CERDÁ, J. Efficient tool for the scheduling of multiproduct pipelines and terminal operations. **Industrial and Engineering Chemistry Research**, v. 47, n. 24, p. 9941–9956, 2008.

CASTRO, H. P. **Utilização de algoritmos genéticos para solução de problema de programação de produção de uma refinaria de petróleo**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2001.

CHUNG, G.; LANSEY, K. Application of the shuffled frog leaping algorithm for the optimization of a general large-scale water supply system. **Water Resources Management**, Springer Netherlands, v. 23, n. 4, p. 797–823, 2009. ISSN 0920-4741.

COELLO, C. A. C. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In: **Proceedings of the 1999 Congress on Evolutionary Computation. CEC 99.** [S.l.: s.n.], 1999. v. 1.

COELLO, C. A. C. Treating constraints as objectives for single-objective evolutionary optimization. **Engineering Optimization**, v. 32, p. 275–308, 2000b.

COELLO, C. A. C. Evolutionary multiobjective optimization: current and future challenges. In: BENíTEZ, J. et al. (Ed.). **Advances in Soft Computing**. [S.l.]: Springer London, 2003. p. 243–256.

COELLO, C. A. C. Recent trends in evolutionary multiobjective optimization. In: ABRAHAM, A.; JAIN, L.; GOLDBERG, R. (Ed.). **Evolutionary Multiobjective Optimization**. [S.l.]: Springer London, 2005. p. 7–32.

COELLO, C. A. C.; CHRISTIANSEN, A. Multiobjective optimization of trusses using genetic algorithms. **Computers e Structures**, v. 75, n. 6, p. 647–660, 2000.

COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. **Evolutionary algorithms for solving multi-objetive problems**. [S.l.]: New York: Springer, 2007.

COELLO, C. A. C.; PULIDO, G. T. A micro-genetic algorithm for multiobjective optimization. In: ZITZLER, E. et al. (Ed.). **Evolutionary Multi-Criterion Optimization**. [S.l.]: Springer Berlin Heidelberg, 2001a, (Lecture Notes in Computer Science, v. 1993). p. 126–140. ISBN 978-3-540-41745-3.

COELLO, C. A. C.; PULIDO, G. T. Multiobjective optimization using a micro-genetic algorithm. In: **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)**. San Francisco, California: Morgan Kaufmann Publishers, 2001b. p. 274–282.

COELLO, C. A. C.; VELDHUIZEN, D. A. V.; LAMONT, G. B. **Evolutionary algorithms for solving multi-objective problems**. New York: Kluwer Academic Publishers, 2002.

CORNE, D.; KNOWLES, J.; OATES, M. The pareto envelope-based selection algorithm for multiobjective optimization. In: SCHOENAUER, M. et al. (Ed.). **Parallel Problem Solving from Nature PPSN VI**. [S.l.]: Springer Berlin Heidelberg, 2000, (Lecture Notes in Computer Science, v. 1917). p. 839–848. ISBN 978-3-540-41056-0.

CORNE, D. W. et al. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In: **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001**. [S.l.]: Morgan Kaufmann Publishers, 2001. p. 283–290.

CRANE, D. S.; WAINWRIGHT, R. L.; SCHOENEFELD, D. A. Scheduling of multi-product fungible liquid pipelines using genetic algorithms. In: **Proceedings of the 1999 ACM symposium on Applied computing**. New York, NY, USA: ACM, 1999. p. 280–285.

CRUZ, J. M. et al. Multiobjective optimization of the transport in oil pipeline networks. In: **IEEE International Conference on Emerging Technologies and Factory Automation**. [S.l.: s.n.], 2003. p. 566–573.

DAS, I.; DENNIS, J. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. **Structural optimization**, Springer-Verlag, v. 14, n. 1, p. 63–69, 1997. ISSN 0934-4373.

DAWKINS, R. **The selfish gene**. Oxford: Oxford University Press, 1976.

DE-MING, L.; ZHI-MING, W. Crowding-measure based multi-objective evolutionary algorithm. **Chinese Journal of Computers**, v. 28, n. 8, p. 1320–1326, 2005.

DEB, K. **Evolutionary algorithms for multi-criterion optimization in engineering design**. 1999.

DEB, K.; AGRAWAL, S. Simulated binary crossover for continuos search space. **Complex Systems**, v. 9, p. 115–148, 1995.

DEB, K.; GOLDBERG, D. E. An investigation of niche and species formation in genetic function optimization. In: **Proceedings of the 3rd International Conference on Genetic Algorithms**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 42–50. ISBN 1-55860-066-3.

DEB, K.; KALYANMOY, D. **Multi-objective optimization using evolutionary algorithms**. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 047187339X.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, Apr 2002a. ISSN 1089-778X.

DEB, K. et al. Scalable multi-objective optimization test problems. In: **Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC 02.** [S.l.: s.n.], 2002b. v. 1, p. 825–830.

DREO, J. et al. **Metaheuristics for hard optimization**. [S.l.]: Springer-Verlag, Heidelberg, 2006.

ELBEHAIRY, H. et al. Comparison of two evolutionary algorithms for optimization of bridge deck repairs. **Computer-Aided Civil and Infrastructure Engineering**, Blackwell Publishing Inc, v. 21, n. 8, p. 561–572, 2006. ISSN 1467-8667.

ELBELTAGI, E.; HEGAZY, T.; GRIERSON, D. Comparison among five evolutionary-based optimization algorithm. **Advanced Engineering Informatics**, v. 19, p. 43–53, 2005.

ELBELTAGI, E.; HEGAZY, T.; GRIERSON, D. A modified shuffled frog-leaping optimization algorithm: applications to project management. **Structure and Infrastructure Engineering**, v. 3, p. 53–6, 2007.

ERICKSON, M.; MAYER, A.; HORN, J. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. In: ZITZLER, E. et al. (Ed.). **Evolutionary Multi-Criterion Optimization**. [S.l.]: Springer Berlin Heidelberg, 2001, (Lecture Notes in Computer Science, v. 1993). p. 681–695. ISBN 978-3-540-41745-3.

EUSUFF, M.; LANSEY, K. E.; PASHA, F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. **Engineering Optimization**, v. 38, p. 129–154, 2006.

EUSUFF, M. M.; LANSEY, K. E. Optimization of water distribution network design using the shuffled frog leaping algorithm. **Water Resources Planning Mgmt**, v. 129, p. 210–225, 2003.

FELIZARI, L. C. **Programação das operações de transporte de derivados de petróleo em redes de dutos**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2009.

FELIZARI, L. C. et al. Sequencing batches in a real-world pipeline network using constraint programming. In: ALVES, R. M. de B.; NASCIMENTO, C. A. O. do; BISCAIA, E. C. (Ed.). **10th International Symposium on Process Systems Engineering: Part A**. [S.l.]: Elsevier, 2009, (Computer Aided Chemical Engineering, v. 27). p. 303 – 308.

FONSECA, C.; FLEMING, P. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: **Proceedings of the Fifth International Conference on Genetic Algorithms**. [S.l.: s.n.], 1993. p. 416–423.

GARCIA, J. M. C. et al. Hybrid heuristic and mathematical programming in oil pipelines networks. In: **IEEE Congress on Evolutionary Computation**. [S.l.: s.n.], 2004. v. 2, p. 1479–1486.

GENDREAU, M.; POTVIN, J. Y. Metaheuristics in combinatorial optimization. **Annals of Operations Research**, Springer Netherlands, v. 140, p. 189–213, 2005.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers and Operations Research**, v. 13, n. 5, p. 533–549, 1986.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear - modelos e algoritmos**. [S.l.]: Editora Campus, Rio de Janeiro, 2000.

GOLDBERG, D. E. **Genetic algorithm in search, optimization and machine learning**. [S.l.]: Addison-Wesley Publishing, 1989.

GOLDBERG, D. E.; DEB, K. A comparative analysis of selection schemes used in genetic algorithms. In: **Foundations of Genetic Algorithms**. [S.l.]: Morgan Kaufmann, 1991. p. 69–93.

GREFENSTETTE, J. J. Optimization of control parameters for genetic algorithms. **IEEE transactions on Systems, Man and Cybernetics**, v. 16, p. 122–128, 1986.

HERTZ, A.; KOBLER, D. A framework for the description of evolutionary algorithms. **European Journal of Operational Research**, v. 126, p. 1–12, 2000.

HERTZ, A.; WIDMER, M. Guidelines for the use of meta-heuristics in combinatorial optimization. **European Journal of Operational Research**, Elsevier, v. 151, p. 247–252, 2003.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Ann Arbor, MI: University of Michigan Press, 1975.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. A niched pareto genetic algorithm for multiobjective optimization. In: **Proceedings of the First IEEE Conference on Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence**. [S.l.: s.n.], 1994. p. 82–87 vol.1.

HUBAND, S. et al. A review of multiobjective test problems and a scalable test problem toolkit. **IEEE Transactions on Evolutionary Computation**, v. 10, n. 5, p. 477–506, Oct 2006. ISSN 1089-778X.

IERAPETRITOU, M. G.; FLOUDAS, C. A. Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. **Industrial and Engineering Chemistry Research**, v. 37, n. 11, p. 4341–4359, 1998.

JIN, Y.; OLHOFER, M.; SENDHOFF, B. Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how? In: **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)**. San Francisco, California: Morgan Kaufmann Publishers, 2001. p. 1042–1049.

JOLY, M. **Técnicas de otimização mista-inteira para o scheduling e gerenciamento da produção em refinarias de petróleo**. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 1999.

KNOWLES, J. D.; CORNE, D. W. Approximating the nondominated front using the pareto archived evolution strategy. **Evolutionary Computation**, MIT Press, Cambridge, MA, USA, v. 8, n. 2, p. 149–172, jun 2000. ISSN 1063-6560.

LAKSHMI, K.; RAO, A. A memetic algorithm for combinatorial problems with multiple objectives. In: **Second International Conference on Advanced Computing (ICoAC)**. Chennai, India: [s.n.], 2010. p. 33–41.

LAMBOIA, F.; ARRUDA, L. V. R.; NEVER-JR, F. Otimização multiobjetivo do transporte de produtos em redes dutoviárias através do algoritmo shuffled frog-leaping modificado. In: **XLIV Simposio Brasileiro de Pesquisa Operacional**. Rio de Janeiro, RJ, Brasil: SOBRAPO, 2012. p. 2658–2669.

LAMBOIA, F.; ARRUDA, L. V. R.; NEVES-JR, F. A modified shuffled frog-leaping algorithm to model products transport in pipeline networks. In: **EngOpt - IV International Conference on Engineering Optimization**. Lisboa, Portugal: [s.n.], 2014. p. 885–890.

LEE, H. et al. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. **Industrial and Engineering Chemistry Research**, v. 35, n. 5, p. 1630–1641, 1996.

LEE, H. C.; DAGLI, C. H. A. A parallel genetic-neuro scheduler for job-shop scheduling problems. **International Journal of Production Economics**, v. 51, p. 115–122, 1997.

LI, Y. et al. Novel multiobjective shuffled frog leaping algorithm with application to reservoir flood control operation. **Journal of Water Resources Planning and Management**, v. 136, n. 2, p. 217–226, 2010.

LINARES, P. et al. **Modelos matemáticos de optimizácion**. Relatório Técnico, Universidad Pontificia de Comillas, Madrid, 2001.

MACIEL, R. S. **Otimização multiobjetivo na análise da integração de geração distribuída às redes de distribuição**. Tese (Doutorado) — Universidade Estadual Paulista, Faculdade de Engenharia - UNESP, Campus de Ilha Solteira - SP, 2012.

MAGALHãES, M. V. O. **Refinery scheduling**. Tese (Doutorado) — Imperial College London, London, United Kingdom, 2004.

MAGATãO, L. **Mixed integer linear programming and constraint logic programming: Towards a unified modeling framework**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2005.

MAGATãO, L.; ARRUDA, L.; NEVER-JR, F. A mixed integer programming approach for scheduling commodities in a pipeline. **Computers and Chemical Engineering**, v. 28, n. 1 - 2, p. 171 – 185, 2004.

MÉNDEZ, C. A.; CERDÁ, J. An efficient milp continuous-time formulation for short-term scheduling of multiproduct continuous facilities. **Computers and Chemical Engineering**, v. 26, n. 4-5, p. 687–695, 2002.

MICHALEWICZ, Z. **Genetic algorithms + Data structures = evolution programs**. [S.l.]: Springer-Verlag Berlin Heidelberg, 1996.

MIETTINEN, K. M. **Nonlinear multiobjective optimization**. Boston, Massachusetts: Kluwer Academic Publishers, 1998.

MOCKUS, L.; REKLAITIS, G. V. Mathematical programming formulation for scheduling of batch operations based on nouniform time discretization. **Computers and Chemical Engineering**, v. 21, p. 1147–1156, 1997.

MORO, L. F. L. **Técnicas de otimização mista-inteira para o planejamento e programação de produção em refinarias de petróleo**. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 1999.

NEVES-JR F., M. L. et al. An efficient approach to the operational scheduling of a real-world pipeline network. In: PLESU, V.; AGACHI, P. S. (Ed.). **17th European Symposium on Computer Aided Process Engineering**. [S.l.]: Elsevier, 2007, (Computer Aided Chemical Engineering, v. 24). p. 697–702.

OEI, C.; GOLDBERG, D.; CHANG, S.-J. **Tournament selection, niching, and the preservation of diversity**. Urbana, Illinois: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1991.

OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. **Annals of Operations Research**, Springer Netherlands, v. 63, n. 5, p. 511–623, 1996.

PARETO, V. **Cours D'Economie Politique**. Rouge, Lausanne, Switzerland: [s.n.], 1896.

PEKNY, J. F.; ZENTNER, M. G. **Learning to solve process scheduling problems: The rule of rigorous knowledge acquisition frameworks**. [S.l.]: School of Chemical Engineering, Purdue University, West Lafayette, USA, 1993.

PINTO, J. M.; GROSSMANN, I. E. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. **Industrial and Engineering Chemistry Research**, v. 34, n. 9, p. 3037–3051, 1995.

RAHIMI-VAHED, A. et al. A novel hybrid multi-objective shuffled frog-leaping algorithm for a bi-criteria permutation flow shop scheduling problem. **The International Journal of Advanced Manufacturing Technology**, Springer-Verlag, v. 41, n. 11-12, p. 1227–1239, 2009. ISSN 0268-3768.

RAHIMI-VAHED, A.; MIRZAEI, A. H. Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. **Soft Computing**, Springer-Verlag, Berlin, Heidelberg, v. 12, n. 5, p. 435–452, dec 2007. ISSN 1432-7643.

REJOWSKI, R.; PINTO, J. M. Scheduling of a multiproduct pipeline system. **Computers and Chemical Engineering**, v. 27, n. 8-9, p. 1229–1246, 2003.

REKLAITIS, G. Overview of scheduling and planning of batch process operations. In: **Proceedings of the NATO Advanced Study Institute on Batch**. [S.l.: s.n.], 1992. p. 660–705.

REKLAITIS, G. Scheduling approaches for the batch process industries. **ISA Transactions**, Elsevier, v. 34, p. 349–358, 1995.

RIBAS, P. C. et al. A micro-genetic algorithm for multi-objective scheduling of a real world pipeline network. **Engineering Applications of Artificial Intelligence**, v. 26, n. 1, p. 302–313, 2013.

SASIKUMAR, M. et al. Pipes: A heuristic search model for pipeline schedule generation. **Knowledge-Based Systems**, v. 10, n. 3, p. 169–175, 1997.

SCHAFFER, J. **Some experiments in machine learning using vector evaluated genetic algorithms**. Tese (Doutorado) — Vanderbilt University, Nashville, 1984.

SCHAFFER, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In: **Proceedings of the 1st International Conference on Genetic Algorithms**. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985. p. 93–100. ISBN 0-8058-0426-9.

SCHOTT, J. R. **Fault tolerant design using single and multicriteria genetic algorithm optimization**. Dissertação (Mestrado) — Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics, Cambridge, MA, 1995.

SILVER, E. A. An overview of heuristic solution methods. **Journal of the Operational Research Society**, Haskayne School of Business, The University of Calgary, v. 55, p. 936–956, 2002.

SRINIVAS, N.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary Computation**, v. 2, n. 3, p. 221–248, 1994.

STEBEL, S. et al. Modelling liquefied petroleum gas storage and distribution. In: GRIEVINK, J.; SCHIJNDEL, J. van (Ed.). [S.l.]: Elsevier, 2002, (Computer Aided Chemical Engineering, v. 10). p. 805–810.

STEBEL, S. L. **Técnicas de otimização aplicadas em problemas de scheduling dos recursos de estocagem**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2006.

TIWARI, S. et al. Amga: an archive-based micro genetic algorithm for multi-objective optimization. In: **Proceedings of Genetic and Evolutionary Computation conference (GECCO-2008)**. Atlanta, USA: [s.n.], 2008. p. 729–736.

TRANSPETRO. **Acessado em**. maio, 2013. Disponível em: <http://www.transpetro.com.br/>.

VELDHUIZEN, D. A. V. **Multiobjective evolutionary algorithms: classifications, analyses, and new innovations**. [S.l.], 1999.

VELDHUIZEN, D. A. V.; LAMONT, G. B. **Multiobjective evolutionary algorithm research: A history and analysis**. 1998.

WALPOLE, R. E. et al. **Probability and statistics for engineers and scientists**. Prentice Hall Upper Saddle River, NJ: Pearson Prentice Hall, 1998. ISBN 132047675.

WESTPHAL, H. **Algoritmo genético aplicado à otimização multiobjetivo em redes de distribuição de petróleos e derivados**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2006.

WESTPHAL, H.; ARRUDA, L. V. R. Multiobjective optimization applied to the distribution of petroleum products in pipelines networks. In: **Proceedings of 17th European Symposium on Computer Aided Process Engineering**. [S.l.]: Elsevier, 2007. p. 1–6.

WESTPHAL, H.; NEVER-JR, F.; ARRUDA, L. V. R. Algoritmo micro-genético aplicado ao scheduling de uma rede de distribuição de derivados de petróleo. In: ____. **Computação Evolucionária em Problemas de Engenharia**. 1. ed. Curitiba (PR): Omnipax, 2011. p. 331–354.

YAMAMOTO, L. **Um modelo baseado em metaheurística para o seqüenciamento de bateladas em redes dutoviárias**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI, Curitiba - PR, 2009.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: empirical results. **Evolutionary Computation**, MIT Press, Cambridge, MA, USA, v. 8, n. 2, p. 173–195, jun 2000. ISSN 1063-6560.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. **SPEA2: Improving the Strength Pareto Evolutionary Algorithm**. Swiss Federal Institute of Technology (ETH), 2001.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: **Evolutionary Methods for Design, Optimisation, and Control**. [S.l.]: CIMNE, Barcelona, Spain, 2002. p. 95–100.

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