ESCOLA POLITÉCNICA

Departamento de Sistemas e Computação

UMA FERRAMENTA DE SIMULAÇÃO PARA OTIMIZAÇÃO MULTI-OBJETIVA EVOLUCIONÁRIA

Trabalho de Conclusão de Curso

Engenharia da Computação

Filipe Rolim Cordeiro
Orientador: Prof. Carmelo José Albanez Bastos Filho

Recife, novembro de 2008

UNIVERSIDADE DE PERNAMBUCO

ESCOLA POLITÉCNICA

UMA FERRAMENTA DE SIMULAÇÃO PARA OTIMIZAÇÃO MULTI-OBJETIVA EVOLUCIONÁRIA

Trabalho de Conclusão de Curso

Engenharia da Computação

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Filipe Rolim Cordeiro
Orientador: Prof. Carmelo José Albanez Bastos Filho

Recife, novembro de 2008

UNIVERSIDADE DE PERNAMBUCO

FILIPE ROLIM CORDEIRO

UMA FERRAMENTA DE SIMULAÇÃO PARA OTIMIZAÇÃO MULTI-OBJETIVA EVOLUCIONÁRIA

Resumo

Este trabalho propõe uma ferramenta de simulação e análise de Algoritmos Evolucionários Multi-Objetivos, onde é disponibilizado ao usuário, através de um ambiente gráfico intuitivo, a possibilidade de simulação de quatro das principais técnicas de otimização multi-objetiva: NSGAII, PAES, PESA2 e SPEA2. Para efeitos de validação e análise do comportamento dos algoritmos é disponibilizado um conjunto de 36 funções de teste e 8 métricas de análise.

Inicialmente é realizada uma breve descrição das técnicas de otimização utilizadas, observando-se também a importância das funções de teste e métricas envolvidas no projeto. Este trabalho tem como objetivo demonstrar a capacidade da ferramenta desenvolvida como instrumento de apoio a pesquisas na área de otimização multi-objetiva, assim como apresentar uma arquitetura modular que permita a adição de trabalhos futuros.

Utilizando a ferramenta proposta foi possível realizar estudos comparativos entre os algoritmos disponibilizados, analisando-se os resultados obtidos através de diferentes funções de teste. Pôde-se concluir que o algoritmo PAES apresentava melhores resultados para a função de teste Poloni, quando comparado aos outros algoritmos. Também foi possível observar como a alteração dos parâmetros de simulação pode influenciar na qualidade dos resultados obtidos.

Abstract

This work presents a computational tool for simulation and analysis of Multi-Objective Evolutionary Algorithms. The tool has an intuitive graphical environment and allows the user to simulate four different main multi-objective optimization techniques: NSGAII, PAES, PESA2 and SPEA2. 36 test functions and 8 metrics are available to help in the validation and analysis processes.

In this work, a brief description of the optimization techniques used is presented, observing the importance of the test functions and metrics involved in the project. This work aims to show the ability of the developed tool as a support research resource in the area of multi-objective optimization, and present a modular architecture that allows the addition of other modules..

Using this tool, some comparative studies between the available algorithms were performed in order to present the simulation tool features. We concluded that the PAES algorithm presented better results to the Poloni test function when compared to other algorithms. It was also possible to show that the simulation parameters values can influence the quality of the results.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

EA – Algoritmo Evolucionário

PAES – *Pareto Archived Evolution Strategy*

NSGA II – *Nondominated Sorting Genetic Algorithm II*

SPEA2 – *Strength Pareto Evolutionary Algorithm 2*

PESA2 – *Pareto Envelope based Selection*

MOEA – Algoritmos Evolucionários Multi-Objetivos

MOP – Problema de Otimização Multi Objetivo

NFL – *No Free Lunch*

Agradecimentos

Agradeço a Deus, por ter me ajudado nos momentos mais difíceis e ter me dado saúde e força para chegar até aqui.

À minha família, em especial aos meus pais, por toda criação e educação e apoio que me deram e continuam a dar ainda hoje.

Ao meu professor orientador Carmelo Filho, que sempre foi um modelo de motivação e dedicação pelos trabalhos realizados. Agradeço à orientação recebida e a forma como foi realizada, buscando o incentivo e mostrando que os problemas podem ser vistos de uma forma simples, interessante e desafiadora.

Ao professor Abel Guilhermino, por ter me orientado na Iniciação Científica e ter contribuído bastante na minha formação acadêmica.

Ao meu amigo Gilliard Alan, que me ajudou muito na conclusão desse trabalho, contribuindo na implementação de mais de 30 funções de teste, com equações nada triviais.

Agradeço a todos os meus amigos de turma, pelos momentos de diversão, trabalhos em equipe, apoio e estudo, sem os quais não estaria aqui.

Agradeço ainda a todos que de forma direta ou indireta contribuíram para a conclusão deste trabalho.

Capítulo 1

Introdução

A cada instante é possível observar problemas cujas soluções trariam maior benefício se fossem otimizadas. Seja na indústria, comércio ou vida pessoal, a busca pela solução de um problema sempre visa alcançar a melhor solução viável possível. Porém, a análise dessas soluções muitas vezes envolve um custo alto, o que pode se tornar inviável caso seja feita a análise de um número elevado de possibilidades.

Um problema de otimização consiste em encontrar as melhores soluções que satisfaçam às restrições de um domínio. A busca dessas soluções é baseada nos objetivos que se desejam atingir com a solução encontrada. Dessa forma, é possível observar constantemente a presença de problemas de otimização em diversos setores da sociedade, uma vez que a otimização apresenta vantagens para o usuário. Cada vez mais, devido à alta competitividade do mercado, percebe-se a importância das técnicas de otimização, destacando-se objetivos como maximização de lucros e redução de custos. No entanto, essas buscas muitas vezes envolvem objetivos conflitantes e um elevado número de soluções possíveis, dificultando o processo de otimização.

Problemas de otimização multi-objetivos (MOP) são problemas que envolvem mais de um objetivo a ser otimizado e constituem a maioria dos problemas reais encontrados. Eles são caracterizados pela dificuldade em encontrar uma solução que satisfaça simultaneamente todos os objetivos, sendo necessário muitas vezes penalizar alguns objetivos para se encontrar uma solução mais adequada ao problema. Por essa razão, é comum realizar a busca por um conjunto de soluções mais viáveis ao problema, ao invés de apenas uma, deixando a decisão ser tomada de acordo com o nível de prioridade de cada objetivo.

Na maioria dos casos torna-se inviável analisar todas as soluções do domínio do problema devido ao elevado custo computacional e tempo requerido. Sendo assim, é importante a aplicação de algoritmos de otimização de problemas multi-objetivos a fim de evitar buscas desnecessárias no conjunto de soluções, permitindo redução de tempo e custo envolvidos na busca das soluções.

Coello [1] relata que a comunidade científica tem contribuído com novas abordagens para solução de problemas multi-objetivos desde a década de 50. Deb [2], observa que métodos clássicos encontram no máximo uma solução a cada simulação, tornando esses métodos inconvenientes para resolver MOPs. No entanto, métodos estocásticos de busca foram propostos posteriormente, ganhando destaque os algoritmos evolucionários (EA), que se baseiam em princípios da evolução para encontrar as soluções ótimas.

Os EAs baseiam-se no uso de população de soluções, que evoluem a cada iteração. Desta forma, é possível obter um conjunto de soluções a cada iteração. Os EAs também utilizam o conceito de Pareto *Front*, definido em [3]. Diz-se que uma solução pertence ao Pareto *Front* se não houver nenhuma outra solução possível que domine ela, ou seja, se nenhuma outra solução for melhor que ela em todos os objetivos.

Baseado nos EAs, diversos algoritmos evolucionários de otimização multi-objetiva (MOEA) foram propostos, destacando-se algoritmos como PAES (*Pareto Archived Evolution Strategy*) [4], NSGA-II (*Nondominated Sorting Genetic Algorithm II*) [5], SPEA2 (*Strength Pareto Evolutionary Algorithm 2*) [6] e PESA-II (*Pareto Envelope based Selection II*) [7]. Cada algoritmo tem suas vantagens e desvantagens, tornando-os apropriados a cada tipo de problema.

Silva-Filho [8] realiza uma aplicação de otimização de memória cache utilizando o algoritmo NSGAII para encontrar soluções de configuração de arquitetura de memória que atinjam os objetivos de redução de consumo de energia e aumento do desempenho. No mesmo trabalho é mostrado como uma aplicação real pode se beneficiar através da utilização de algoritmos de otimização multi-objetiva.

Em [9] é apresentada a ferramenta jMetal, que realiza a implementação de alguns MOEAs existentes, permitindo ao usuário realizar simulações baseado em funções de teste. Esse tipo de ferramenta tem grande importância para o pesquisador e projetista, uma vez que permite analisar as melhores técnicas existentes para a solução do problema, assim como possibilita a análise de características relevantes para o desenvolvimento de técnicas híbridas.

Esse documento foi estruturado em capítulos, conforme é descrito a seguir. O capítulo 2 apresenta o conceito de funções de teste e métricas de avaliação de desempenho, apresentando as principais funções de teste e métricas utilizadas pela comunidade científica mundial, assim como neste trabalho. No capítulo 3 são descritos os MOEAs implementados na ferramenta desenvolvida. No capítulo 4 é apresentada a metodologia de implementação da ferramenta, citando os métodos de modularização utilizados e modo de implementação dos algoritmos. No capítulo 5 é analisado um estudo de caso executado pela ferramenta desenvolvida. Ao final, o

capítulo 6 apresenta as considerações finais, além de propor aspectos que podem ser melhorados na ferramenta.

Capítulo 2

Funções de Teste e Métricas de Avaliação de Algoritmos Multi-Objetivos

Neste capítulo são apresentados os principais conceitos de funções de teste e métricas de avaliação de algoritmos multi-objetivos. No entanto é importante realizar a descrição de alguns conceitos gerais presentes à maioria dos MOEAs e que podem ser necessários para uma melhor compreensão dos termos utilizados a partir deste capítulo.

2.1 Conceitos Gerais

Nesta seção são apresentados alguns dos conceitos presentes na utilização de funções de teste, métricas, e algoritmos multi-objetivos. A seguir é realizado um breve resumo sobre os principais termos utilizados na área de otimização multi-objetiva.

2.1.1 População

O conceito de população está presente na maioria dos Algoritmos Evolucionários. Uma população representa o conjunto atual de indivíduos encontrados em uma determinada iteração do algoritmo. A idéia é que a população evolua com o tempo, gerando soluções mais interessantes para a solução de um determinado problema. Normalmente o conjunto de soluções final está contido na última iteração da população.

2.1.2 Geração

Uma geração corresponde a uma iteração realizada sobre a população, de forma que realize alguma mudança sobre os indivíduos da população. Muitas vezes o critério de parada do algoritmo é definido pelo número de gerações. O número de gerações necessárias para o algoritmo convergir está diretamente associada ao número de iterações executadas, sendo muitas

vezes relacionado com o grau de eficiência do algoritmo. Entretanto, existem outros critérios de parada, como, por exemplo, o algoritmo de otimização pára quando as soluções na iteração atingem uma qualidade mínima.

2.1.3 Crossover

A operação de *crossover* é um dos métodos utilizados para promover a evolução genética da população. Ela é uma das formas de possibilitar a criação de novos indivíduos, permitindo a obtenção de novas soluções a partir de outras já existentes. O funcionamento do *crossover* é descrito a seguir. Inicialmente são escolhidos dois indivíduos da população para serem os pais. Após isso, em cada indivíduo escolhido são definidos pontos de corte onde haverá troca de dados entre as características genéticas dos indivíduos. Essa troca é ilustrada na Figura 1.

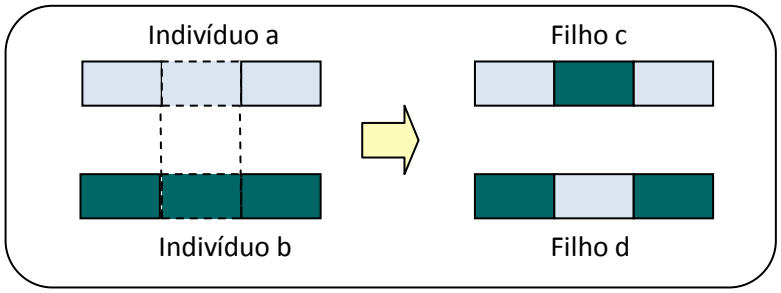

Figura 1. Operação de *crossover* de dois pontos.

Após realizado o *crossover* são gerados novos elementos contendo dados de ambos os indivíduos que iniciaram a operação de *crossover*.

2.1.4 Mutação

A mutação é outro operador importante que visa garantir a variabilidade genética da população. Seu conceito consiste em mutar aleatoriamente um indivíduo existente de forma que um novo indivíduo seja criado. O processo de mutação é ilustrado na Figura 2.

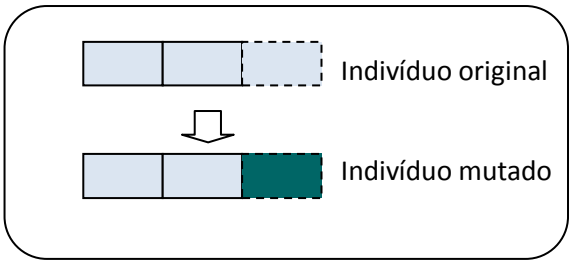

Figura 2. Operação de mutação.

2.1.5 Dominância

O conceito de dominância é bastante utilizado em algoritmos multi-objetivos e serve para indicar e comparar o grau de qualidade de uma solução em relação à outra. Diz-se que uma solução *a* domina *b*, se, e somente se, para todos os objetivos envolvidos no domínio da função, *a* não for pior que *b* em nenhum objetivo e existir pelo menos um objetivo em que *a* seja melhor que *b.* A Figura 3 ilustra a relação de dominância entre diferentes pontos do conjunto de solução.

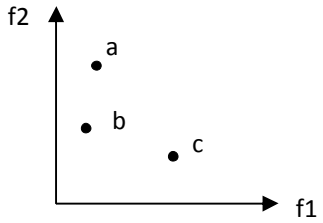

Figura 3. Relação de dominância entre soluções.

No exemplo da Figura 3, existem pontos que compõem o conjunto de soluções encontradas: *a, b* e *c.* De acordo com a definição, pode-se dizer que a solução *b* domina *a*, pois ela é melhor que *a* em pelo menos um objetivo, no caso o objetivo *f1*, e também não é pior que *a* em nenhum deles. De forma análoga, não se pode dizer que a solução *b* domina c, pois a solução *b* é pior que *c* em pelo menos um objetivo, no caso *f2*. Dessa forma, pode-se dizer que *b* domina *a*, mas não domina *c*. Com esse conceito, pode-se dizer também que *a* não domina *c*, pois existe pelo menos um objetivo em *a* que é pior que em *c*. O mesmo vale para afirmar que *c* não domina *a.* Esta relação em que uma solução não é dominada por outra é chamada de não-dominância. Normalmente costuma-se dizer que uma solução é não-dominada como sendo aquela que apresenta uma relação de não-dominância com todas as outras soluções do conjunto de soluções.

Outro conceito que envolve dominância é o de se afirmar que uma solução é fracamente dominada por outra, o que pode ser representado pelo termo $\preccurlyeq$. Sendo assim, a expressão $a \preccurlyeq b$ indica que a solução *a* é fracamente dominada pela solução *b.* Isso quer dizer que para todos os objetivos existentes, a solução *a* não é pior do que *b* em nenhum deles, podendo ser igual. Uma solução seria fortemente dominada por outra se para todos os objetivos ela apresentasse melhores resultados.

2.1.6 Front

***Front* é o conjunto de soluções de mesmo nível de não-dominância que compõem um Pareto. O Pareto *Front* representa o conjunto de soluções que compartilham entre si a mesma quantidade de indivíduos pelos quais são dominados. A**

Figura 4 ilustra a diferença entre níveis de Paretos *Front.*

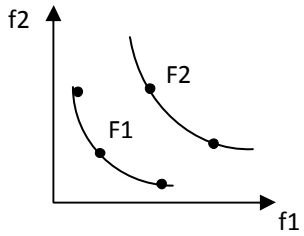

Figura 4. Diferença entre níveis de Paretos *Front.*

Como pode ser visto na

Figura 4, as soluções do *front F1* não são dominados por nenhuma outra solução, se aproximando mais do Pareto Ótimo. Já as soluções do *front F2* são dominadas por pelo menos uma solução do front *F1*, o que representam soluções menos interessantes para a seleção final.

2.1.7 Elitismo

O conceito de elitismo quer dizer que a melhor solução encontrada em cada iteração do algoritmo irá ser preservada para a próxima iteração. Embora essa técnica garanta que as melhores soluções encontradas sejam sempre mantidas, existem técnicas de otimização que não utilizam esse conceito.

2.2 Importância das Funções de Teste

A utilização de problemas de teste na análise de algoritmos estudados ou desenvolvidos é de grande importância para validação e comparação dos trabalhos realizados. Sem os problemas de teste não se tem como avaliar coerentemente as características de uma solução desenvolvida. Os testes são importantes para se estudar o comportamento do algoritmo, avaliar vantagens e desvantagens e realizar análises comparativas. Dessa forma, os problemas de teste são fundamentados para a comprovação de uma nova abordagem proposta pelo pesquisador.

Veldhuizen [10] observa que na história da literatura, a comparação visual de MOEAs não tem se mostrado adequada para determinar o desempenho e eficiência desses algoritmos. Sendo assim, é importante para a etapa de validação selecionar funções de teste que possam representar bem as principais vantagens da solução proposta, porém sem ocultar elementos importantes inerentes ao contexto da aplicação.

Problemas de teste são utilizados para comparar diferentes MOEAs, uma vez que se conhece o Pareto ótimo da aplicação e pode-se então avaliar parâmetros como convergência e qualidade de representação. Diversos problemas de teste têm sido propostos e utilizados para

estudar o comportamento dos MOEAs. Teoremas NFL (*No Free Lunch*) [11] mostram que para um MOEA poder oferecer garantia de eficiência genérica é necessário incluir conhecimento do domínio do problema no domínio do algoritmo. No entanto, a inclusão de muitas características do domínio do problema poderá influenciar negativamente na eficiência do algoritmo, sendo importante estabelecer um limite entre características a serem exploradas pelo algoritmo naquele domínio. Problemas diferentes tendem a explorar características diferentes dos algoritmos, sendo necessário utilizar um conjunto diversificado de testes para estudar as distintas características do algoritmo que está sendo aplicado. De Jong [12], sugere cinco funções de teste com as seguintes características [13]: contínuo ou descontínuo, convexo ou não-convexo, unimodal ou multimodal, quadrático ou não quadrático, baixa ou alta dimensionalidade e determinístico ou estocástico.

Problemas de teste devem conter elementos similares ao de uma aplicação em contexto real. Whitley *et al.* [14] sugere algumas diretrizes em relação à utilização de conjuntos de teste, tais como conter problemas não-separáveis, não-lineares e assimétricos. Adicionalmente, é importante que o conjunto de testes contenha características comuns à maioria dos MOPs conhecidos, caso contrário o algoritmo poderá mostrar-se eficiente apenas a um determinado domínio de testes.

A maioria dos MOPs encontrados na literatura utiliza apenas duas funções objetivo, o que, segundo David [10], podem não avaliar completamente o desempenho de um MOEA, a não ser que o espaço de busca seja muito grande.

2.3 Características Principais das Funções de Teste

Cada problema de teste possui características específicas que podem auxiliar ou dificultar a busca de um algoritmo. No entanto existem características presentes em todas as funções de teste, onde o conhecimento delas pode ajudar na escolha do conjunto de testes mais apropriado. Dentre essas características estão conectividade, simetria, número de variáveis de decisão, número de restrições, geometria do Pareto e concavidade.

A conectividade expressa se o Pareto é conectado ou desconectado. Paretos desconectados tendem a oferecer maior dificuldade aos MOEAs. A simetria indica que existem soluções equivalentes. A geometria do Pareto pode ser composta de uma ou mais curvas, assim como um ou mais pontos. Uma curva é caracterizada como um conjunto de pontos não-distantes entre si. Quanto à concavidade, o Pareto poder ser côncavo ou convexo.

A seguir é apresentada uma tabela com as principais características das funções de testes desenvolvidas e encontradas na literatura. As características estão divididas em genótipo e

fenótipo. O genótipo é caracterizado pelo Pareto referente ao domínio das variáveis de decisão (PTrue), enquanto o fenótipo representa o Pareto referente ao domínio dos objetivos (PFtrue).

Idealmente, um conjunto de teste deve conter combinações de todas as possíveis características.

Tabela 1. Características principais das Funções de Teste.

	Genótipo						Fenótipo				
Função	Conectado	Desconectado	Simétrico	Variáveis de decisão	# Funções	Restrições	Geometria	Conectado	Desconectado	Côncavo	Convexo
Binh	x		x	2	2	2	Curva	x			x
Binh (3)	x			2	3	2	Ponto				
DTLZ1					n	1	Hyperplano	x			
DTLZ2					n	1	Esférica	x			
DTLZ3					n						
DTLZ4					n	1	Esférica	x			
DTLZ5					n	1	Esférica	x			
DTLZ6									x		
DTLZ7					n	n	Linha e Hyperplano				
Fonseca	x		x	2	2	0	Curva	x			
Fonseca (2)	x		x	n	2	n	Curva	x		x	
Kursawe		x	x	n	2	0	Curva		x	x	
Laumanns	x		x	2	2	2	Pontos		x	x	
Lis	x		x	2	2	2	Pontos		x		
Murata	x			2	2	2	Curva	x		x	
Poloni		x		2	2	2	Curvas		x		
Quagliarella		x		n	2	n	Pontos		x		
Rendon	x		x	2	2	2	Curva	x			
Rendon (2)	x		x	2	2	2	Curva	x			x
Schaffer	x		x	1	2	0	Curva	x			x
Schaffer (2)		x	x	1	2	1	Curvas		x		x
Vicini	x		x	2	2	2	Curva	x			
Viennet	x			2	3	2	Superfície	x			
Viennet (2)	x			2	3	2	Superfície	x			
Viennet (3)		x		2	3	2	Curva	x			
ZDT1				n	2	1	Curva	x			x
ZDT2				n	2	1	Curva	x		x	
ZDT3	x			n	2	1	Curva		x		x
ZDT4				n	2	2	Curva	x			

ZDT5				n	2	2	Curva	x			x
ZDT6				n	2	2			x	x	

2.3.1 Exemplos de Problemas de Teste

A seguir serão apresentados alguns dos problemas de teste mais conhecidos e utilizados, a fim de mostrar as diferentes características de cada um e o comportamento de seus Paretos.

Schaffer [51] propõe uma função bi-objetiva com conceitos MOP relevantes. A função Schaffer (1), definida por ele, tem sido utilizada para testar muitos MOEAs propostos. No entanto, por conter apenas uma variável de decisão talvez não seja apropriada para explorar as capacidades de um MOEA. As equações que definem essa função de teste são apresentadas abaixo:

$$F = (f_1(x), f_2(x)), \quad (1)$$

onde

$$f_1(x) = x^2, \quad (2)$$

$$f_2(x) = (x-2)^2, \quad (3)$$

A equação (1) indica que os resultados da função de teste *F* dependem das funções $f_1(x)$ e $f_2(x)$, as quais são descritas de acordo com as equações (2) e (3). A equação (1) não é uma representação usual de função segundo modelos matemáticos, mas na área de otimização multi-objetiva é bastante utilizada para representar dependência de uma função de teste com outras funções. O comportamento do Espaço de decisão ótimo e Pareto *Front* podem ser visualizados na Figura 5:

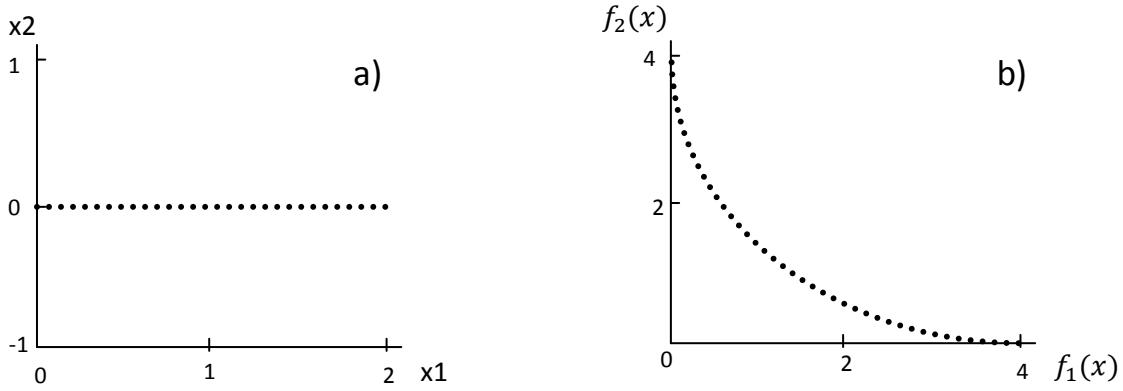

Figura 5. Função de teste Schaffer(1): a) Espaço de decisão ótimo e b) Pareto *Front*.

Na Figura 5, na situação (a), é apresentado o comportamento do espaço de decisão ótimo. O espaço de decisão é representado pelas variáveis *x1* e *x2*, que representam valores que serão mapeados para o espaço de objetivos. O espaço de objetivos é representado na Figura 5, na situação (b), e mostra os melhores resultados possíveis de se obter para os objetivos $f_1(x)$ e $f_2(x)$, utilizando-se a função de teste Schaffer (1). Em um exemplo real, tal como o de um projeto de sistema embarcado, as variáveis *x1* e *x2* poderiam ser percebidas como valores de velocidade do dispositivo e tamanho de área ocupado, enquanto os valores de $f_1(x)$ e $f_2(x)$ poderiam ser vistos como o custo e o desempenho final do sistema.

Fonseca e Fleming [15] utilizam uma função bi-objetiva, chamada Fonseca (4), que tem a vantagem de adicionar arbitrariamente variáveis de decisão sem mudar a forma do Pareto *Front* ou localização no espaço de objetivos. As equações que definem essa função de teste são apresentadas abaixo:

$$F = \left(f_1(x,y), f_2(x,y)\right), \tag{4}$$

onde

$$f_1(x,y) = 1 - \exp(-(x-1)^2 - (y+1)^2), \tag{5}$$

$$f_2(x,y) = 1 - \exp(-(x+1)^2 - (y-1)^2), \tag{6}$$

O comportamento do Espaço de decisão ótimo e do Pareto *Front* podem ser visualizados na Figura 6.

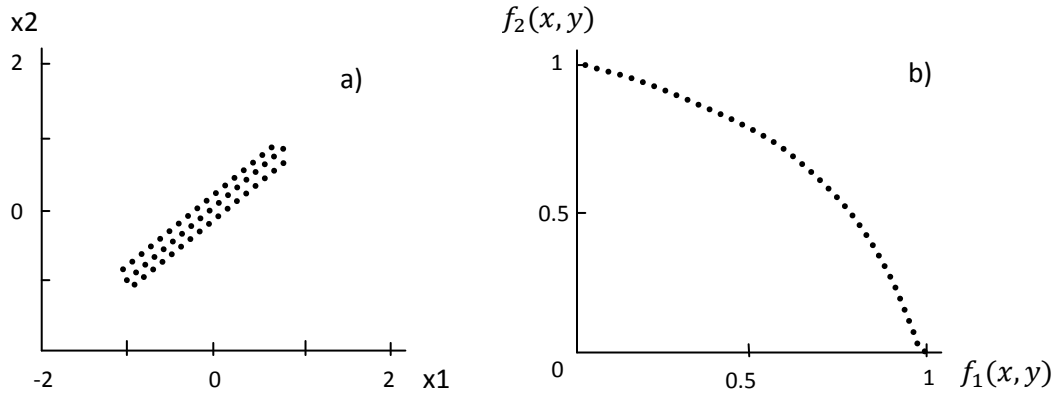

Figura 6. a) Espaço de decisão ótimo e b) Pareto *Front* da função de teste Fonseca(1).

Poloni [16] utiliza um problema de maximização. A função de teste é bi-objetiva e o Pareto é desconectado em duas curvas. O comportamento do Pareto *Front* pode ser visualizado na Figura 7.

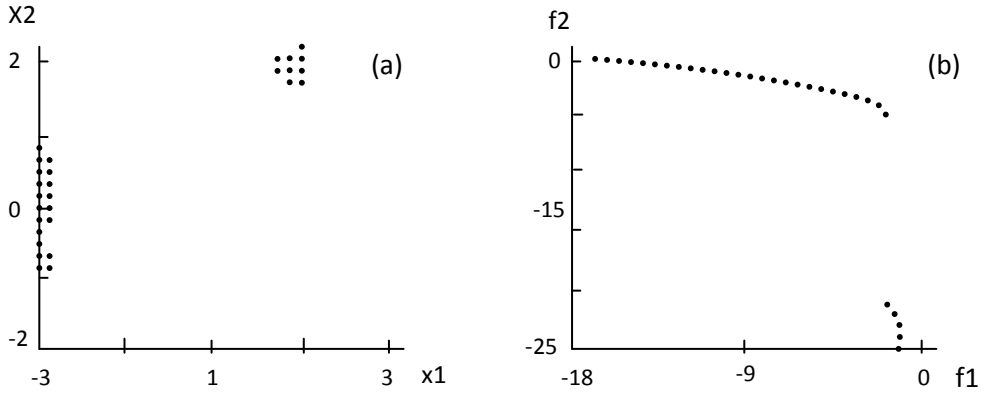

Figura 7. Função de teste Poloni: a)Espaço de decisão ótimo e b)Pareto *Front*.

No estudo de Kursawe [17] é apresentado uma função bi-objetiva cujo Pareto *Front* é desconectado em três curvas. Da mesma forma que Fonseca(1), o número de variáveis de decisão é arbitrário. As equações que definem essa função de teste são apresentadas abaixo:

$$F = \left(f_1(\vec{x}), f_2(\vec{x})\right), \tag{7}$$

onde

$$f_1(\vec{x}) = \sum_{i=1}^{n-1}\left(-10\, e^{(-0.2)*\sqrt{x_i^2+x_{i+1}^2}}\right), \tag{8}$$

$$f_2(\vec{x}) = \sum_{i=1}^{n}\left(|x_i|^{0.8} + 5sin(x_i{}^3)\right), \tag{9}$$

A função F depende de $f_1(x_1, x_2, \ldots, x_n)$ e $f_2(x_1, x_2, \ldots, x_n)$, onde x_i representa *i-ésima* variável de decisão do vetor de variáveis de decisão recebido como entrada. O comportamento do Pareto *Front* pode ser visualizado na Figura 8.

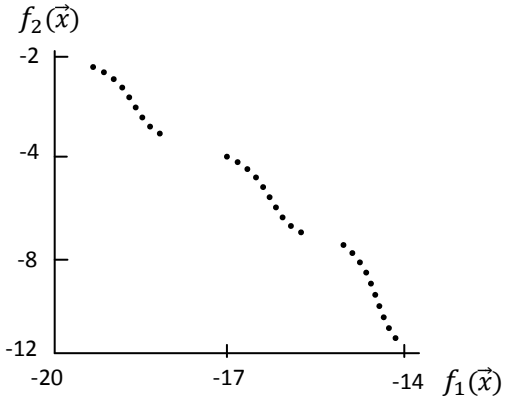

Figura 8. Pareto *Front* da função de teste Kursawe.

Em [18] são utilizadas as funções de teste descritas acima (SCH, FON E KUR), juntamente com os problemas de teste ZDT1, ZDT2, ZDT3, ZDT5 e ZDT6 [19], para validar o

algoritmo NSGAII. Os problemas de teste ZDT foram construídos utilizando a metodologia proposta por Deb [20] e que foram baseados nos principais problemas encontrados nos MOEA e descritos em [19].

As funções ZDT são definidas de acordo com a seguinte equação:

$$F = \left(f_1(\vec{x}), f_2(\vec{x})\right), \tag{10}$$

onde

$$f_1(\vec{x}) = f(x_1, \dots, x_m), \tag{11}$$

$$f_2(\vec{x}) = g(x_{m+1}, \dots, x_N)h(f(x_1, \dots, x_m), g(x_{m+1}, \dots, x_N)). \tag{12}$$

A função F é um problema de minimização e depende de $f(x_1, \dots, x_m)$, $g(x_{m+1}, \dots, x_N)$ e $h(f(x_1, \dots, x_m), g(x_{m+1}, \dots, x_N))$, as quais são descritas a seguir, de acordo com cada função de teste. Note que $f_2(\vec{x})$ depende do resultado em $f_1(\vec{x})$. A variável x_i representa a *i-ésima* variável de decisão do vetor de entrada. A variável $\vec{x}$, que representa o vetor de entrada, é apresentada de forma geral nas equações (10), (11) e (12), porém poderá ser representada através de um número de fixo de elementos do vetor, tais como x_1 ou $(x_2, \dots, x_m)$. Essa notação é utilizada nas equações seguintes e representa que o objetivo depende de um número de elementos do vetor inicial, e não necessariamente dele todo.

A função ZDT1 é definida pelas seguintes equações:

$$f_1(x_1) = x_1, \tag{13}$$

$$g(x_2, \dots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1), \tag{14}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}, \tag{15}$$

onde m=30 e $x_i \epsilon [0,1]$. O front do Pareto ótimo é convexo e é formado com $g(x) = 1$. A Figura 9 mostra o Pareto *Front* dessa função, onde os eixos $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos mapeados pela função de teste, os quais são representados nas equações (11) e (12) e detalhados nas equações (13), (14) e (15).

A função ZDT2 é definida pelas seguintes equações:

$$f_1(x_1) = x_1, \tag{16}$$

$$g(x_2, \dots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1), \tag{17}$$

$$h(f_1, g) = 1 - (f_1/g)^2, \quad (18)$$

onde *m* representa o número de variáveis de decisão e $x_i \epsilon [0,1]$. Em [19] sugere-se usar *m*=30. O front do Pareto *Front* é côncavo e é formado com $g(x) = 1$. A Figura 10 mostra o Pareto *Front* dessa função, onde os eixos $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos mapeados pela função de teste, os quais são representados nas equações (11) e (12) e detalhados nas equações (16), (17) e (18).

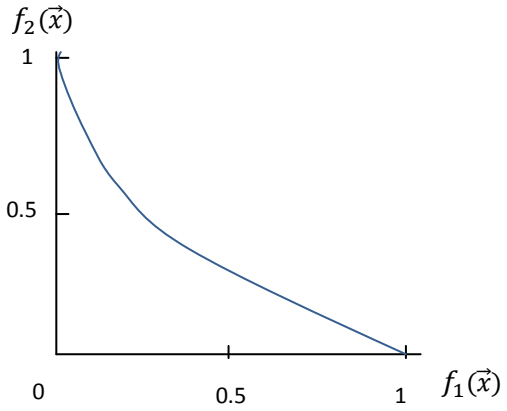

Figura 9. Pareto Front da Função de Teste ZDT1 (Convexo).

.

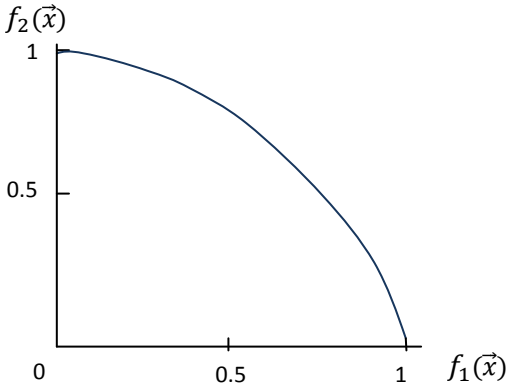

Figura 10. Pareto *Front* da função de teste ZDT2 (côncavo).

A função ZDT3 é definida pelas seguintes equações:

$$f_1(x_1) = x_1, \quad (18)$$

$$g(x_2, \dots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1), \quad (19)$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1), \quad (20)$$

onde *m* representa o número de variáveis de decisão e $x_i \epsilon [0,1]$. Em [19] é sugerido utilizar *m*=30 nas equações acima. O front do Pareto *Front* é formado com $g(x) = 1$. O Pareto dessa função é

discreto, possuindo diversas partes convexas não-contínuas. No entanto, não há descontinuidade no espaço de variáveis. O Pareto *Front* dessa função é apresentado na Figura 11, onde os eixos $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos mapeados pela função de teste, os quais são representados nas equações (11) e (12) e detalhados nas equações (18), (19) e (20).

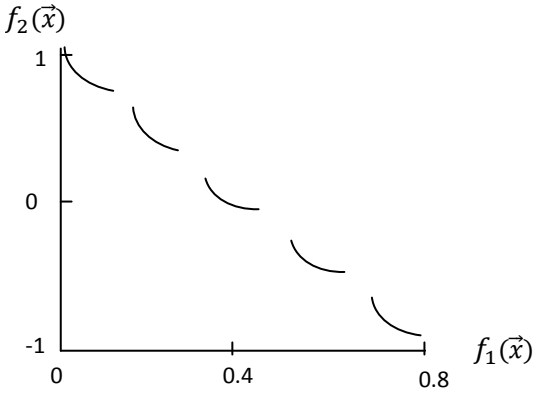

Figura 11. Pareto *Front* da função de teste ZDT3 (discreto).

A função de teste ZDT4 possui 21^9 Paretos locais, testando assim a habilidade do algoritmo de lidar com múltiplos fronts. As equações são definidas a seguir:

$$f_1(x_1) = x_1, \tag{21}$$

$$g(x_2, \dots, x_m) = 1 + 10(m-1) + \sum_{i=2}^{m}(x_i^2 - 10\cos 4\pi x_i), \tag{22}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}, \tag{23}$$

onde *m* é o número de variáveis de decisão, $x_i \epsilon [0,1]$ e $x_2, \dots, x_m \epsilon [-5,5]$. Em [19] sugere-se utilizar *m* = 10. O *front* do Pareto Ótimo global é formado com g(x) = 1. O melhor *front* de Pareto ótimo local é formado com g(x) = 1.25, utilizando a equação (22). O Pareto ótimo dessa função está representado na Figura 12, onde os eixos $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos mapeados pela função de teste, os quais são representados nas equações (11) e (12) e detalhados nas equações (21), (22) e (23).

A função de teste ZDT5 explora a característica de deceptividade, que é quando há um mínimo global e um local e as soluções do mínimo local tendem a atrair o algoritmo de busca. As equações que definem essa função de teste são definidas abaixo:

$$f_1(x_1) = 1 + u(x_1), \tag{23}$$

$$g(x_2, \dots, x_m) = \sum_{i=2}^{m} v(u(x_i)), \tag{24}$$

$$h(f_1, g) = 1/f_1, \tag{25}$$

onde x_i é um vetor binário e $u(x_i)$ representa o número de 1s presentes nesse vetor,

$$v(u(x_i)) = \begin{cases} 2 + u(x_i) \; if \;\; u(x_i) < 5 \\ 1 \qquad\qquad if \; u(x_i) = 5 \end{cases}, \tag{26}$$

$x_1 \epsilon \{0,1\}^{30}$, e $x_2, \ldots, x_m \epsilon \{0,1\}^5$, onde $\{0,1\}^n$ representa que até n bits podem estar contidos naquele vetor. Em [19] sugere-se utilizar *m*=11 e é observado que o *front* de Pareto Ótimo é formado com *g(x)*=10. A figura do Pareto *Front* pode ser visualizada na

Figura 13, onde os eixos $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos mapeados pela função de teste, os quais são representados nas equações (11) e (12) e detalhados nas equações (23), (24) e (25).

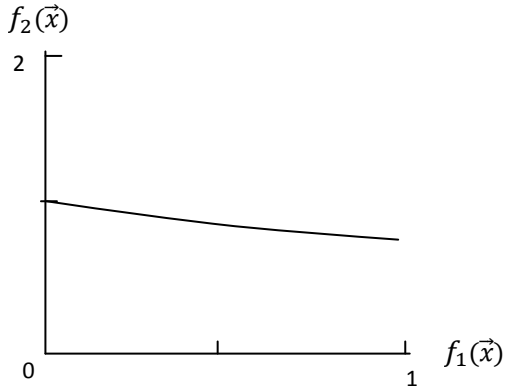

Figura 12. Pareto front da função de teste ZDT4 (multimodal).

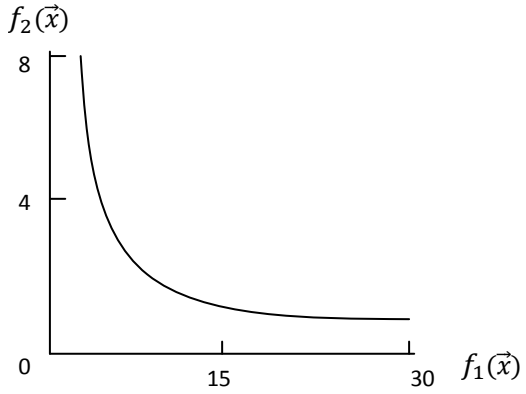

Figura 13. Pareto Front da função de teste ZDT5 (deceptivo).

A função de teste ZDT6 tenta explorar duas dificuldades encontradas nos algoritmos de busca. A primeira é que as soluções do Pareto Ótimo são distribuídas de maneira não uniforme. A segunda é que a densidade das soluções é baixa próxima ao Pareto e alta longe dele. As equações que definem essa função são apresentadas abaixo:

$$f_1(x_1) = 1 - \exp(-4x_1)\, sin^6(6\pi x_1), \tag{27}$$

$$g(x_2, \dots, x_m) = 1 + 9 \cdot \left(\sum_{i=2}^{m} x_i/(m-1)\right)^{0,25}, \quad (28)$$

$$h(f_1, g) = 1 - (f_1/g)^2, \quad (29)$$

onde x_i representa representa o *i-ésimo* termo vetor de variáveis de decisão recebido como parâmetro e $x_i \epsilon$ [0,1]. Em [19] é observado que o front do Pareto ótimo é formado com *g(x) = 1* e sugere-se utilizar *m*=10. O *front* do Pareto Ótimo é côncavo e pode visualizado na Figura 14, onde os eixos $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos mapeados pela função de teste, os quais são representados nas equações (11) e (12) e detalhados nas equações (27), (28) e (29).

.

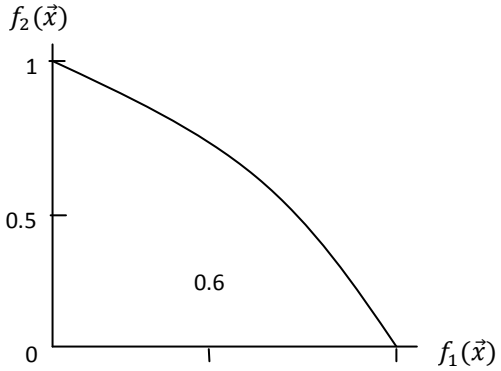

Figura 14. Pareto *Front* da função de teste ZDT6 (não uniforme).
1

2.4 Métricas de Avaliação de Desempenho

Métricas são bastante utilizadas a fim de mensurar características de MOEAs, ajudando a entender seu comportamento no domínio do problema e permitindo uma avaliação mais concreta do desempenho do algoritmo. As métricas também são um importante parâmetro de comparação entre algoritmos, uma vez que muitas vezes é difícil perceber qual algoritmo apresenta um melhor conjunto de soluções para o problema. A Figura 15 ilustra essa característica.

Conforme mostra a Figura 15, na situação (a) é possível concluir que o conjunto de soluções do algoritmo A é mais apropriado do que o do algoritmo B, uma vez que para cada solução encontrada em B existe pelo menos uma solução em A que seja melhor ou igual à de B em todos os objetivos. Mas ainda assim é difícil dizer o quanto o algoritmo A é melhor que B.

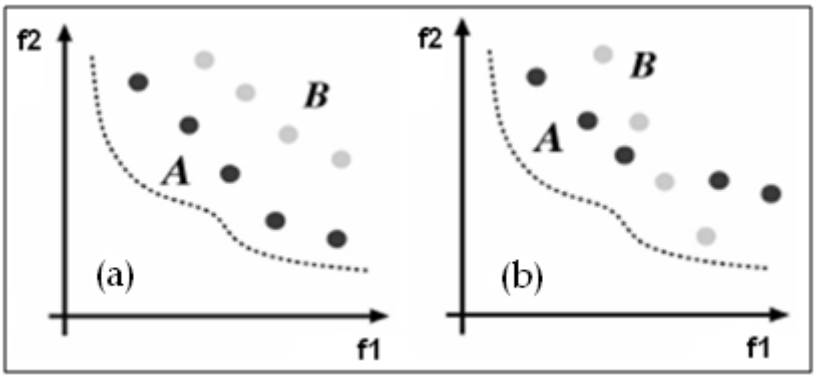

Figura 15. Comparação entre algoritmos: (a) Possível concluir qual o melhor algoritmo e (b) Dificuldade em se encontrar qual o melhor algoritmo para o problema.

Na situação (b) já não se pode assegurar qual algoritmo é melhor, pois no conjunto das melhores soluções encontradas na união dos resultados de A e B, algumas soluções pertencem a A e outras a B. Dessa forma, as melhores soluções que pertencem ao algoritmo A podem apresentar características que sejam boas para um tipo de aplicação, enquanto que as de B podem ser boas apenas para outros tipos de aplicações. Sendo assim, a asserção de que os resultados de um algoritmo são melhores do que de outro algoritmo dependerá de quais características são importantes para o usuário. Fonseca *et al.* [21] também tira essa mesma conclusão e observa que métricas podem ser combinadas a fim de se buscar um conjunto de características mais apropriadas para análise comparativa entre os algoritmos.

As métricas podem ser classificadas em unitárias e binárias. Métricas unitárias são aquelas que atribuem um valor que reflete uma característica junto a um conjunto de soluções. É comum encontrar a utilização dessas métricas de forma combinada [22][5]. As métricas binárias, por sua vez, atribuem valores aos pares de conjuntos de soluções. Algumas das métricas mais utilizadas são apresentadas a seguir.

2.3.1 Distância da Geração

A distância da geração mede a distância Euclidiana entre as soluções do Pareto *Front* conhecido e o Pareto Ótimo. Conforme ilustra a

Figura 16, essa métrica é calculada através do somatório da distância Euclidiana entre cada solução obtida e o Pareto Ótimo. Na mesma figura, a distância Euclidiana é representada pelo menor caminho entre a solução obtida no Pareto *Front* e a solução mais próxima do Pareto Ótimo, o que torna a distância Euclidiana perpendicular à curva do Pareto Ótimo. A distância da geração foi proposta por Van Veldhuizen e Lamont [23] e é calculada pela equação abaixo:

$$GD = \frac{\sqrt{\sum_{i=1}^{S} d_i^2}}{s}, \tag{30}$$

onde s é o número de soluções não-dominadas no Pareto *Front* e d_i é a menor distância Euclidiana entre a solução i e o Pareto Ótimo.

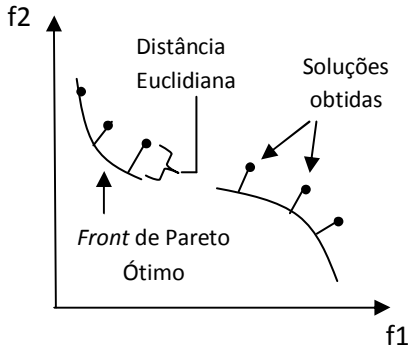

Figura 16. Métrica de Distância da Geração: baseada na distância euclidiana das soluções obtidas.

2.3.2 Diversidade

A métrica de diversidade mede o grau de cobertura entre as soluções obtidas e as soluções de extremidade. Esta medida reflete a boa separação das soluções e, por conseqüência, uma boa representação do Pareto *Front*. A Figura 17 ilustra a distribuição e diversidade das soluções encontradas.

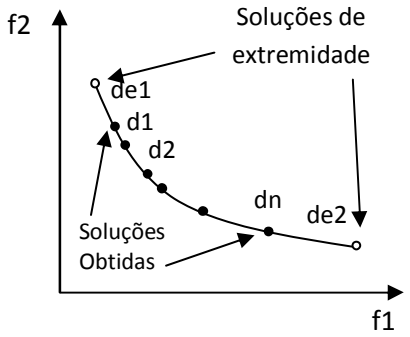

Figura 17. Cálculo da métrica de diversidade: baseado na distância entre as soluções.

Essa métrica foi proposta por Deb *et al.*[24] e é calculada pela equação abaixo:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{s-1} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + (s-1)\bar{d}}, \tag{31}$$

$$\bar{d} = \frac{\sum_{i=1}^{s-1} d_i}{s-1}, \tag{32}$$

onde d_i é a distância entre soluções vizinhas e *s* o número de soluções no Pareto-Front. Desta forma, d_i pode ser representada como a distância Euclidiana entre a *i-ésima* e *(i+1)-ésima* soluções. O termo $\bar{d}$ representa o valor médio de todas as distâncias d_i. O parâmetro d_m^e representa a distância entre a solução extrema do Pareto *Front* e a solução correspondente no *m-ésimo* objetivo, onde *m* corresponde ao número de objetivos. Para uma distribuição ideal de soluções a variável Δ é igual a zero.

2.3.1.3 Espalhamento Máximo (MS)

A métrica de espalhamento máximo [25], conforme o próprio nome sugere, indica o grau de espalhamento máximo do conjunto de soluções. Isso indica que as extremidades foram alcançadas com uma determinada distância Euclidiana entre eles. Desta forma, é desejável que a distância entre as extremidades sejam a maior possível, pois isso representa um conjunto maior para se encontrar as soluções ótimas. A Figura 18 ilustra a distância entre as extremidades.

A equação é apresentada abaixo:

$$MS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\min(f_i^{max},F_i^{max})-\max(f_i^{min},F_i^{min})}{F_i^{max}-F_i^{min}}\right\}^2}, \qquad (33)$$

onde f_i^{max} e f_i^{min} são os valores máximo e mínimo, respectivamente, do *i*-ésimo objetivo encontrado nas soluções do Pareto atingido, enquanto F_i^{max} e F_i^{min} representam os valores máximo e mínimo do *i*-ésimo objetivo no conjunto do Pareto Ótimo. As funções $\min(f,F)$ e $\max(f,F)$ retornam, respectivamente, os valores mínimos e máximos quando comparados os parâmetros *f* e *F*.

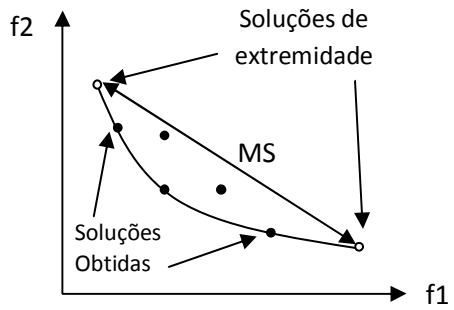

Figura 18. Distância entre as soluções de extremidade.

2.3.1.4 Taxa de Erro

Essa métrica indica a porcentagem de soluções que não estão no Pareto Ótimo. Ela foi proposta por Van Veldhuizen [10] e é calculado através da seguinte equação:

$$ER = \frac{\sum_{i=1}^{s} e_i}{s}, \tag{45}$$

onde *s* é o número de soluções que não estão no Pareto ótimo P e e_i =1 se $i \notin P$ e e_i = 0, caso contrário. O valor ER = 0 quer dizer que todas as variáveis fazem parte do Pareto ótimo.

2.3.1.5 Hipervolume

O indicador de hipervolume [26] calcula o volume da região coberta entre os pontos das soluções do *front* encontrado e um ponto de referência. Matematicamente, para cada solução *i* pertencente ao Pareto *Front* Q, é construído um hipercubo *vi* com referência à um ponto W . O ponto de referência pode ser encontrado construindo-se um vetor com os piores valores de função objetivo. O resultado da métrica é a união de todos os hipercubos encontrados. Nessa métrica, quanto maior o valor do hipervolume melhor, pois um alto valor de hipervolume indica que houve um elevado espalhamento entre as soluções extremas do Pareto encontrado e indica também que houve uma maior convergência, pois quanto mais o algoritmo convergir maior será o volume calculado nessa métrica. A Figura 19 ilustra o cálculo de hipervolume.

Apesar de ser conceitualmente intuitiva, essa métrica apresenta um elevado custo computacional. Essa métrica é calculada através de

$$HV = volume\left(U_{i=1}^{|Q|} v_i\right), \tag{46}$$

onde *i* representa a solução do Pareto *Front Q* e *vi* representa o hipercubo daquela solução.

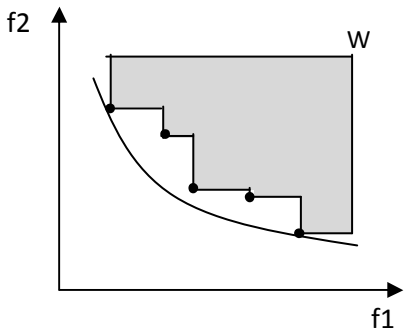

Figura 19. Hipervolume gerado pelas soluções do Pareto *Front*.

2.3.1.6 Espaçamento (SS)

Schott [27] apresentou uma métrica de espaçamento que mensura as distribuições das soluções no Pareto *Front*. Essa métrica calcula a distância relativa entre soluções consecutivas no Pareto *Front*. A equação é descrita a seguir:

$$SS = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - \bar{d})^2}\,, \tag{47}$$

onde $|Q|$ é o número de soluções do Pareto *Front*, $d_i = min_{k \in Q \wedge k \neq i} \sum_{m=1}^{M} |f_m^i - f_m^k|$ e $\bar{d}$ é a média de todos os d_i, representado por $\bar{d} = \sum_{i=1}^{|Q|} d_i / |Q|$. O parâmetro m representa o número de objetivos e f_m^i representa o valor do objetivo m para a solução i. A métrica sendo igual a zero indica que todos os membros do Pareto *Front* estão eqüidistantes.

2.3.1.7 Escala de Progresso (RP)

A escala de progresso é utilizada para medir a velocidade de convergência do algoritmo e foi definida para algoritmos de um único objetivo em [28]. Em [10] é apresentada uma adaptação da equação para lidar com algoritmos multiobjetivos. Essa métrica é calculada através de

$$RP \triangleq ln\sqrt{\frac{G_1}{G_T}}\,, \tag{48}$$

onde G_1é a Distância de Geração da geração 1 e G_T a distância na geração T.

2.3.1.8 Cobertura

A métrica de cobertura [26] é capaz de mensurar quanto um determinado conjunto de soluções domina outro conjunto. Essa relação é exemplificada na Figura 20.

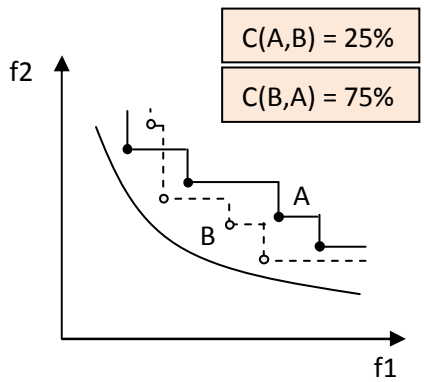

Figura 20. Métrica de cobertura entre as soluções do conjunto *A* e *B*.

O valor C(A,B) na Figura 20 indica a porcentagem de elementos em B que são fracamente dominados por elementos de A. Essa métrica possui um baixo custo computacional comparado com o hipervolume. A equação que representa a métrica de cobertura é calculada através de

$$C(A,B) := \frac{|\{b \in B; \exists a \in A: a \preccurlyeq b\}|}{|B|}, \tag{49}$$

onde b e a representam soluções do conjuntos de solução B e A, respectivamente. O termo $a \preccurlyeq b$ indica que b é fracamente dominado pela solução a. O valor da métrica $C(A,B)=1$ significa que todos os membros de B são fracamente dominados por A.

Capítulo 3

MOEAs Utilizados

Neste capítulo é feita uma descrição dos MOEAs utilizados na implementação da ferramenta desenvolvida. Foram selecionados os algoritmos evolucionários NSGA-II, SPEA2, PAES e PESA-II por estarem entre os algoritmos multi-objetivos mais utilizados. Cada algoritmo é descrito a seguir.

3.1 NSGA-II

O algoritmo NSGA, proposto em [29], foi um dos primeiros EAs desenvolvidos. Porém, ao longo dos anos foi alvo das seguintes críticas: alta complexidade computacional do algoritmo de ordenação baseado em não-dominância, ausência de elitismo e necessidade de especificar o parâmetro σ_{share}. Posteriormente, foi proposto o algoritmo NSGAII [5], que apresentava melhoras em relação ao anterior, implementando elitismo, reduzindo complexidade computacional e retirando a necessidade de especificar o parâmetro σ_{share}, que era um parâmetro utilizado para verificar a diversidade de soluções equivalentes.

O NSGAII baseia-se no conceito de não-dominância para selecionar indivíduos, assim como na métrica *crowding distance*, que é utilizada para selecionar os melhores indivíduos de um conjunto de soluções não-dominadas. O método de cálculo do *crowding distance* é mostrado na Tabela 2.

Um esquema do processo de ordenação e seleção utilizado no NSGAII é mostrado na Figura 21. De acordo com essa figura, inicialmente tem-se um conjunto de soluções formado por P_t (gerado através de operadores de seleção, mutação e cruzamento) e Q_t(população inicial da iteração *t)*. Em seguida é feita a ordenação das soluções de acordo com o grau de não-dominância. Com isso, as soluções são divididas em *fronts*, onde aquelas contidas nos melhores *fronts* (com menos soluções dominadas) são selecionadas para a próxima iteração.

O algoritmo presente na Tabela 3 representa o pseudo-código do NSGA-II

Tabela 2. Cálculo do *crowding distance.*

Passo 1: Ranquear a população e identificar os fronts não-dominados $F_1, F_2, .. F_R$.

Para cada j = 1,..., R, repetir os passos 2 e 3.

Passo 2 : Para cada função objetivo k, ordene as funções em F_j em ordem crescente.

Seja $L = |F_j|$ e $x_{[i,k]}$ a representação da i-ésima solução na lista ordenada, com respeito ao objetivo k. Atribua $cd_k(x_{[1,k]}) = \infty$ e $cd_k(x_{[L,k]}) = \infty$.

Para i=1,2,...L-1 atribua $cd_k(x_{[i,k]}) = \frac{z_k(x_{[i+1,k]}) - z_k(x_{[i-1,k]})}{z_k^{max} - z_k^{\min}}$

Passo 3 : Para encontrar o *crowding distance* total *cd(x)* de uma solução x, some os *crowding distances* das soluções com respeito a cada objetivo.

$cd(x) = \sum_k cd_k(x)$

Tabela 3. Pseudo-código do NSGA-II.

1. Gera população aleatória inicial $P0$ de tamanho N
2. Ordena $P0$ com base em não-dominância
3. População Q_0 de tamanho N é obtida a partir de $P0$ através da aplicação dos operadores de seleção, cruzamento e mutação.
4. **Repetir G-1** vezes, onde G é o número de gerações
5. $R_t = P_t \; U \; Q_t$, onde t é a iteração atual
6. F = ordenar R_t por não-dominância
7. F = (F_1, F_2, F_3, ...), todos os fronts de R_t
8. **enquanto** $|P_{t+1}| < N$
9. atribuir *crowding distance* para cada solução em F_i
10. $P_{t+1} = P_{t+1} \; U \; F_i$
11. ordenar P_{t+1} com base em não dominância e *crowding distance* (*crowded operator)*
12. $P_{t+1} = P_{t+1}[0:N]$
13. População Q_{t+1} de tamanho N é obtida a partir da aplicação dos operadores de seleção (baseado no *crowded operator*), cruzamento e mutação sobre a população P_{t+1}
14. t = t+1

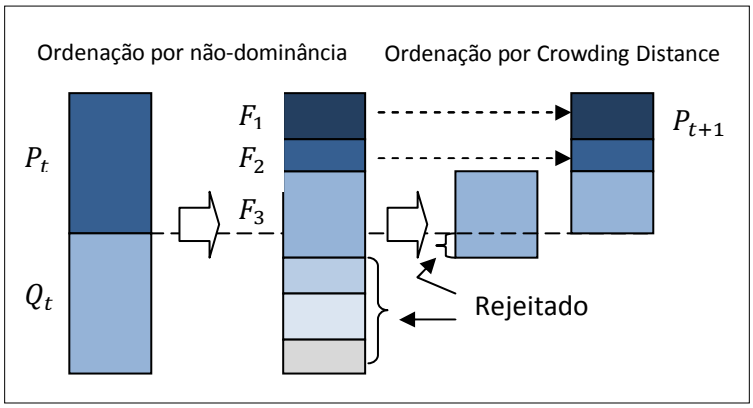

Figura 21. Esquema do procedimento do NSGAII.

3.2 SPEA2

O SPEA2 [6] é um algoritmo que utiliza um arquivo externo para armazenar as melhores soluções encontradas entre as gerações. Ele se baseia no conceito de dominância e densidade de vizinhos para avaliar o *fitness,* que é uma medida de qualidade da solução encontrada. Pela densidade de vizinhos é possível observar quais indivíduos são mais representativos para o conjunto de solução final. Desta forma, a densidade é utilizada como critério de decisão entre indivíduos com mesmo grau de dominância. A densidade é calculada através de

$$D(i) = \frac{1}{\sigma_i^k + 2}, \tag{50}$$

onde

$$k = \sqrt{P_t + \bar{P}_t}, \tag{51}$$

para i igual à *i-ésima* solução para a qual se está calculando a densidade, P_t igual ao tamanho da população inicial da iteração t, $\bar{P}_t$ igual ao tamanho do arquivo externo de soluções não dominadas e σ_i^k igual à distância Euclidiana entre a solução i e a *k-ésima* solução mais próxima. O pseudo-código do SPEA2 é apresentado na tabela abaixo.

Tabela 4. Pseudo-código do SPEA2.

1: Entrada : N(Tamanho da população)
2: $\bar{N}$ (Tamanho do arquivo)
3: T (Número máximo de gerações)
4: Saída : A (Soluções não dominadas)
5 : **Inicialização** :Gere população inicial P_0 e cria um arquivo externo vazio $\bar{P}_0 = \emptyset$.
6 : Seta $t = 0$.

7 : **Atribuição de *fitness*** :Calcule o *fitness* de P_t e $\bar{P}_t$.

8: **Seleção** : Copie os indivíduos não dominados de P_t e $\bar{P}_t$ para $\bar{P}_{t+1}$.

9: Se $|\bar{P}_{t+1}| = \bar{N}$ então

10: Teste condição de parada.

11: Senão se $|\bar{P}_{t+1}| > \bar{N}$ então

12: Aplique o truncamento a $\bar{P}_{t+1}$

13: Senão

14: Complete o arquivo com elementos dominados de P_t e $\bar{P}_t$

15: **Condição de Parada**: se $t >= T$ ou outro critério for satisfeito, atribua $A = \bar{P}_{t+1}$ e pare.

16: **Cruzamento** : Seleção por torneio direto (binário) em $\bar{P}_{t+1}$. Aplique *crossover* em $\bar{P}_{t+1}$ até completar o *mating pool.*

17: **Diversidade** : Aplique o operador de mutação ao *mating pool* e sete a população resultante em P_{t+1}. Incremente o contador de geração e volte para a linha 7.

Como pode ser visto na Tabela 4, inicialmente é gerada uma população inicial P_0 e um arquivo externo vazio $\bar{P}_0$. Em seguida é calculada a função *fitness*, onde é possível ter uma percepção melhor do relacionamento entre as soluções envolvidas. Posteriormente, os indivíduos não dominados do conjunto de soluções são copiados para compor o arquivo externo da próxima iteração. Baseado no tamanho do arquivo externo, pode haver, nessa etapa, a necessidade de truncamento ou de completar o novo arquivo externo com elementos dominados. Caso a condição de parada não seja atingida é realizado seleção, cruzamento e mutação. Desta forma, é gerada uma nova população e esse processo se repete até que a condição de parada seja atingida.

3.3 PAES

O algoritmo PAES, inicialmente proposto por Knowles e Corne [4], surgiu da tentativa de resolver problemas de roteamento de redes de computadores. Eles observaram que uma busca local, em problemas mono-objetivos, apresentava um desempenho melhor do que abordagens baseadas em população. Baseado nisso, foi desenvolvido o algoritmo evolucionário de otimização muti-objetiva, utilizando busca local. O código principal do algoritmo é mostrado na Tabela 5.

Tabela 5. Pseudo-Código para o PAES.

1. Gera solução aleatória *c* e adiciona ao arquivo
2. Gera mutação *m* a partir de *c* e avalia *m*
3. **Se** *c* domina *m* **então**
4. Descarta *m*
5. **Senão se** *m* domina *c* **então**
6. Substitui *c* por *m* e adiciona *m* ao arquivo
7. **Senão se** *m* é dominado por algum membro do arquivo **então**
8. Descarta *m*
9. **Senão** aplique *teste(c, m,* arquivo) para determinar qual será a nova solução e se *m* deverá ser adicionado ao arquivo.

O código da função *teste* é mostrado na Tabela 6. Essa função baseia-se na densidade da região onde se encontra a solução para determinar se ela fará parte da próxima iteração.

Tabela 6. Pseudo-código para teste(*c*, *m*, *arquivo)* usada no PAES.

1. **Se** o arquivo não está cheio **então**
2. Adiciona *m* ao arquivo
3. **Se** *m* está numa região menos densa que *c* **então**
4. Aceita *m* como a nova solução atual
5. **Senão** mantém *c* como a solução atual
6. **Senão**
7. **Se** *m* está numa região menos densa que algum membro do arquivo **então**
8. Adiciona *m* ao arquivo
9. Remove o membro que estiver na região mais densa do arquivo
10. **Se** *m* está numa região menos densa do arquivo que *c* **então**
11. Aceita *m* como a nova solução atual
12. **Senão** mantém *c* como solução atual
13. **Senão se** *m* está numa região menos densa que *c* **então**
14. Aceita *m* como a nova solução atual
15. **Senão** mantém *c* como a solução atual

De acordo com a Tabela 5, inicialmente é gerada uma solução aleatória *c* e uma mutação *m* a partir dessa solução. Caso *c* domine m então m é descartado. Caso *m* domine *c* então *m* é

adicionado ao arquivo externo e o processo se repete. Caso *m* não seja dominado por nenhum membro do arquivo externo, então é utilizada uma função *teste* para verificar a densidade das regiões onde se encontram as soluções e poder decidir qual a melhor solução para fazer parte do conjunto de soluções final. O processo se repete até que o critério de parada seja atingido.

3.4 PESA-II

O algoritmo PESA-II foi criado por Knowles *et al*¸ em [7] e é baseado no algoritmo PESA [30]. A principal diferença entre o PESA e o PESA-II é que o último apresenta uma nova estratégia de seleção baseada em hiper grades (*hypergrid)*. As hiper grades consistem na divisão do espaço de busca em células com intervalos proporcionais à cada dimensão, formando assim hiper cubos. A análise dessas regiões permite a verificação da densidade das soluções, auxiliando no processo de seleção e atualização do arquivo externo utilizado. O código do algoritmo é mostrado na Tabela 7.

Tabela 7. Pseudo-código do PESA-II.

1. Inicializa a População Inicial *PI* aleatoriamente com *N* indivíduos.
2. Move os indivíduos não-dominados para o arquivo externo *PE*.
3. **Se** Critério de Parada foi atingido
4. Pára e retorna *PE* como o Pareto.
5. **Senão**
6. Apaga a PI.
7. **Repetir** até serem criados novos *N* indivíduos em *PI*.
8. Selecionar 2 indivíduos de *PE*.
9. Usando-se hiper cubos, cruzar e produzir um único filho.
10. Muta-se o filho criado.
11. Selecionar 1 indivíduo e produzir um novo filho.
12. **Retornar** para o passo 2.

De acordo com o pseudo-código da Tabela 7, inicialmente são gerados aleatoriamente *N* indivíduos, que compõem a população inicial. Desses indivíduos, aqueles que não são dominados são movidos para o arquivo externo *PE*. Caso o critério de parada não seja atingido é criado uma nova população de tamanho *N* a partir da seleção dos indivíduos de *PE*, utilizando-se como critérios de seleção a densidade dos hiper cubos. As soluções que encontram-se em regiões

menos densas tem maior probabilidade de serem escolhidas. Esse processo se repete até que o critério de parada seja atingido.

Capítulo 4

Descrição da Ferramenta

A ferramenta proposta tem como objetivo fornecer mecanismos de apoio ao pesquisador para a simulação e análise de MOEAs. Para o desenvolvimento da ferramenta foi necessário considerar algumas características importantes, como modularidade, interface amigável e possibilidade de futuras extensões do código desenvolvido.

Foi utilizada a linguagem de programação *Java* [31] no desenvolvimento do simulador, o qual foi construído com o auxílio das ferramentas *Eclipse* [32] e *Netbeans* [33]. Para a ilustração dos gráficos dos algoritmos simulados utilizou-se a biblioteca gráfica *JFreeChart* [34], que contém funcionalidades de auxílio à apresentação de gráficos em Java.

As seções seguintes descrevem cada módulo da arquitetura desenvolvida e a estruturação da interface com o usuário.

4.1 Módulos da Ferramenta

A arquitetura utilizada para o desenvolvimento da ferramenta proposta é constituída de diversos módulos que interagem entre si para a realização conjunta das funcionalidades do sistema. Cada módulo tem sua característica específica, que agrupa um conjunto de classes desenvolvidas para realizar as atividades daquele módulo. Na Figura 22 estão ilustrados os principais módulos utilizados no sistema e como eles interagem entre si.

A estruturação dos módulos também é dividida em camadas, como ilustra a Figura 22. O módulo de Interface Gráfica interage diretamente com o módulo de Negócios, que por sua vez pode ter acesso aos módulos de Técnicas, Métricas e Funções de Teste. Esses, por sua vez têm acesso aos módulos de tratamento de exceções, classes básicas e classes de apoio ao sistema. Cada um desses módulos será descrito nas seções seguintes. É importante observar que a arquitetura em camadas foi estruturada a fim de facilitar reutilização do código e eventuais

alterações futuras. Na arquitetura utilizada o nível de acesso entre as camadas é realizado sempre de cima para baixo.

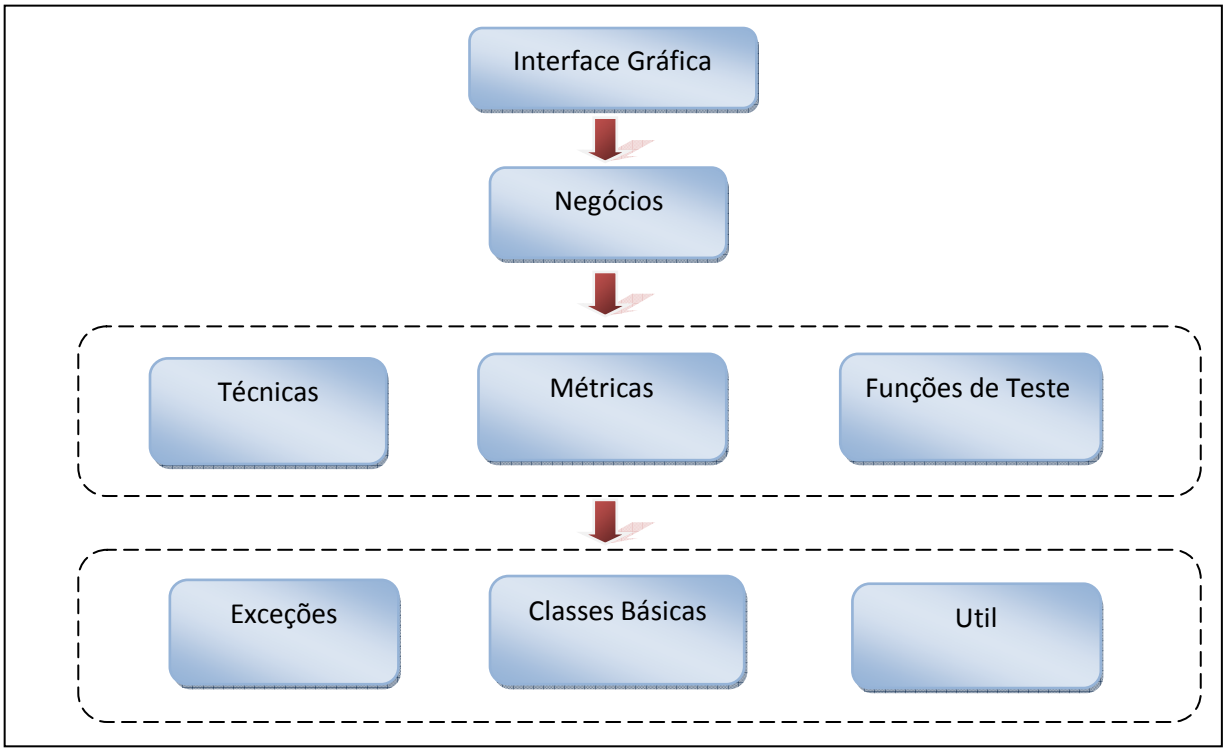

Figura 22. Representação da interação entre os módulos da ferramenta.

4.1.1 Módulo de Classes Básicas

O módulo de classes básicas contém todas as classes primárias necessárias para o funcionamento do sistema. Muitos dos conceitos envolvidos na aplicação de MOEAs estão descritos nas classes desse módulo. Dentre eles, podem-se encontrar os conceitos de gene, cromossomo, população, *Front*, objetivo e operadores de algoritmos genéticos. Cada conceito foi modelado de forma a se construir classes que representem objetos do sistema.

4.1.2 Módulo Util

O módulo Util é responsável por agregar classes que manipulam dados comuns a grande parte do sistema. Em sua maioria, as classes contidas nesse módulo não precisam ser instanciadas para a utilização de seus métodos e apresentam um fraco vínculo com o restante do sistema. A maior funcionalidade desse módulo é oferecer suporte à cálculos comuns a diversas partes do sistema, como cálculo de Distância Euclidiana, ordenação e manipulação de dados das soluções encontradas.

4.1.3 Módulo de Exceções

O módulo de exceções contém o conjunto de exceções que podem ser emitidas pelo sistema. É um módulo importante na construção de um projeto de *software*, pois permite um detalhamento melhor dos erros que possam vir a ocorrer, assim como auxilia na detecção de falhas no sistema. Dentre exceções que podem ocorrer estão: parâmetro inválido para a função de teste, função de teste incompatível com o número de variáveis de decisão, entre outros. Futuras exceções que venham a ser detectadas serão agregadas a esse módulo.

4.1.4 Módulo de Funções de Teste

O módulo de funções de teste realiza a implementação de todas as funções de teste utilizadas na ferramenta. A lista de funções de teste utilizadas encontra-se na seção de Apêndice deste documento, onde são detalhas as equações de cada função, assim como é ilustrado o Pareto Ótimo de cada uma delas.

Inicialmente foi definido uma interface, que conteria todos os métodos implementados por cada uma das classes de função de teste. Assim, cada classe nesse módulo, que representa uma função de teste, implementa em sua estrutura os métodos para calcular os valores dos objetivos dado um conjunto de variáveis de decisão, gerar números aleatórios que satisfaçam suas restrições, assim como um método que verifica essas restrições.

No total foram implementadas mais de 30 funções de teste, cada uma com características específicas, o que possibilita uma análise dos diferentes graus de habilidades inerentes aos MOEAs estudados. Conforme observado no capítulo 2 desse documento, as funções de teste são importantes para avaliação e validação das técnicas analisadas.

4.1.5 Módulo de Métricas

O módulo de métricas contém métodos que implementam as métricas descritas no Capítulo 2 deste documento. Esse módulo utiliza alguns dos métodos desenvolvidos no módulo Util para a realização dos cálculos das métricas.

4.1.6 Módulo de Técnicas

Esse módulo contém as técnicas de MOEAs utilizadas na ferramenta proposta, que são NSGAII, PAES, PESA2 e SPEA2. As técnicas utilizadas, por apresentarem muitos conceitos relacionados aos algoritmos genéticos, utilizam muitas das estruturas desenvolvidas no módulo de Classes Básicas, onde são descritos os métodos de mutação, *crossover* e seleção. Por se tratar de

utilização de classes já existentes, as técnicas desse módulo diferenciam-se apenas em critérios de seleção, ordem de procedimentos e componentes utilizados. Dessa forma, a adição de novas técnicas poderá ser feita de forma simples e sua maioria, apenas reutilizando muitos dos conceitos já implementados. O procedimento de cada técnica foi codificado utilizando os pseudo-códigos apresentados no capítulo 3 deste documento.

4.1.7 Módulo de Negócios

O módulo de negócios é responsável por realizar a comunicação entre o módulo GUI e os módulos de Técnicas, Métricas e Funções de Teste. Toda requisição de processamento do usuário passa por esse módulo, que é então repassado para os módulos das camadas abaixo. O módulo de negócios facilita a portabilidade do sistema para diferentes tipos de interface.

4.1.8 Módulo de Interface Gráfica

O módulo de Interface Gráfica é responsável por oferecer a interface do sistema com o usuário. É nele que estão contidas as classes que realizam a construção das janelas gráficas, menus e exibição dos resultados processados nas camadas mais baixas do sistema. Esse módulo também contém as classes que utilizam a biblioteca *JFreeChart*, que são responsáveis pela apresentação dos resultados em um gráfico. Todas as telas desenvolvidas nesse módulo são detalhadas na seção seguinte.

4.2 Funcionamento

A ferramenta proposta foi desenvolvida para utilização de forma intuitiva, permitindo ao usuário a configuração dos parâmetros de simulação de forma simples e precisa. Inicialmente, é apresentado ao usuário um conjunto de opções de customização, que servirão como entradas para o processo de simulação. Após a simulação, o usuário poderá analisar os resultados obtidos de acordo com as configurações solicitadas. A figura abaixo mostra o processo de utilização da ferramenta, desde a escolha dos parâmetros, até a análise dos resultados.

Conforme apresentado, a seleção de parâmetros de simulação envolve escolha do algoritmo, métricas e funções de teste. A configuração desses parâmetros pode ser realizada em qualquer ordem e inicialmente são apresentados valores padrões. A Figura 24 ilustra a tela de seleção do algoritmo.

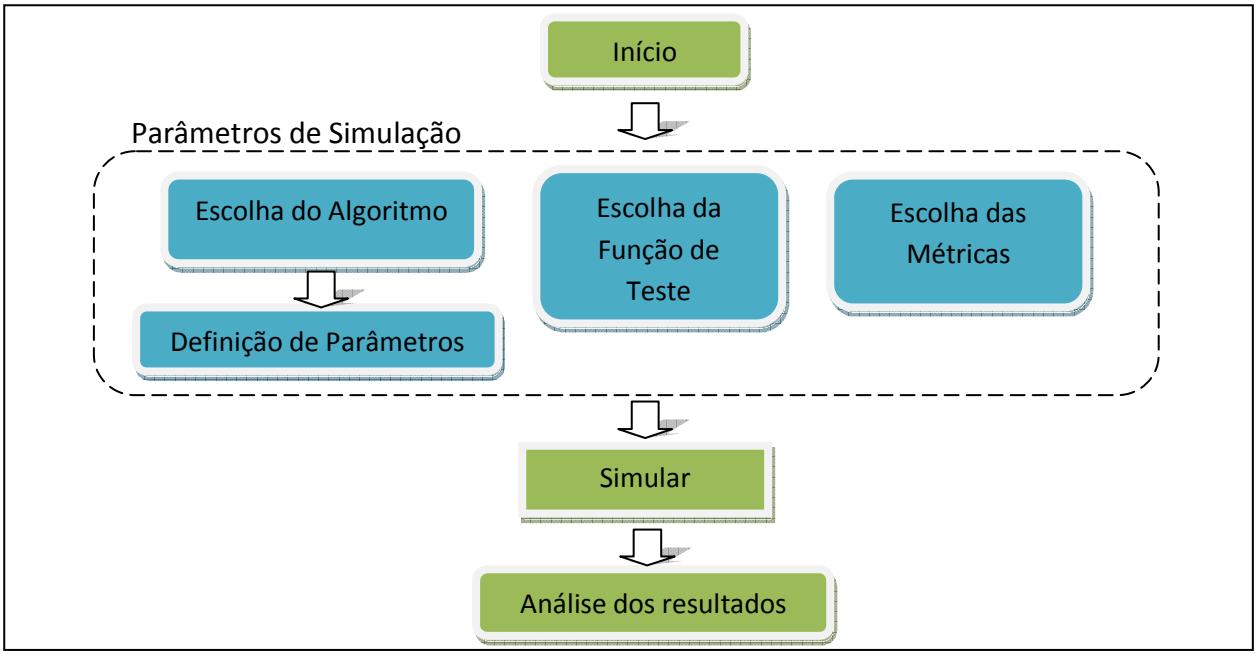

Figura 23. Fluxo de utilização da ferramenta proposta.

No quadro "Adicionar Algoritmo", é possível escolher um nome que representará a simulação daquele algoritmo, assim como o algoritmo base dele. A escolha do algoritmo base permite a seleção entre os quatro algoritmos implementados: NSGAII, PAES, PESA-II e SPEA2. Podem ser adicionados quantos algoritmos o usuário quiser, sendo exibidos os algoritmos adicionados no quadro "Algoritmos Adicionados". Ao clicar no nome de um dos algoritmos na lista de algoritmos selecionados, é mostrado abaixo, no quadro "Parâmetros", os parâmetros referentes àquele algoritmo. As opções de parâmetros variam de acordo com o algoritmo base. Inicialmente, cada algoritmo é iniciado com valores pré-definidos para todos os parâmetros, mas podem ser customizados pelo usuário.

A tela de escolha de funções de teste permite escolher uma função de teste, dentre o conjunto de funções implementadas, para servir como aplicação dos algoritmos selecionados. A Figura 25 mostra essa tela.

Como pode ser observado, é possível escolher uma dentre a lista de funções de teste apresentadas no quadro esquerdo. Ao selecionar alguma função de teste são apresentadas informações sobre ela no quadro "Descrição", no lado direito da tela. Esse quadro contém informações como número de objetivos, número de variáveis de decisão, autor e se é maximização ou minimização.

A tela de escolha de métricas permite a seleção das métricas que serão calculadas para cada algoritmo simulado. Nessa tela é possível gerenciar quais métricas farão parte dos resultados

finais, assim como é possível observar a descrição de cada uma delas. A Figura 26 mostra essa tela.

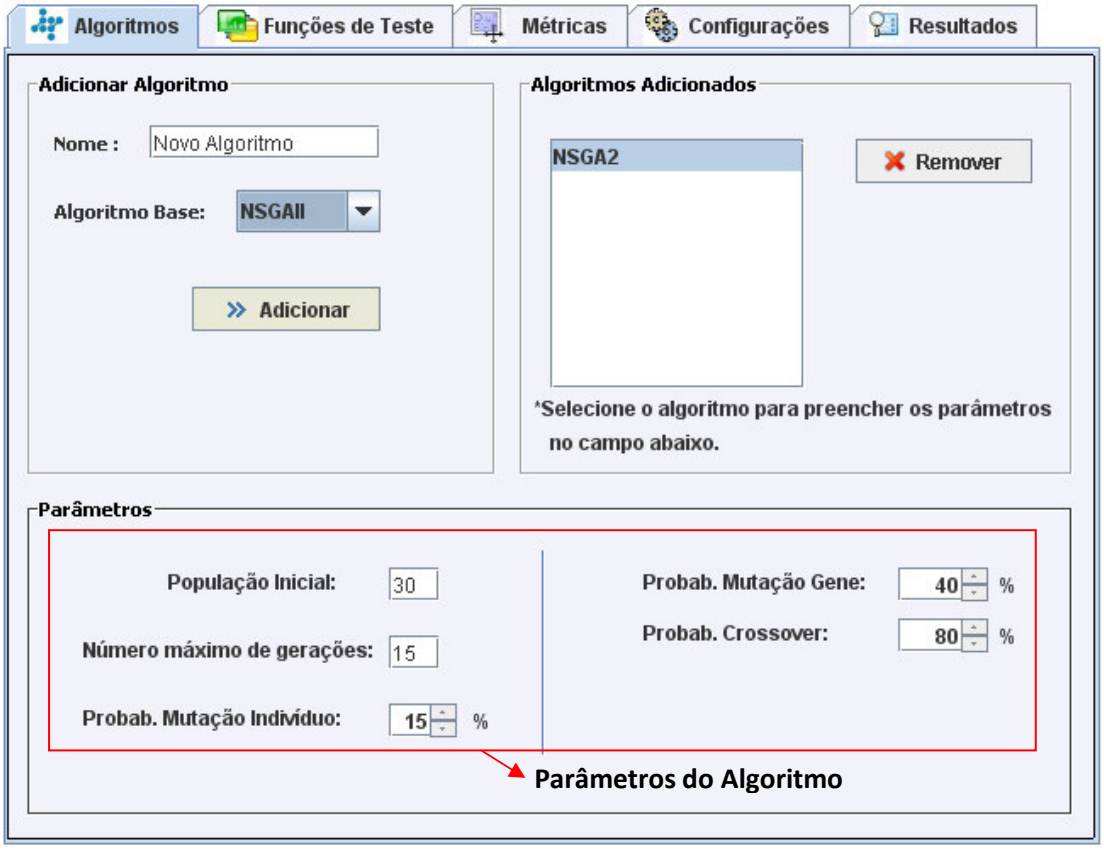

Figura 24. Tela de escolha dos algoritmos de simulação.

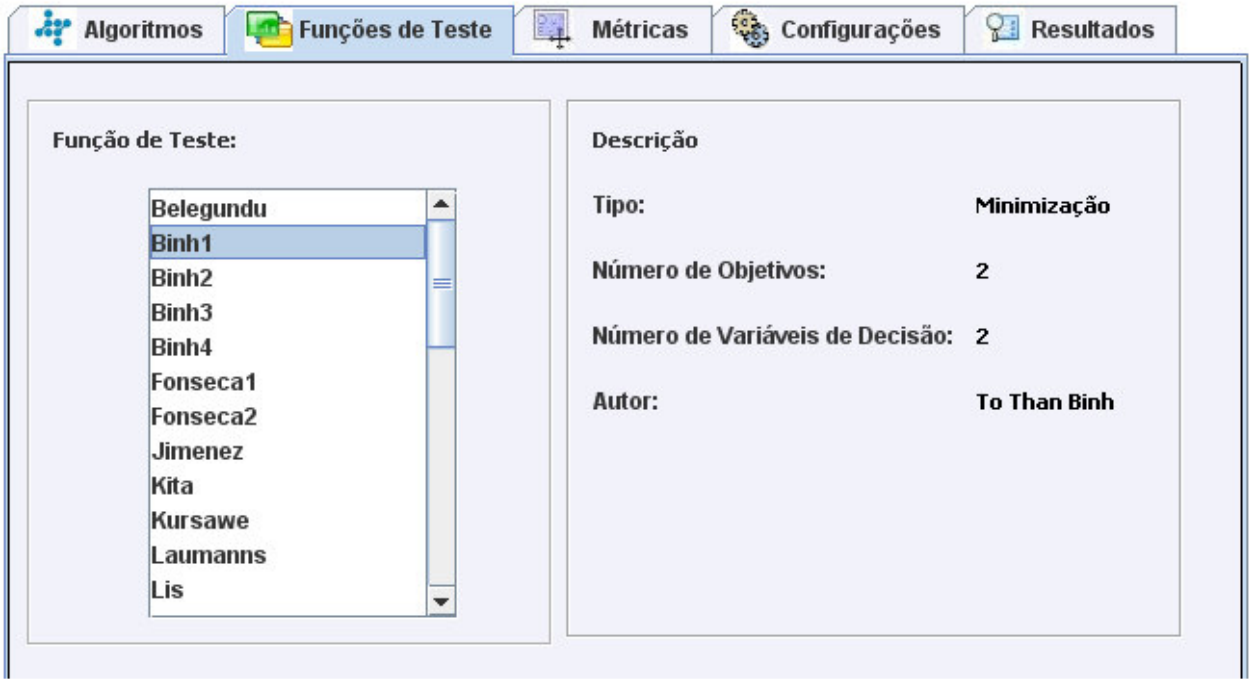

Figura 25. Tela de escolha da função de teste.

Há também uma tela de configurações que permite que o usuário escolha se quer que os resultados dos algoritmos contenham apenas o Pareto Ótimo ou todas as soluções finais. Nessa tela também é possível escolher um local para salvar a imagem do gráfico de resultados.

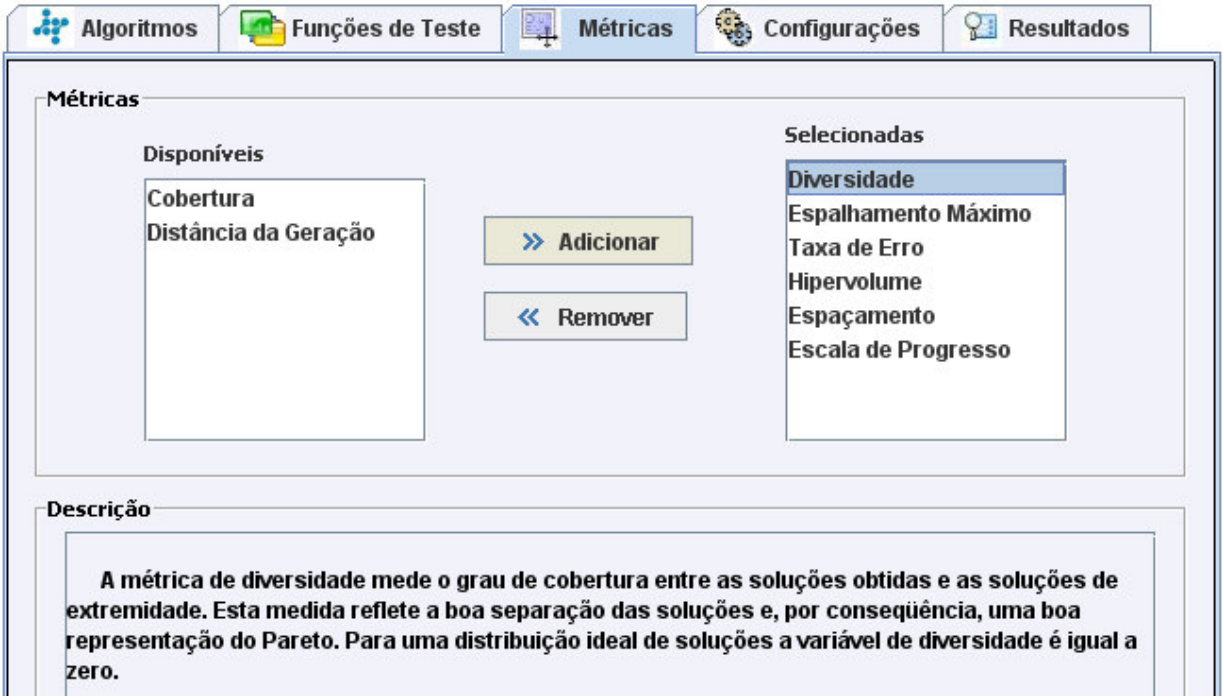

Figura 26. Tela de seleção de métricas.

Após serem selecionados os parâmetros de configuração, é realizada a simulação dos algoritmos e em seguida são apresentados seus resultados. A tela de resultados mostra os dados obtidos na simulação. Essa tela é apresentada na Figura 27.

Na tela de resultados é apresentado um gráfico onde são apresentados os indivíduos obtidos na última geração dos algoritmos simulados. Caso tenha sido escolhido mais de um algoritmo o gráfico apresentará cores distintas para cada algoritmo, tendo sua legenda logo abaixo do gráfico. Ao final da tela é possível observar os resultados das métricas que foram escolhidas pelo usuário. No canto direito é exibida a lista de algoritmos que foram simulados. Ao selecionar o algoritmo são mostrados os resultados das métricas para o mesmo.

No canto superior direito existe mais duas opções de interação com os resultados. No botão “Salvar”, pode-se escolher um local para salvar o gráfico apresentado. Ao clicar no botão “Expandir” é apresentada uma nova tela, onde é possível ajustar as dimensões do gráfico, assim como permite a aproximação e distanciamento dos pontos observados. A

Figura 28 ilustra a tela de expansão.

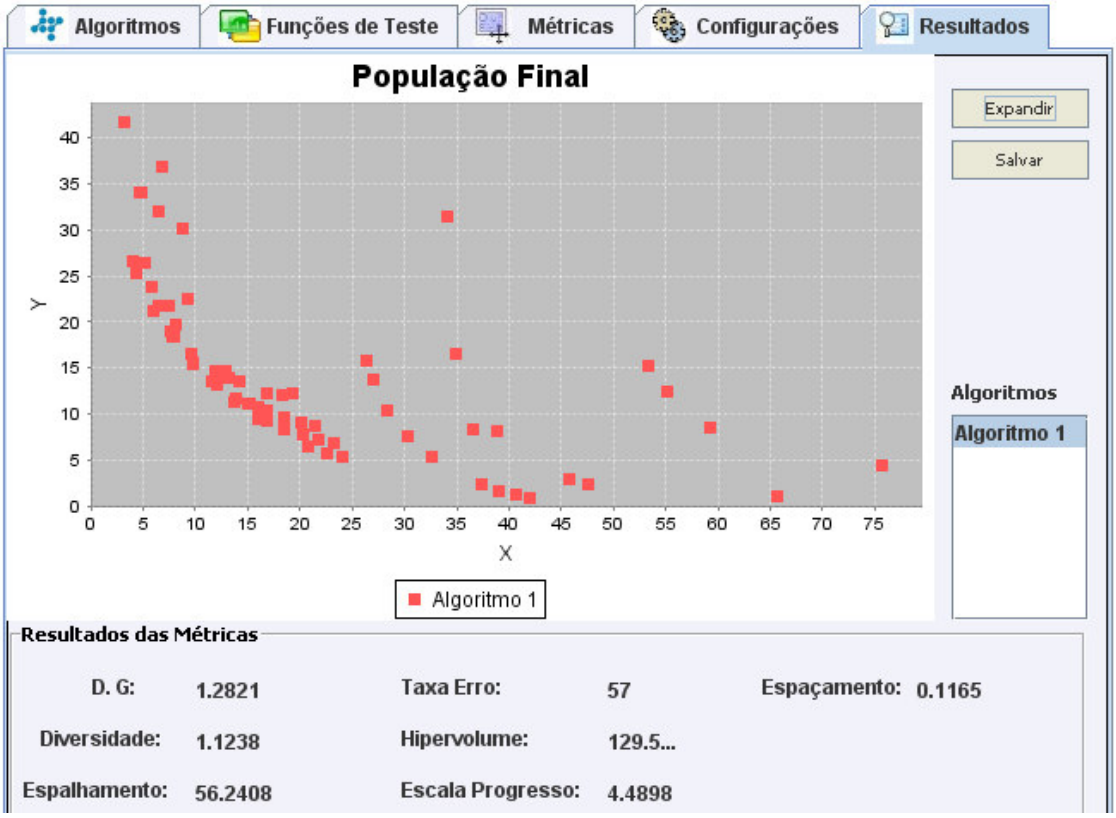

Figura 27. Tela de Resultados da Simulação.

Após o processo de simulação ter sido concluído, é possível realizar novas simulações alterando-se os parâmetros de simulação. Isso permite a análise comparativa do desempenho de diferentes algoritmos executando funções de teste distintas e com parâmetros variados. Conforme mencionado, também é possível a análise simultânea de diversos algoritmos em uma mesma simulação, observando-se assim os resultados de cada métrica e as características de seus Paretos.

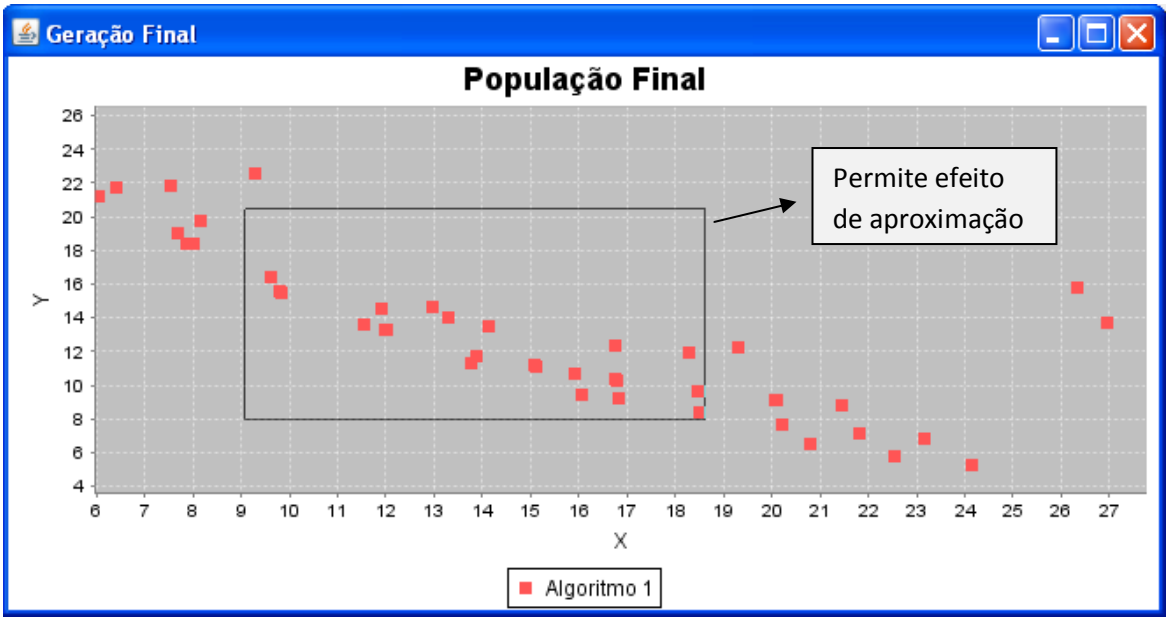

Figura 28. Tela de expansão dos resultados de simulação.

Capítulo 5

Estudo de Caso

5.1 Análise de Desempenho entre Algoritmos de Otimização Multi-Objetivos

Nesta seção são apresentados os resultados dos experimentos realizados com os algoritmos NSGAII, PAES, SPEA2 e PESA-II, simulados com a função de teste Poloni [16]. Essa função é uma função de maximização, composta por dois objetivos e duas variáveis de decisão.

A equação que a representa, assim como seu Pareto Ótimo pode ser visualizado nos apêndices deste documento.

A função de teste Poloni foi escolhida para realizar essa análise por se tratar de uma função cujo Pareto apresenta descontinuidade e concavidade, oferecendo assim maior complexidade para os algoritmos multiobjetivos [20].

A Figura 29 mostra o resultado da simulação com o algoritmo NSGAII, utilizando a ferramenta proposta. Como parâmetros de simulação foram utilizados população inicial de 40 indivíduos e número máximo de gerações igual a 100. A taxa de mutação foi ajustada em 10% e a de cruzamento em 80%.

Como pode ser visualizado na Figura 29, o Pareto *Front* obtido se aproxima bastante do Pareto Ótimo (apresentado na seção de apêndices). No entanto, na região superior do gráfico o Pareto *Front* apresenta regiões com representação ruim e pouco uniforme. Isso reflete nos resultados de algumas métricas, como por exemplo a métrica de diversidade, que apresentou um valor alto de 0,95.

A Figura 30 apresenta os resultados obtidos para a simulação do algoritmo PAES, onde também é realizado para a função de teste Poloni. Foram utilizados como parâmetros de simulação tamanho de arquivo externo igual a 20, número máximo de geração igual a 200 e número de grids por eixo igual a 20.

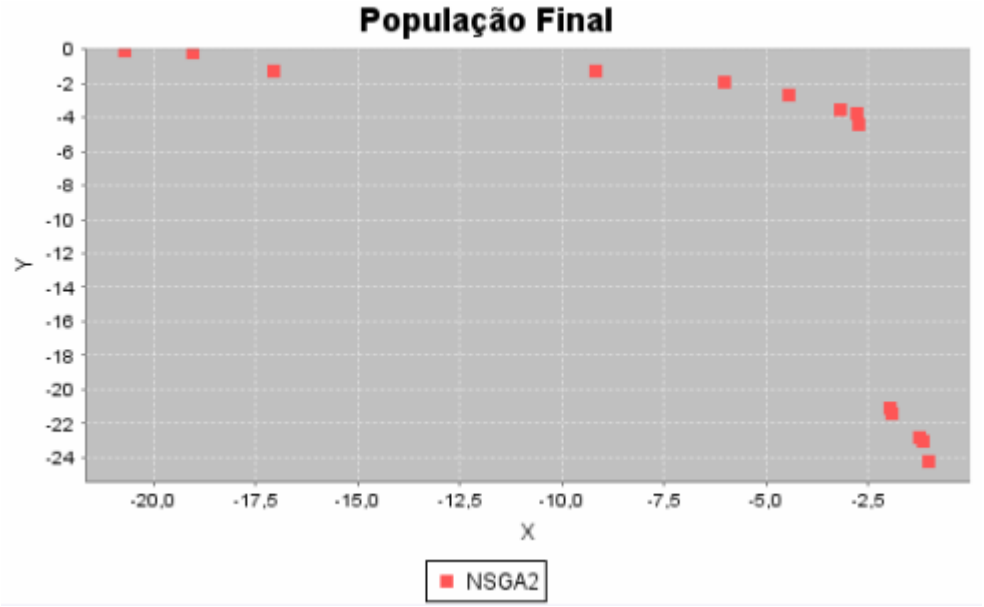

Figura 29. Pareto *Front* do algoritmo NSGAII para a função de teste Poloni.

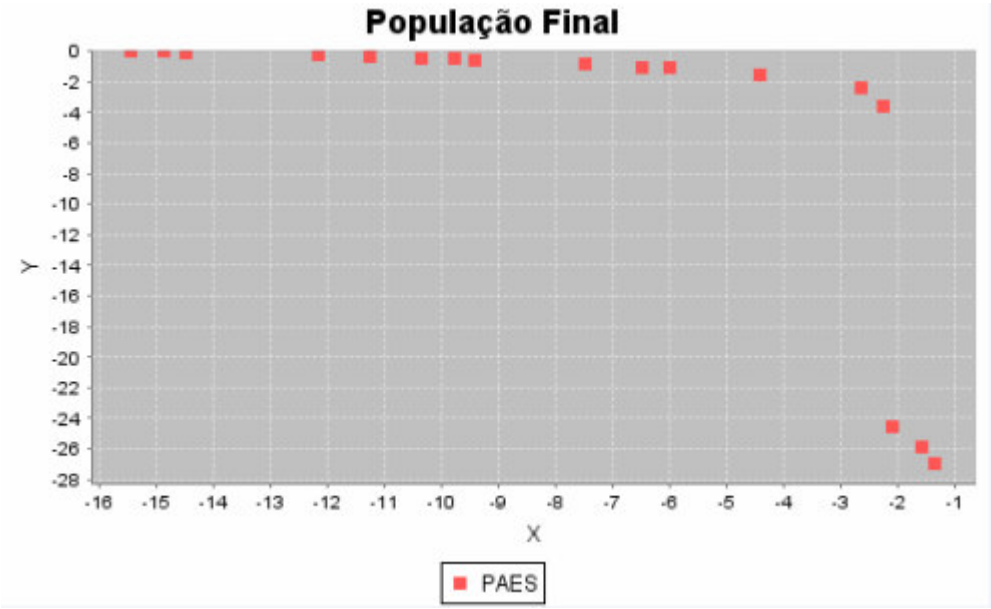

Figura 30. Pareto Front do algoritmo PAES para a função de teste Poloni.

Pode-se observar, de acordo com a Figura 30, que o Pareto *Front* econtrado apresenta-se bem próximo ao Pareto Ótimo da função de teste, apresentando o Pareto *Front* com a melhor distribuição entre os algoritmos simulados.

A Figura 31 apresenta os resultados obtidos na simulação com o algoritmo PESA-II, utilizando a função de teste Poloni. Dentre os parâmetros de simulação utilizados estão tamanho da população e número máximo de geração igual a 40, probabilidade de mutação igual a 10% e utilização de 10 *grids* por eixo.

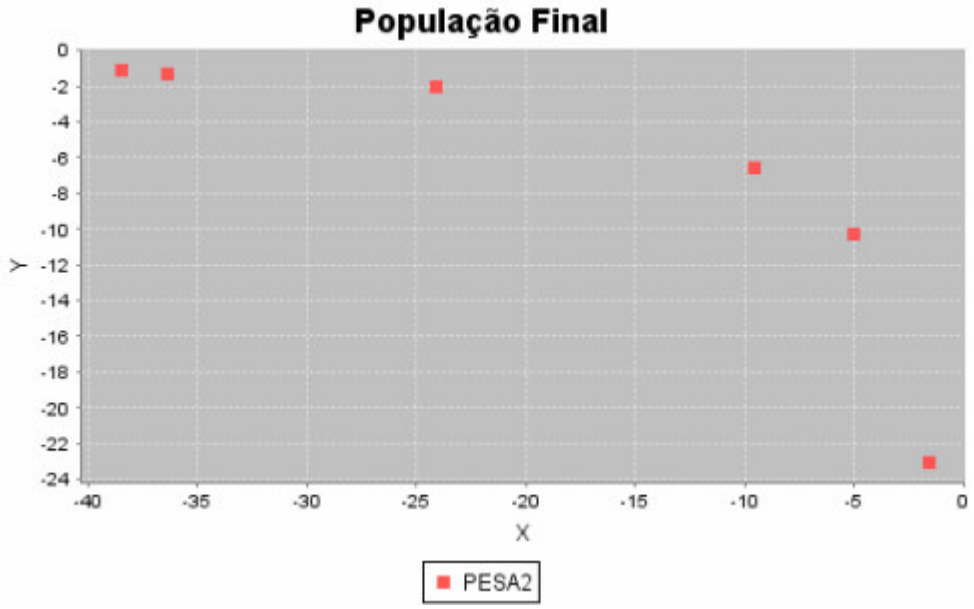

Figura 31. Pareto *Front* do algoritmo PESA-II para a função de teste Poloni.

Como pode ser visto na Figura 31, o Pareto apresentado pelo algoritmo PESA-II aproxima-se do Pareto Ótimo da função de teste, porém pouco distribuído e menos soluções não dominadas quando comparado aos outros algoritmos analisados. A métrica de diversidade apresenta bom resultado para esse algoritmo, com valor de 0,77, pois o Pareto *Front* é composto por poucas soluções afastadas entre si.

A Figura 32 apresenta os resultados obtidos na simulação com o algoritmo SPEA2, simulado para a função de teste Poloni. Para a simulação foram utilizados os seguintes parâmetros: população inicial e tamanho de arquivo externo igual a 30, número máximo de gerações igual a 200, probabilidade de mutação igual 10% e probabilidade de cruzamento igual a 80%.

De acordo com a Figura 32, o Pareto *Front* encontrado é bastante semelhante ao Pareto Ótimo da função de teste Poloni, apresentando o segundo melhor resultado entre os algoritmos analisados, por apresentar bastante soluções dentro do Pareto Ótimo, apesar de apresentar uma leve descontinuidade em seu Pareto *Front* .

Entre os resultados analisados, o algoritmo PAES apresentou o Pareto *Front* com melhor convergência, apresentando a melhor distribuição entre seus pontos e com Pareto *Front* semelhante ao Pareto Ótimo. Na seção seguinte será feito um estudo desse algoritmo para outras funções de teste.

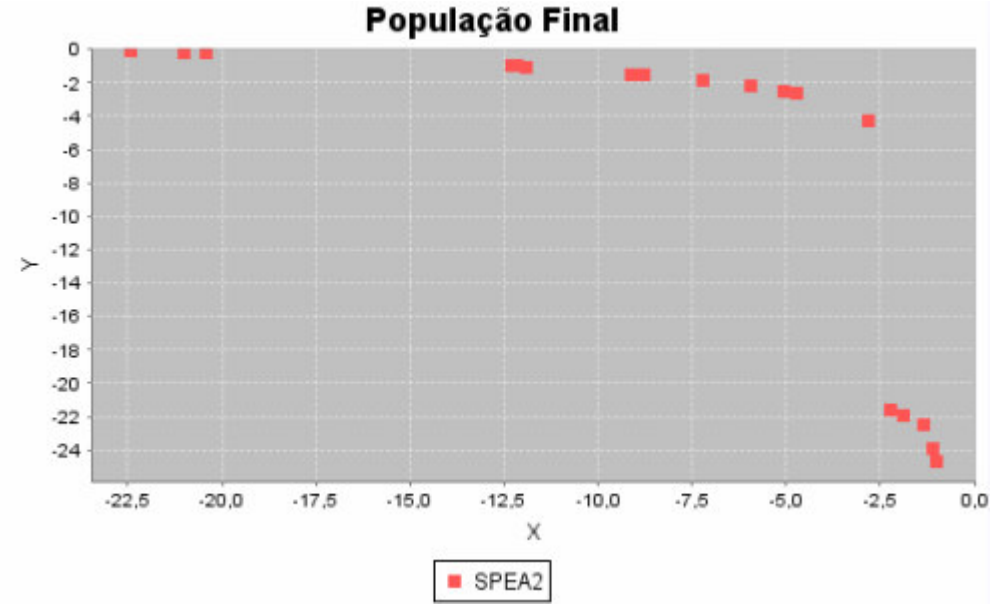

Figura 32. Pareto *Front* do algoritmo SPEA2 para a função de teste Poloni.

5.2 Análise de Desempenho com diferentes Funções de Teste

Nessa seção é realizado um estudo sobre o comportamento do algoritmo PAES para diferentes funções de teste. Todas as simulações foram realizadas utilizando-se os seguintes parâmetros de simulação: probabilidade de mutação do gene igual a 40%, número máximo de gerações igual a 200, tamanho do arquivo externo igual a 20 e número de grids por eixo igual a 20.

Na Figura 33 é apresentado o Pareto *Front* obtido na simulação utilizando a função de teste Rendon2 [35]. É possível observar a uniformidade e convergência do Pareto *Front*, apresentando-se bem próximo ao Pareto Ótimo. É possível notar que algumas soluções estão bem distantes do Pareto *Front*, isto se deve a mutações ocorridas durante o processo, o que permite a diversidade das soluções encontradas.

Na Figura 34 é apresentado o Pareto *Front* para a função de teste Kursawe [17]. Por apresentar um Pareto Ótimo côncavo, oferece maior dificuldade para os algoritmos de otimização. De acordo com o resultado apresentado pela figura, é possível observar que o Pareto *Front* encontrado conseguiu se aproximar do Pareto Ótimo, embora tenha encontrado poucas soluções devido ao número de restrições envolvidas na função de teste.

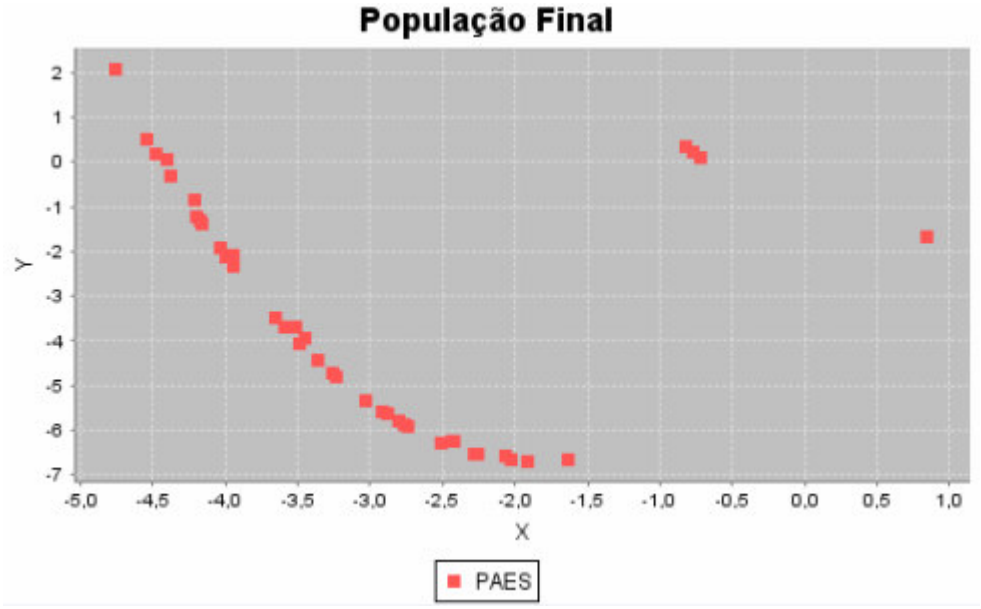

Figura 33. Pareto *Front* do algoritmo PAES para a função de teste Rendon2.

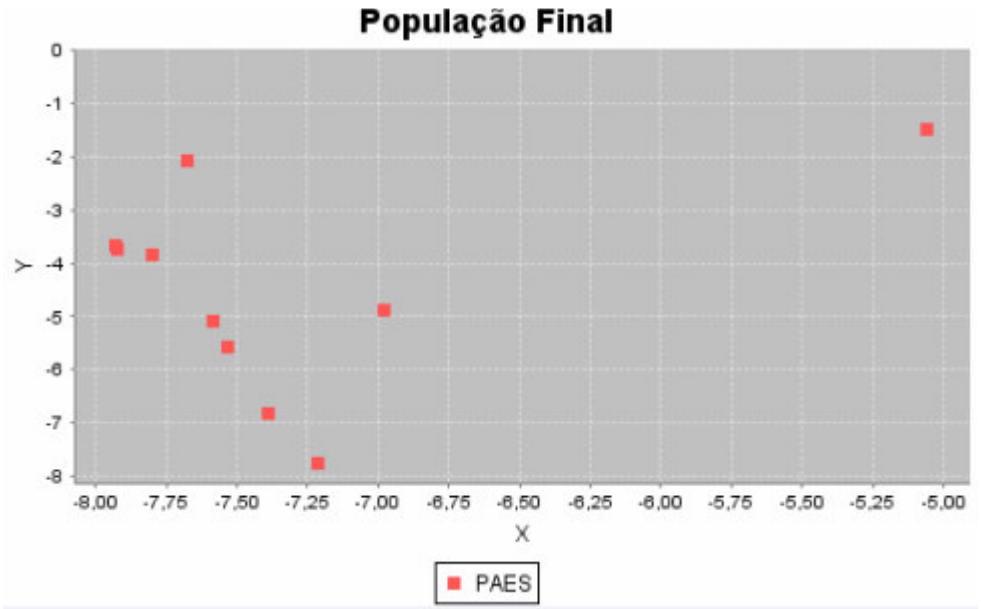

Figura 34. Pareto *Front* do algoritmo PAES para a função de teste Kursawe.

Na Figura 35 é apresentado o apresentado o Pareto *Front* encontrado para a função de teste Belegundu [36], cujo Pareto Ótimo é representado por uma função linear. Apesar de o Pareto Ótimo parecer simples de ser encontrado, o algoritmo PAES não conseguiu obter uma boa distribuição ao longo do Pareto *Front*, havendo maior densidade de soluções em apenas uma parte do Pareto.

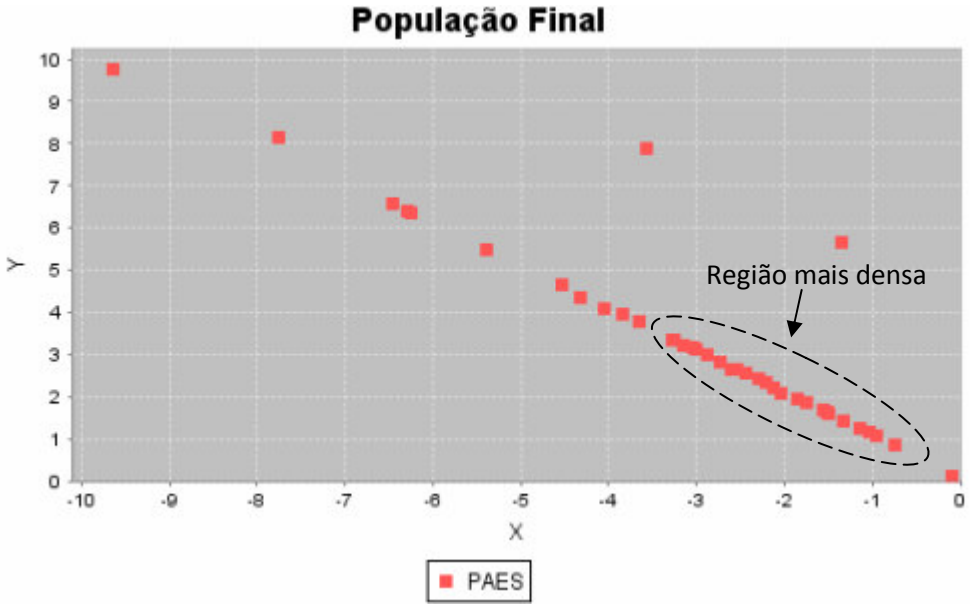

Figura 35. Pareto *Front* do algoritmo PAES para a função de teste Belegundu.

A descontinuidade apresentada no Pareto *Front* encontrado na Figura 35 pode ser melhorada através de ajustes nos parâmetros de simulação dos algoritmos, realizada na aba de algoritmos da ferramenta. Essa observação permitiu a análise de um terceiro estudo de caso que verifica o impacto do da variação de parâmetros no Pareto *Front,* conforme é mostrado na seção seguinte.

5.3 Análise do Impacto da Variação dos Parâmetros

Inicialmente foi realizada a simulação do algoritmo, para a função de teste Binh3 [37], utilizando-se os seguintes parâmetros: população inicial e número máximo de gerações iguais a 30, tamanho de arquivo externo igual a 10, probabilidade de mutação de 10%, probabilidade de cruzamento de 80% e cinco *grids* por eixo.

Na Figura 36 é apresentado o Pareto *Front* encontrado a partir da simulação com os parâmetros de simulação descritos. Apesar da aproximação do Pareto *Front* em relação ao Pareto Ótimo, é possível observar que as soluções não estão dispostas uniformemente ao longo do Pareto *Front.*

É realizada uma segunda simulação para o algoritmo PESA2, utilizando-se a mesma função de teste. Para essa simulação são mantidos os mesmos parâmetros de simulação utilizados anteriormente, variando apenas o número de *grids* por eixo de 5 para 20.

Na Figura 37 é apresentado o Pareto *Front* obtido para com a nova configuração de parâmetros. É possível observar que ouve uma melhora da distribuição de soluções no Pareto

Front em relação ao experimento anterior. Também foi possível notar uma diminuição no valor da métrica de diversidade, onde no primeiro experimento era de 1,6 e no segundo passou a ser de 0,82. Pode-se concluir que isso ocorre devido à diminuição do tamanho dos *grids*, havendo uma maior percepção do algoritmo das regiões que estão menos densamente povoadas.

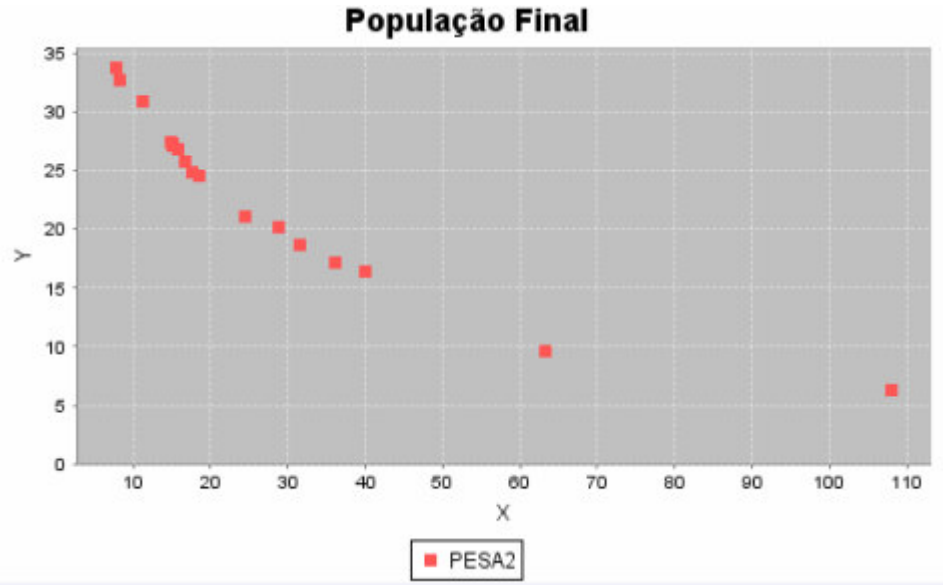

Figura 36. Pareto *Front* do algoritmo PESA2 para a função de teste Binh3, com 5 grids por eixo.

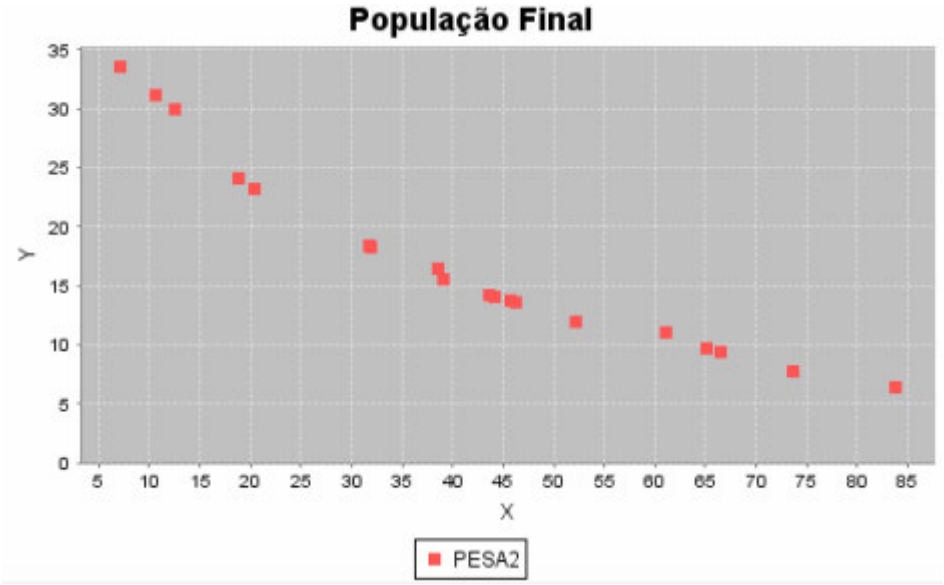

Figura 37. Pareto *Front* do algoritmo PESA2, para a função Binh3, com 20 grids por eixo.

Outras variações foram utilizadas, tais como variação de probabilidade de mutação. Para esse caso foi observado que altas taxas de probabilidade de mutação causam uma maior incerteza

nos resultados finais, podendo ajudar a convergir mais devagar, assim como a qualidade do Pareto, deixando o resultado muito aleatório.

Capítulo 6

Considerações Finais e Trabalhos Futuros

Foi apresentada, neste trabalho, uma ferramenta de simulação e análise de algoritmos evolucionários multi-objetivos, onde foram implementadas algumas das principais técnicas existentes, dentre elas: NSGAII, PAES, PESA-II e SPEA2. O desenvolvimento da ferramenta foi realizado de forma modular, utilizando arquitetura em camadas, o que permitirá a adição de futuras funcionalidades à ferramenta, de forma simplificada.

Foi realizado um levantamento bibliográfico, obtendo-se as principais funções de testes encontradas na literatura, as quais foram agregadas à ferramenta proposta. Foi realizada a implementação de 36 funções de teste, dentre as que foram coletadas, oferecendo ao usuário a possibilidade de simular diversos domínios de problema, com diferentes graus de dificuldade, a fim de permitir uma validação mais completa dos algoritmos analisados. Para avaliar os resultados encontrados, foram adicionadas 8 métricas à ferramenta, as quais estão entre as mais utilizadas na validação de MOEAs.

Na seção de Estudo de Caso foi possível comparar os resultados obtidos na ferramenta, demonstrando a usabilidade do sistema e a capacidade de análise das técnicas implementadas. Dessa forma, foi realizado um estudo comparativo entre os algoritmos NSGAII, PAES, PESA2 e SPEA2, simulados para a função de teste Poloni. Nesse estudo foi possível observar que todos os algoritmos conseguiam obter um Pareto *Front* próximo ao Pareto Ótimo da função, demonstrando que todos os algoritmos convergiam. No entanto, o Algoritmo PAES apresentou melhores resultados por apresentar um Pareto mais uniformemente distribuído, o que é uma das condições almejadas em um MOEA. Em seguida, uma análise do algoritmo PAES, realizando-se simulações para outras funções de teste. Foi observado que para outras funções de teste também eram obtidos Paretos *Front* com boa convergência, apesar de que, conforme observado, o número de restrições da função de teste pode interferir no número de soluções finais encontradas e na distribuição de soluções do Pareto *Front*.

Outra análise realizada, utilizando-se a ferramenta proposta, foi feita através da variação de parâmetros de simulação, utilizando-se o algoritmo PESA-II, para a função de teste Binh3. Nesse estudo foi verificado que a quantidade de *grids* por eixo utilizado no algoritmo pode afetar significativamente o grau de diversidade apresentado pelo Pareto *Front*. Foi possível observar também que o aumento da taxa de mutação do indivíduo na população pode provocar um efeito aleatório indesejado nos resultados do algoritmo, podendo em determinados momentos ajudar ou dificultar a convergência do mesmo.

Por fim, pôde-se obter uma ferramenta de simulação de MOEAs, que, através de uma interface amigável e intuitiva, permite realizar a simulação e análise dos algoritmos NSGAII, PAES, PESA-II e SPEA2, além de oferecer um conjunto de 36 funções de teste para validação dos algoritmos e 8 métricas de análise de resultados.

Dentre as dificuldades encontradas, pode-se destacar a dificuldade de adaptação de algumas funções de teste encontradas em artigos científicos, o que proporcionou, em alguns casos, a não identificação dos Paretos Ótimos simulados com os dos apresentados nos artigos.

Entre os trabalhos futuros é pretendido melhorar o nível de compatibilidade entre os Paretos Ótimos obtidos em algumas funções de teste e os dos representados nos artigos correspondentes.

Pretende-se também, futuramente, realizar a adição de novos algoritmos à ferramenta. O objetivo é que a adição dos novos algoritmos seja feita pelo usuário através da própria ferramenta, sendo possível a adição de variações das técnicas existentes ou a criação de novas técnicas definidas pelo usuário.

Bibliografia

[1] COELLO, C. A. C. **Evolutionary Multi-Objective Optimizaton : A Historical View of the Field**, IEEE Computational Intelligence Maganize,

[2] DEB K. **Multi-Objective Optimization using Evolutionary Algorithms**, Wiley-Interscience series in systems and optimization, p 17.

[3] PARETO, V. **Cours D'Economie Politique,** vol I e II, 1986.

[4] KNOWLES, J. e CORNE D. , **Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy,** Evolutionary Computation 8(2): 149-172.

[5] DEB, K. AGRAWAL, S. e MEYARIVAN T., **A Fast Elitist Nondominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II**, Parallel Problem Solving from Nature, 2000, p. 849-858.

[6] ZITZLER, E., LAUMMANS, M., THIELE, L. **SPEA2: Improving the Strength of Pareto Evolutionary Algorithm for Multiobjective Optimization**, Technical Report 103, Computer Engineering and Networks Laboratory(TIK), Swiss Federal Institute of Technology (ETH), Switzerland, 2001.

[7] CORNE, D.W., JERRAM, N.R., KNOWLES, J. D., OATES, M. J., **PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization**. Proceedings of the Genetic and Evolutionary Computation Conference, São Francisco, 2001.

[8] SILVA-FILHO, A. G., *et al*. **An Optimization Mechanism Intended for Two-Level Cache Hierarchy to Improve Energy and Performance using the NSGAII Algorithm**. In: SBAC-PAD, 2008, Campo Grande - MS. 20th International Symposium on Computer Architecture and High Performance Computing. Danvers, MA : IEEE Computer Society - Conference Publishing Services, 2008. v. v1. p. 19-2

[9] DURILLO, J. J. *et al*, **jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics,** Tech *Report*, 2006.

[10] VELDHUIZEN, D. VAN, **Multiobjective evolucionary algortihms: Classifications, analyzes, and new innovations**, Air Force Inst. Technol., Dayton, OH, Tech. Rep. 1999.

[11] WOLPERT, DAVID H. e WILLIAM G. MACREADY. **No Free Lunch Theorems for Optimization**, IEEE Transactions on Evolucionary Computation, 1(1):67-82, Abril, 1997.

[12] DE JONG, KENNETH A. **An Analysis of the Behavior of a Class of Genetic Adaptive Systems**, dissertação PhD, The University of Michigan, Ann Arbor MI, 1975.

[13] GOLDBERG, DAVID E. **Genetic Algorithms in Search Otimization and Machine**. Reading, MA: Addison Wesley, 1989.

[14] WHITLEY, DARRELL, *et al.* **Evaluating Evolutionary Algorithms**, Artificial Intelligence, 85:245-276, 1996.

[15] FONSECA, C. M. e FLEMING, P.J. **Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part II: Application example**, IEEE Trans. Syst., Man, Cybern. A, vol. 28, p. 38-47, 1998.

[16] POLONI, C., *et al.*, **Multiobjective Optimization by Gas: Application to System and Component Design.** Computational Methods in Applied Sciences '96, p. 258-264, 1996.

[17] KURSAWE, F., **A Variant of Evolution Strategies for Vector Optimization**, Parallel Problem Solving from Nature. Berlin: Springer-Verlag, 1990.

[18] DEB, K., PRATAP, A., AGRAWL, S. e MEYARIVAN, T. **A fast and elitist multiobjective genetic algorithm: NSGA-II**. Technical Report No. 2000001, 2000.

[19] ZITZLER, E., DEB, D., e THIELE L. **Comparison of multiobjective evolutionary algorithms : Empirical results**, Evol. Comput., vol 8, no. 2, p. 173-195, 2000.

[20] DEB, K. **Multiobjective genetic algorithms: Problems difficulties and construction of test functions**, Evol. Comput., 1999, vol 7, p. 205-230.

[21] FONSECA, C. M., KNOWLES, J. D., THIELE, L. e ZITZLER, E. **A tutorial on the performance assessment of stochastic multiobjective optimizers**. Third International Conference on Evolutionary Multi-Criterion Optimization, 2005.

[22] VELDHUIZEN D. A. V., LAMONT, G. B. **On measuring multiobjective evolutionary algorithm performance**, Congress on Evolutionary Computation, 2000, vol. 1, p. 204-211.

[23] VELDHUIZEN D. A. V., LAMONT G. B. **Multiobjective evolutionary algorithm research: A history and analysis**,Tech. Rep. TR-98-03, 1998.

[24] TAN, K. C., LEE, T. H. e KHOR, E. F, **Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization**, vol. 5, p. 565-599, 2001.

[25] ZITZLER, E. **Evolutionary algorithms for multiobjective optimization: Methods and applications**, dissertação Ph.D, 1999.

[26] ZITZLER, E., THIELE, L. **Multiobjective optimization using evolutionary algorithms – A comparative case study**, Parallell Problem Solving from Nature, 1998, p. 292-301.

[27] SCHOOT, J. R. **Fault tolerant design using single and multicriteria genetic algorithm optimization**, Tese de Mestrado, 1995.

[28] TOMAS, B. **Evolutionary Algorithms in Theory and Practice**, 1996.

[29] SRINIVAS, N. e DEB, K., **Multiobjective Function Otimization Using Nondominated Sorting Genetic Algorithms**, Evol. Comput., vol 2, no 3, p. 221-248, 1995.

[30] CORNE, D. W., KNOWLES, D. J., OATES M. J., **The Pareto Envelope-based selection algorithm for multiobjective optimization**. Proceedings of sixth international conference on parallel problem solving from Nature, p. 18-20, 2000.

[31] Site da Sun. Última visualização em 02/11/2008.

[32] Site da ferramenta Eclipse. Última visualização em 02/11/2008.

[33] Site da ferramenta Netbeans. Última visualização em 02/11/2008.

[34] Site do JFreeChart. Última visualização em 02/11/2008.

[35] RENDÓN, V., URESTI-CHARRE, M., URESTI-CHARRE, E., **A Non-Generational Genetic Algorithm for Multiobjective Optimization**, p. 658-655.

[36] BELEGUNDU, A. D., *et al,* **Multi-Objective Optimization of Laminated Ceramic Composites Using Genetic Algorithms**, Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, p. 1015-1022, 1994.

[37] BINH, T. T. **A Multiobjective Evolutionary Algorithm: The Study Cases.** Technical Report, Institue for Automation and Communication, Barleben, Alemanha, 1999.

[38] BINH, T. T, KORN, U, **An Evolution Strategy for the Multiobjective Optimization**. Proceedings of the Second International Conference on Genetic Algorithms, p. 23-28, 1996.

[39] BINH, T. T., KORN, U., **Multiobjective Evoluion Strategy with Linear and Nonlinear Constraints,** Proc. of the 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics, p. 357-362, 1997.

[40] FONSECA, C. M, FLEMING, P. J., **An Overview of Evolutionary Algorithms in Multiobjective Optimization**, Evolutionary Computation, 1995.

[41] FONSECA, C. M, FLEMING, P. J., **Multiobjective Genetic Algorithms Made Easy: Selection, Sharing, and Mating Restriction**. Proceedings of the 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995.

[42] JIMÉNEZ, F., VERDEGAY, J. L., **Contrained Multiobjective Optimization by Evolutionary Algorithms.** Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, 1998.

[43] KITA, H., *et. al.,* **Multi-Objective Optimization by Means of the Thermodynamic Genetic Algorithm**. Parallel Problem Solving from Nature – PPSN IV, p. 504-512, 1996.

[44] LAUMANNS, M., *et. al,* **A Spatial Predator-Prey Approach to Multi-Objective Optimization,** 1996.

[45] LIS, J., EIBEN, A. E, **A Multi-Sexual Genetic Algorithm for Multiobjective Optimization,** Proceedings of the 1997 (4^{th}) IEEE International Conference on Evolutionary Computation, Piscataway, 1997.

[46] MURATA, T, *et. al.*, **Multi-Objective Genetic Algorithm and Its Application to Flowshop Scheduling**, Computers and Industrial Engineering Journal, p 957-968, 1996.

[47] OBAYASHI, S., **Pareto Genetic Algorithm for Aerodynamic Design Using the Navier-Stokes Equations.** Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications, p. 245-266, 1997.

[48] OSYCZKA, A., KUNDU, S., **A New Method to Solve Generalized Multicriteria Optimization Problems Using the Simple Genetic Algorithm**, Structural Optimization, 1995.

[49] QUALGLIARELLA, D., VICINI, A., **Subpopulation Policies for a Parallel Multiobjective Genetic Algorithm with Applications to Wing Design**, Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, 1998.

[50] VALENZUELA-RENDÓN, M., URESTI-CHARRE, E., **A Non-Generational Genetic Algorithm for Multiobjective Optimization,** Proceedings of the Seventh Internation Conference on Genetics Algorithms, p. 658-665, 1997.

[51] SCHAFFER, J. D, **Multiple Objective Optimization with Vector Evaluated Genetic Algorithms,** Proceedings of the First International Conference on Genetic Algorithms and Their Applications, p. 93-100, 1985.

[52] SRINIVAS, N., DEB. K., **Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms**, Evolutionary Computation, p. 221-248, 1994.

[53] TAMAKI, H, *et. al.,* **Multi-Objective Optimization by Genetic Algorithms: A Review**. Proceedings of the Third IEEE Conference on Evolutionary Computation, p. 517-522, 1996.

[54] TANAKA, M., *et al.,* **GA-Based Decision Support System for Multicriteria Optimization.** Proceedings of the International Conference on Systems, Man, and Cybernetics2, p. 1556-1561, 1995.

[55] VIENNET, R., *et. al.,* **Multicriteraia Optimization Using a Genetic Algorithm for Determining a Pareto Set**, International Journal of Systems Science, p. 255-260, 1996.

Apêndice A – Funções de Teste

Esta seção apresenta as funções de teste utilizadas na ferramenta proposta, com o intuito de mostrar a estrutura das equações, assim como as imagens dos Paretos-Ótimos dessas funções. As figuras que demonstram os Paretos das funções utilizadas nessa seção são adaptações das imagens contidas em [10]. A seguir são descritas cada função de teste.

Belegundu

A função de teste Belegundu foi originalmente proposta em [36] e é representada por

$$F = \big(f_1(x,y), f_2(x,y)\big), \tag{52}$$

onde

$$f_1(x,y) = -2x + y, \tag{53}$$

$$f_2(x,y) = 2x + y, \tag{54}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pelas equações

$$0 \leq x \leq 5, 0 \leq y \leq 3, \tag{55}$$

$$0 \geq -x + y - 1, \tag{56}$$

$$0 \geq x + y - 7. \tag{57}$$

A Figura 38 apresenta o Pareto Ótimo dessa função de teste.

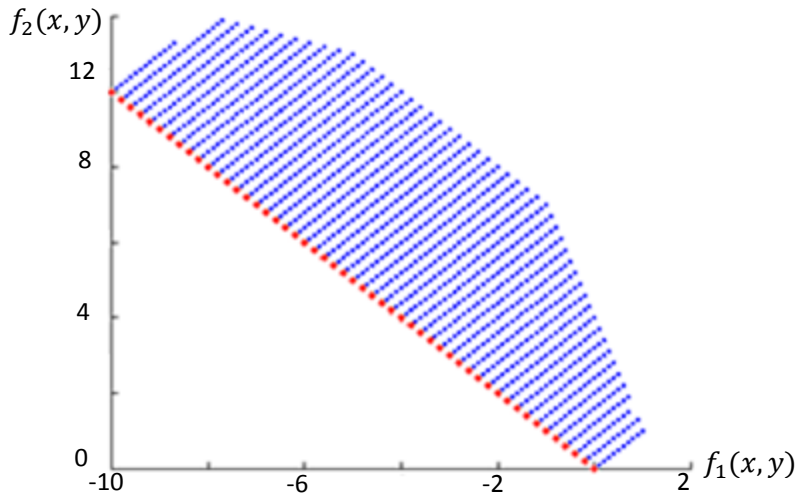

Figura 38. Pareto Ótimo da função de teste Belegundu.

Binh(1)

A função de teste Binh(1) foi originalmente proposta em [38] e é representada pela equação (52), onde

$$f_1(x,y) = x^2 + y^2, \quad (58)$$

$$f_2(x,y) = (x-5)^2 + (x-5)^2, \quad (59)$$

onde *x* e *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, possuindo as seguintes restrições:

$$-5 \leq x,y \leq 10 \quad (60)$$

O Pareto Ótimo dessa função de teste é mostrado na Figura 39. Os pontos em vermelho representam as soluções que compõem o Pareto Ótimo, enquanto que os pontos em azul representam outras soluções possíveis de serem encontradas.

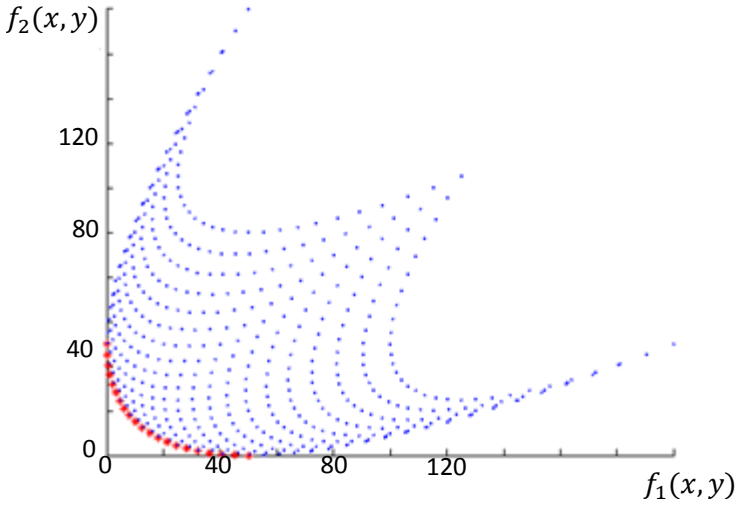

Figura 39. Pareto Ótimo da função de teste Binh(1).

Binh (2)

A função de teste Binh (2) foi originalmente proposta em [38] e é representada pela equação (52), onde

$$f_1(x,y) = 4x^2 + 4y^2, \quad (61)$$

$$f_2(x,y) = (x-5)^2 + (y-5)^2, \quad (62)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pelas seguintes equações:

$$0 \leq x \leq 5, 0 \leq y \leq 3, \tag{63}$$

$$0 \geq (x-5)^2 + y^2 - 25, \tag{64}$$

$$0 \geq -(x-8)^2 - (y+3)^2 + 7.7. \tag{65}$$

A Figura 40 apresenta o Pareto Ótimo dessa função de teste.

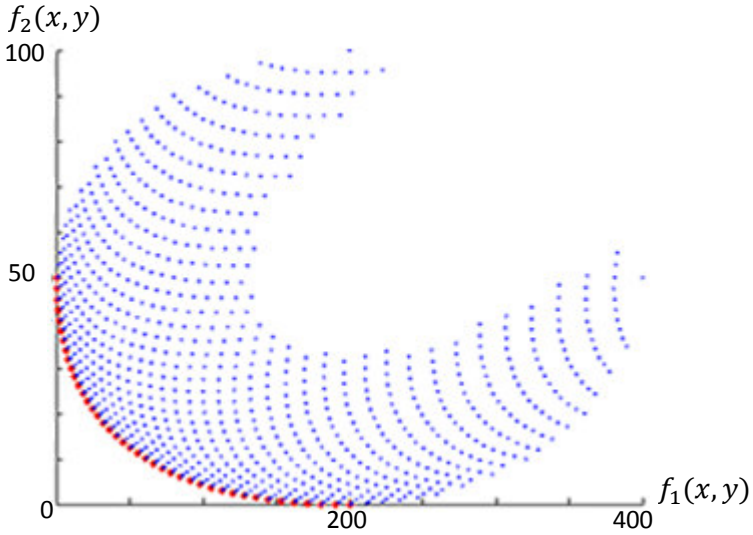

Figura 40. Pareto Ótimo da função de teste Binh (2).

Binh(3)

A função de teste Binh(3) foi originalmente proposta em [37] e é representada por

$$F = \big(f_1(x,y), f_2(x,y), f_3(x,y)\big), \tag{66}$$

onde

$$f_1(x,y) = x - 10^6, \tag{67}$$

$$f_2(x,y) = y - 2 * 10^{-6}, \tag{68}$$

$$f_3(x,y) = xy - 2, \tag{69}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$, $f_2(x,y)$ e $f_3(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, que mapeia duas variáveis de decisão para três objetivos. Seu domínio é restringido pela seguinte equação:

$$10^{-6} \leq x, y \leq 10^{6}. \quad (70)$$

Binh (4)

A função de teste Binh (4) foi originalmente proposta em [39] e é representada pela equação (66), onde

$$f_1(x, y) = 1.5 - x(1 - y), \quad (71)$$

$$f_2(x, y) = 2.25 - x(1 - y^2), \quad (72)$$

$$f_3(x, y) = 2.625 - x(1 - y^3), \quad (73)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x, y)$, $f_2(x, y)$ e $f_3(x, y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, que mapeia duas variáveis de decisão para três objetivos. Seu domínio é restringido pelas equações

$$-10 \leq x, y \leq 10, \quad (74)$$

$$0 \geq -x^2 - (y - 0.5)^2 + 9, \quad (75)$$

$$0 \geq (x - 1)^2 + (y - 0.5)^2 - 6.25. \quad (76)$$

Fonseca(1)

A função de teste Fonseca(1) foi originalmente proposta em [40] e é representada pela equação (52), onde

$$f_1(x, y) = 1 - \exp(-(x - 1)^2 - (y + 1)^2), \quad (77)$$

$$f_2(x, y) = 1 - \exp(-(x + 1)^2 - (y - 1)^2), \quad (78)$$

onde *x*, *y* representam as variáveis de decisão e $\boldsymbol{f_1(x, y)}$ e $\boldsymbol{f_2(x, y)}$ representam os objetivos do problema. A função *F* representa um problema de maximização, não apresentando nenhuma restrição. O Pareto Ótimo dessa função pode ser visualizado na Figura 41.

Fonseca(2)

A função de teste Fonseca(2) foi originalmente proposta em [41] e é representada por

$$F = \left(f_1(\vec{x}), f_2(\vec{x})\right), \quad (79)$$

onde

$$f_1(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n} x_i - \frac{1}{\sqrt{n}}^2\right), \quad (80)$$

$$f_2(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n} x_i + \frac{1}{\sqrt{n}}^2\right), \quad (81)$$

onde $\vec{x}$ representa um vetor de entrada com *n* variáveis, x_i representa a *i-ésima* variável de entrada da função e $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos do problema..

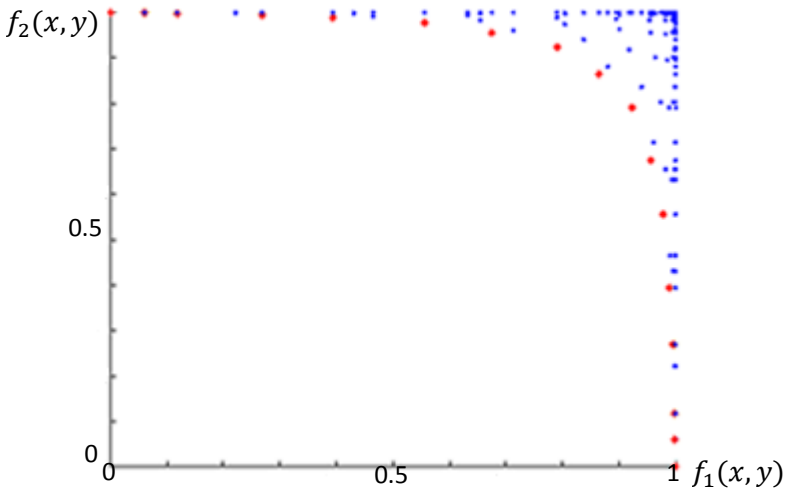

Figura 41. Pareto Front da função de teste Fonseca(1).

A função *F* representa um problema de maximização, que mapeia *n* variáveis de decisão para dois objetivos. Seu domínio é restringido pela equação

$$-4 \leq x_i \leq 4. \quad (82)$$

O Pareto Ótimo dessa função de teste, no caso de termos apenas duas variáveis de decisão, *x* e *y*, é mostrado na Figura 42.

Jimenez

A função de teste Jimenez foi originalmente proposta em [42] e é representada pela equação (52), onde

$$f_1(x,y) = 5x + 3y, \quad (83)$$

$$f_2(x,y) = 2x + 8y, \quad (84)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema.

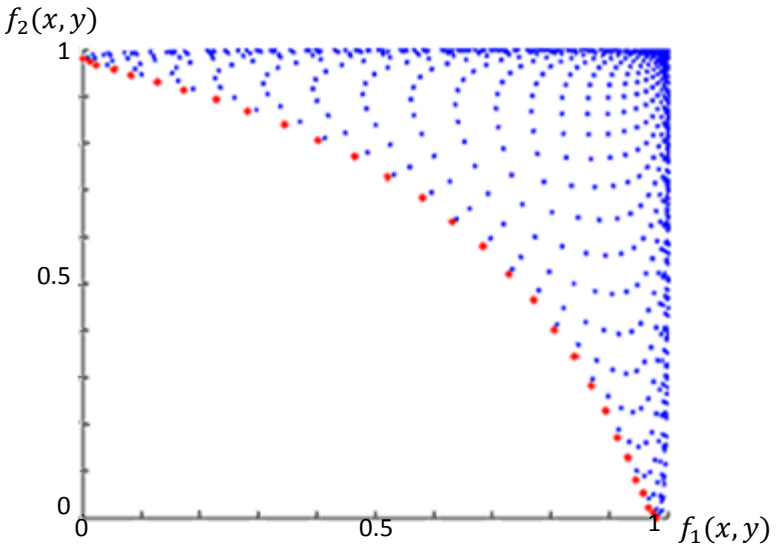

Figura 42. Pareto Ótimo da função de teste Fonseca(2).

A função F representa um problema de maximização e seu domínio é restringido pelas equações

$$x, y \geq 0, \tag{85}$$

$$0 \geq x + 4y - 100, \tag{86}$$

$$0 \geq 3x + 2y - 150, \tag{87}$$

$$0 \geq 200 - 5x - 3y, \tag{88}$$

$$0 \geq 75 - 2x - 8y. \tag{89}$$

A Figura 43 apresenta o Pareto Ótimo dessa função de teste.

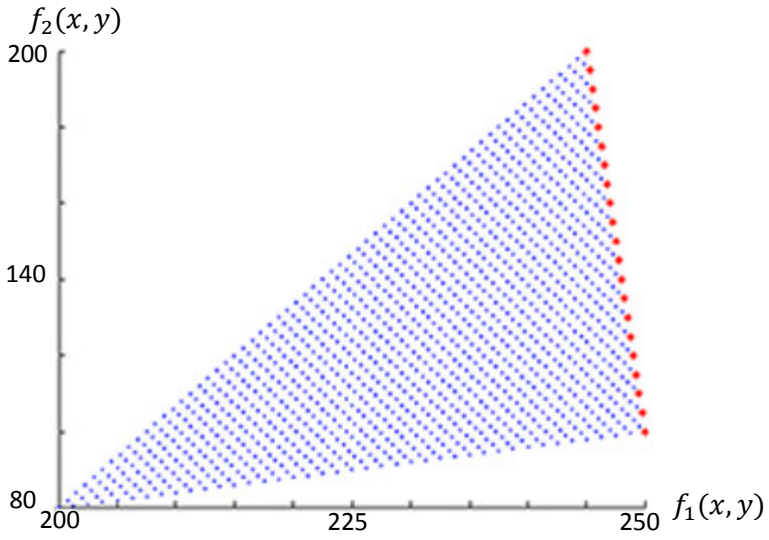

Figura 43. Pareto Ótimo da função de teste Jimenez.

Kita

A função de teste Kita foi originalmente proposta em [43] e é representada pela equação (52), onde

$$f_1(x,y) = -x^2 + y, \tag{90}$$

$$f_2 = \frac{1}{2}x + y + 1, \tag{91}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de maximização e seu domínio é restringido pelas equações

$$x, y \geq 0, \tag{92}$$

$$0 \geq \frac{1}{6}x + y - \frac{13}{2}, \tag{93}$$

$$0 \geq \frac{1}{6}x + y - \frac{13}{2}, \tag{94}$$

$$0 \geq \frac{1}{2}x + y - \frac{15}{2}, \tag{95}$$

$$0 \geq 5x + y - 30. \tag{96}$$

A Figura 44 apresenta o Pareto Ótimo dessa função de teste.

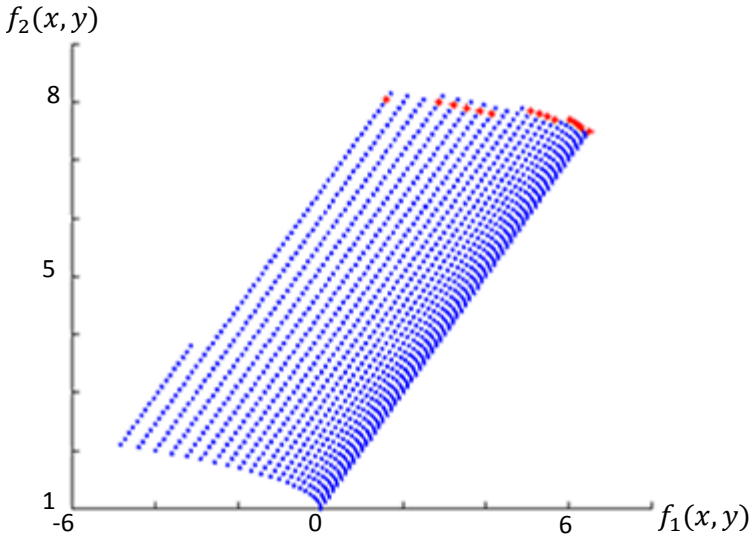

Figura 44. Pareto Ótimo da função de teste Kita.

Kursawe

A função de teste Kursawe foi originalmente proposta em [17] e é representada pela equação (79), onde

$$f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(-10e^{(-0.2)*\sqrt{x_i{}^2+x_{i+1}{}^2}}\right), \quad (97)$$

$$f_2(\vec{x}) = \sum_{i=1}^{n} (|x_i|^{0.8} + 5sin(x_i)^3), \quad (98)$$

onde $\vec{x}$ representa um vetor de entrada com *n* variáveis, x_i representa a *i-ésima* variável de entrada da função e $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos do problema. A função *F* representa um problema de minimização, não apresentando nenhuma restrição.

A Figura 45 mostra o Pareto Ótimo dessa função de teste, no caso de termos apenas duas variáveis de decisão: *x* e *y*.

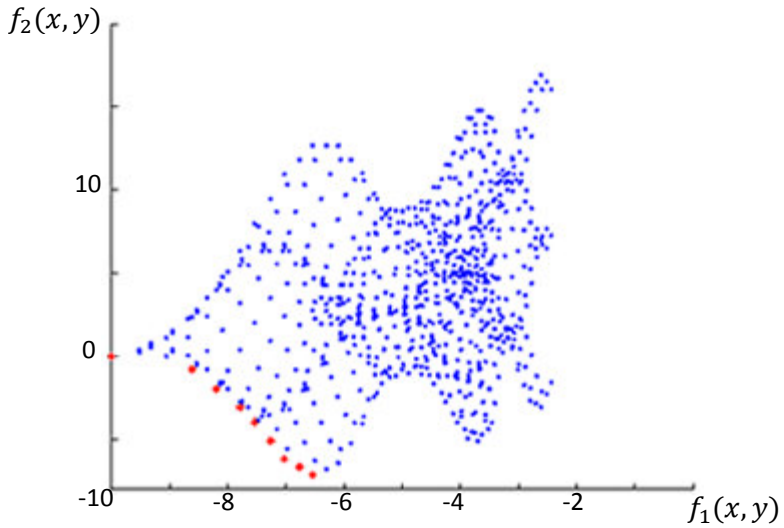

Figura 45. Pareto Front Ótimo da função de teste Kursawe.

Laumanns

A função de teste Laumanns foi originalmente proposta em [44] e é representada pela equação (52), onde

$$f_1(x,y) = x^2 + y^2, \quad (99)$$

$$f_2(x,y) = (x+2)^2 + y^2, \quad (100)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pela equação

$$-50 \leq x,y \leq 50. \quad (101)$$

Lis

A função de teste Lis foi originalmente proposta em [45] e é representada pela equação (52), onde

$$f_1(x,y) = \sqrt[8]{x^2 + y^2}, \tag{102}$$

$$f_2(x,y) = \sqrt[4]{(x-0.5)^2 + (y-0.5)^2}, \tag{103}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pela seguinte equação:

$$-5 \leq x, y \leq 10 \tag{22}$$

A Figura 46 mostra o Pareto Ótimo dessa função de teste.

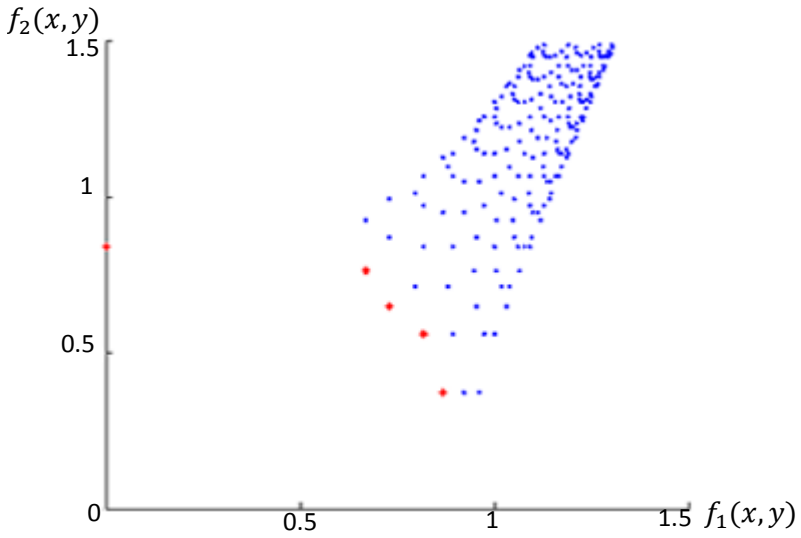

Figura 46. Pareto Ótimo da função de teste Lis.

Murata

A função de teste Murata foi originalmente proposta em [46] e é representada pela equação (52), onde

$$f_1(x,y) = 2\sqrt{x}, \tag{104}$$

$$f_2(x,y) = x(1-y) + 5, \tag{105}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pela equação

$$1 \leq x \leq 4, 1 \leq y \leq 2. \quad (106)$$

A Figura 47 mostra o Pareto Ótimo dessa função de teste.

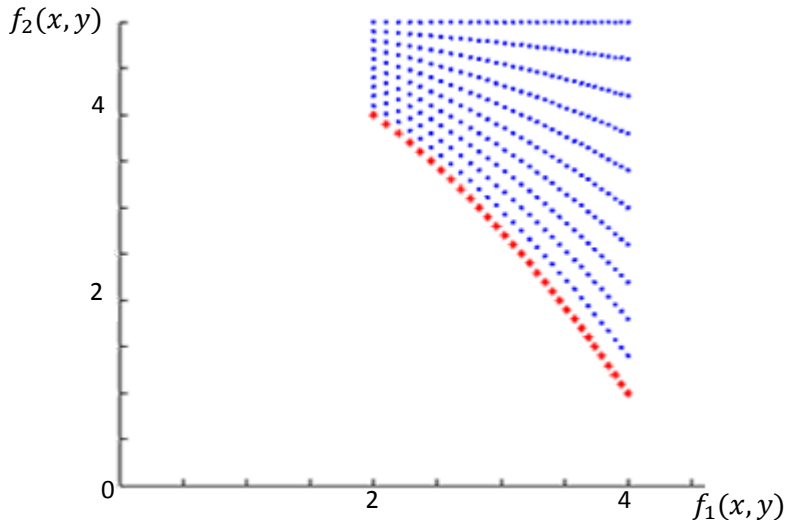

Figura 47. Pareto Ótimo da função de teste Murata.

Obayashi

A função de teste Obayashi foi originalmente proposta em [47] e é representada pela equação (52), onde

$$f_1(x,y) = x, \quad (107)$$

$$f_2(x,y) = y, \quad (108)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de maximização e seu domínio é restringido pelas equações

$$0 \leq x,y \leq 1, \quad (109)$$

$$x^2 + y^2 \leq 1. \quad (110)$$

A Figura 48 apresenta o Pareto Ótimo dessa função de teste.

Osyczka (1)

A função de teste Osyczka (1) foi originalmente proposta em [48] e é representada pela equação (52), onde

$$f_1(x,y) = x + y^2, \quad (111)$$

$$f_2(x,y) = x^2 + y, \quad (112)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pelas equações

$$2 \leq x \leq 7,5 \leq y \leq 10, \quad (113)$$

$$0 \leq 12 - x - y, \quad (114)$$

$$0 \leq x^2 + 10x - y^2 + 16y - 80. \quad (115)$$

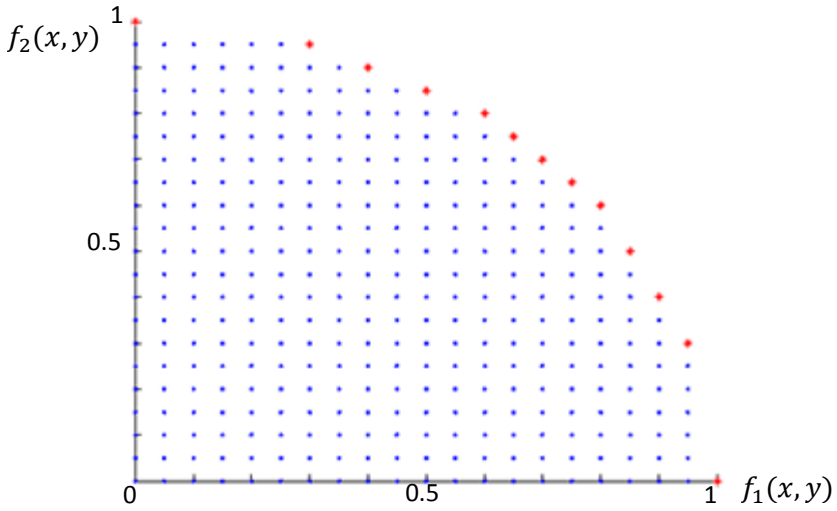

Figura 48. Pareto Ótimo da função de teste Obayashi.

Osyczka (2)

A função de teste Osyczka (2), assim como a Osyczka (1), foi originalmente proposta em [48] e é representada pela equação (79), onde

$$f_1(\vec{x}) = -(25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2)^2 + (x_4 - 4)^2 + (x_5\_1)^2, \quad (116)$$

$$f_2(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2, \quad (117)$$

onde $\vec{x}$ representa um vetor de entrada com *n* variáveis, x_1, x_2, x_3, x_4, x_5 e x_6 representam variáveis de entrada da função e $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos do problema. A função *F* representa um problema de minimização, apresentando as restrições

$$0 \leq x_1, x_2, x_6 \leq 10, \quad (118)$$

$$1 \leq x_3, x_5 \leq 5, \quad (119)$$

$$0 \leq x_4 \leq 6, \quad (120)$$

$$0 \leq x_1 + x_2 - 2, \quad (121)$$

$$0 \leq 6 - x_1 - x_2, \quad (122)$$

$$0 \leq 2 - x_2 + x_1, \quad (123)$$

$$0 \leq 2 - x_1 + 3x_2, \quad (124)$$

$$0 \leq 4 - (x_3 - 3)^2 - x_4, \quad (125)$$

$$0 \leq (x_5 - 3)^2 + x_6 - 4. \quad (126)$$

A Figura 49 apresenta o Pareto Ótimo dessa função de teste, no caso de termos apenas duas variáveis de decisão, *x* e *y*.

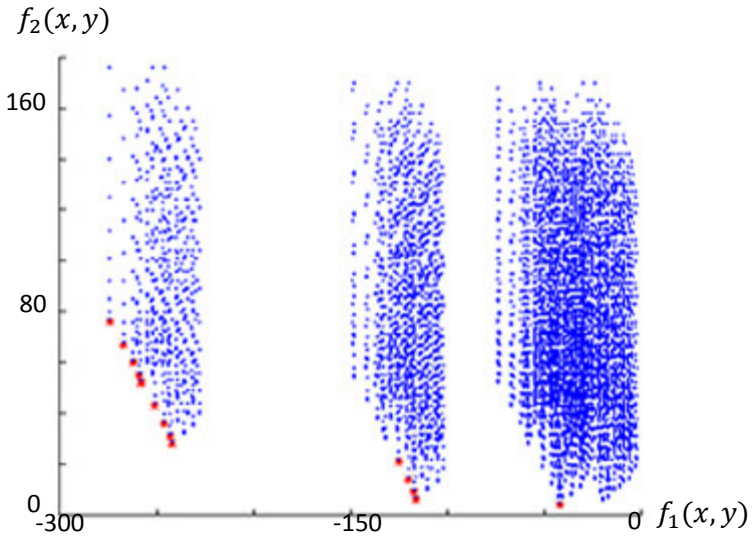

Figura 49. Pareto Ótimo da função de teste Osyczka (2).

Poloni

A função de teste Poloni foi originalmente proposta em [16] e é representada pela equação (52), onde

$$f_1(x,y) = -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2], \quad (127)$$

$$f_2(x,y) = -[(x+3)^2 + (y+1)^2], \quad (128)$$

sendo que

$$A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2, \quad (129)$$

$$A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2, \quad (130)$$

$$B_1 = 0.5\sin x - 2\cos x + \sin y - 1.5\cos y, \quad (131)$$

$$B_2 = 1.5\sin x - \cos x + 2\sin y - 0.5\cos y, \quad (132)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de maximização e seu domínio é restringido pela equação

$$-\pi \leq x, y \leq \pi. \quad (133)$$

A Figura 50 mostra o Pareto Ótimo dessa função de teste.

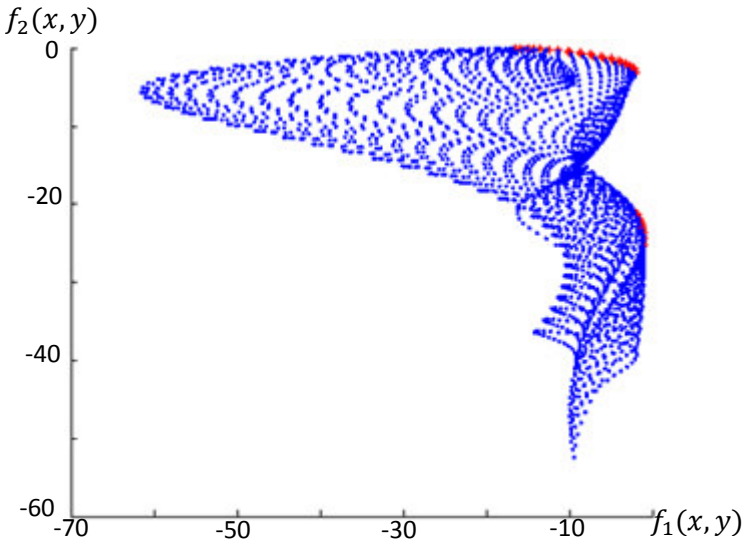

Figura 50. Pareto Ótimo da função de teste Poloni.

Quagliarella

A função de teste Quagliarella foi originalmente proposta em [49] e é representada pela equação (79), onde

$$f_1(\vec{x}) = \sqrt{\frac{A_1}{n}}, \tag{134}$$

$$f_2(\vec{x}) = \sqrt{\frac{A_2}{n}}, \tag{135}$$

sendo que

$$A_1 = \sum_{i=1}^{n} [(x_i)^2 - 10\cos[2\pi(x_i)] + 10], \tag{136}$$

$$A_2 = \sum_{i=1}^{n} [(x_i - 1.5)^2 - 10\cos[2\pi(x_i - 1.5)] + 10], \tag{137}$$

onde $\vec{x}$ representa um vetor de entrada com n variáveis, x_i representa a *i-ésima* variável de entrada da função e $f_1(\vec{x})$ e $f_2(\vec{x})$ representam os objetivos do problema. A função F representa um problema de minimização, apresentando a restrição

$$-5.12 \leq x_i \leq 5.12, n = 16. \tag{138}$$

A Figura 51 mostra o Pareto Ótimo dessa função de teste, no caso de termos apenas duas variáveis de decisão: x e y.

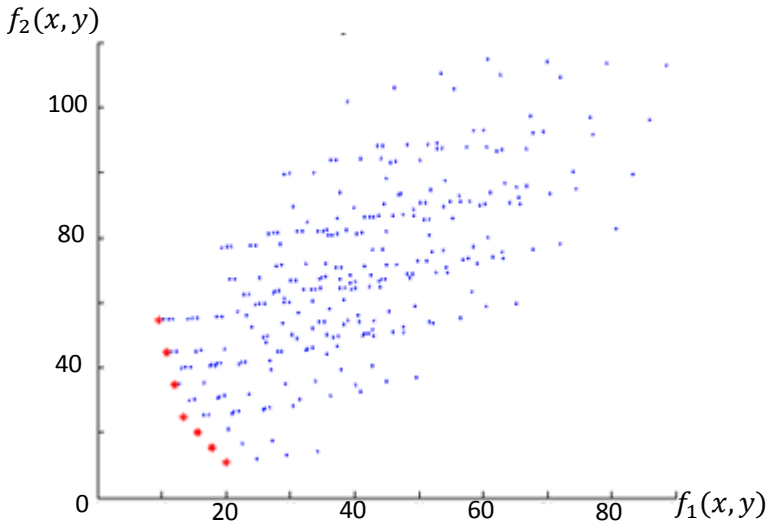

Figura 51. Pareto Ótimo da função de teste Quagliarella.

Rendon(1)

A função de teste Rendon(1) foi originalmente proposta em [50] e é representada pela equação (52), onde

$$f_1(x,y) = \frac{1}{x^2 + y^2 + 1}, \tag{139}$$

$$f_2(x,y) = x^2 + 3y^2 + 1, \tag{140}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido pela equação

$$-3 \leq x, y \leq 3. \tag{141}$$

A Figura 52 mostra o Pareto Ótimo dessa função de teste.

Rendon(2)

A função de teste Rendon(2) foi originalmente proposta em [50] e é representada pela equação (52), onde

$$f_1(x,y) = x + y + 1, \tag{142}$$

$$f_2(x,y) = x^2 + 2y - 1, \tag{143}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema.

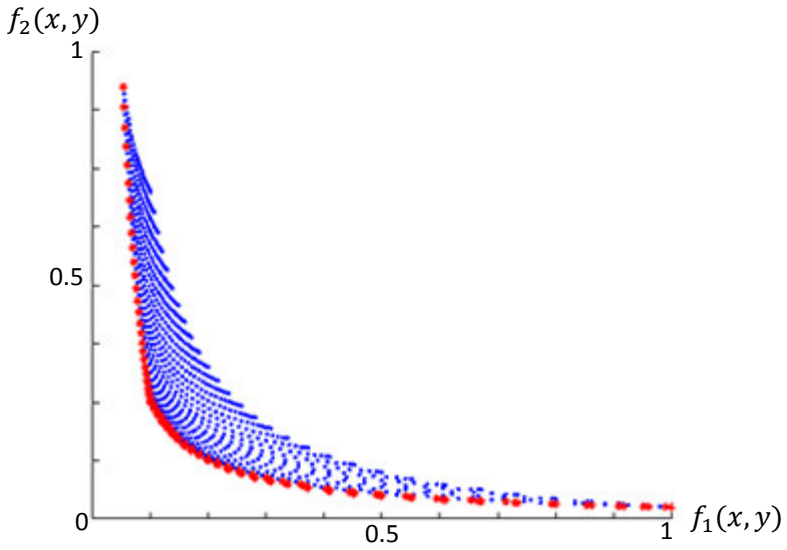

Figura 52. Pareto Ótimo da função de teste Rendon (1).

A função F representa um problema de minimização e seu domínio é restringido pela equação

$$-3 \leq x, y \leq 3. \tag{144}$$

A **Figura 53** mostra o Pareto Ótimo dessa função de teste.

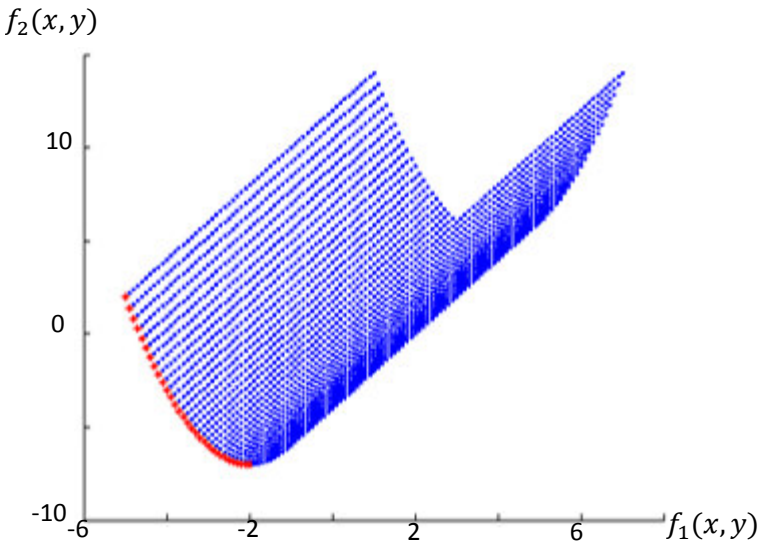

Figura 53. Pareto Ótimo da função de teste Rendon(2).

Schaffer(1)

A função de teste Schaffer(1) foi originalmente proposta em [51] e é representada por

$$F = (f_1(x), f_2(x)), \tag{145}$$

onde,

$$f_1(x) = x^2, \tag{146}$$

$$f_2(x) = (x-2)^2, \tag{147}$$

onde *x* representa a variável de decisão e $f_1(x)$ e $f_2(x)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio não possui nenhuma restrição. A Figura 54 apresenta o Pareto Ótimo dessa função de teste.

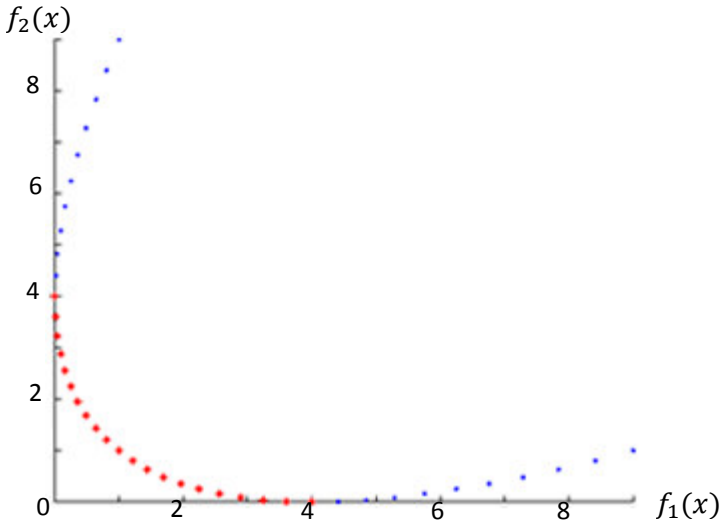

Figura 54. Pareto Ótimo da função de teste Schaffer(1).

Schaffer (2)

A função de teste Schaffer (2) é apresentado em [10] e é composta pela equação (145), onde

$$f_1(x) = \begin{cases} -x, & se\ x \leq 1 \\ -2 + x, & se\ 1 < x < 3 \\ 4 + x, & se\ 3 < x \leq 4 \\ -4 + x, & se\ x > 4 \end{cases}, \tag{148}$$

$$f_2(x) = (x-5)^2 \tag{149}$$

onde *x* representa a variável de decisão e $f_1(x)$ e $f_2(x)$ representam os objetivos do problema. A função *F* representa um problema de minimização, onde é realizado o mapeamento de uma variável de decisão para dois objetivos. O domínio da função é restringido por

$$-5 \leq x \leq 10. \tag{150}$$

A Figura 55 apresenta o Pareto Ótimo dessa função de teste.

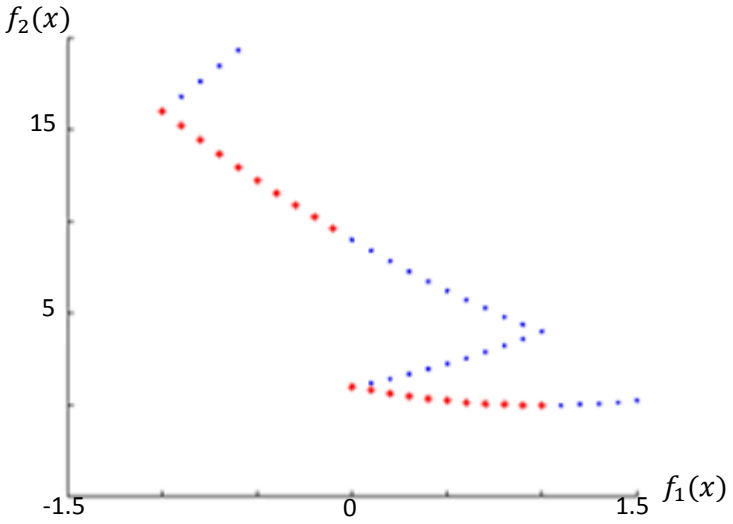

Figura 55. Pareto Ótimo da função de teste Schaffer (2).

Srinivas

A função de teste Srinivas foi originalmente proposta em [52] e é representada pela equação (52), onde:

$$f_1(x,y) = (x-2)^2 + (y-1)^2 + 2, \quad (151)$$

$$f_2(x,y) = 9x - (y-1)^2, \quad (152)$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido por

$$-20 \leq x, y \leq 20, \quad (153)$$

$$0 \geq x^2 + y^2 - 225, \quad (154)$$

$$0 \geq x - 3y + 10. \quad (155)$$

A

Figura 56 apresenta o Pareto Ótimo dessa função de teste.

Tamaki

A função de teste Tamaki foi originalmente proposta em [53] e é representada por

$$F = (f_1(x,y,z), (f_2(x,y,z), (f_3(x,y,z)), \quad (156)$$

onde,

$$f_1(x,y,z) = x, \quad (157)$$

$$f_2(x,y,z) = y, \quad (158)$$

$$f_3(x,y,z) = x, \quad (159)$$

onde *x*, *y* e *z* representam as variáveis de decisão e $f_1(x,y,z)$, $f_2(x,y,z)$ *e* $f_3(x,y,z)$ representam os objetivos do problema.

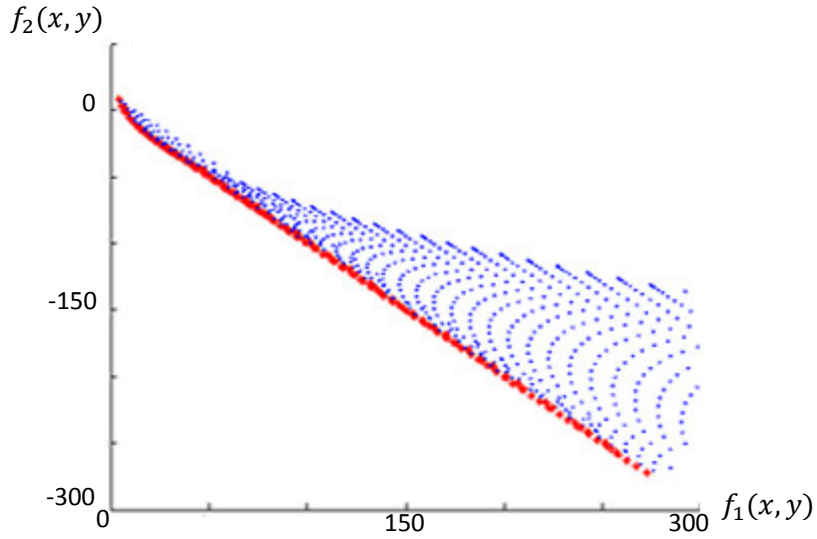

Figura 56. Pareto Ótimo da função de teste Srinivas.

A função *F* representa um problema de minimização que mapeia três variáveis de decisão para três objetivos. O domínio dessa função é restringido por

$$0 \leq x, y, z, \tag{160}$$

$$x^2 + y^2 + z^2 \leq 1. \tag{161}$$

Tanaka

A função de teste Tanaka foi originalmente proposta em [54] e é representada pela equação (52), onde

$$f_1(x,y) = x, \tag{162}$$

$$f_2(x,y) = y. \tag{163}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$ e $f_2(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização e seu domínio é restringido por

$$0 < x, y \leq \pi, \tag{164}$$

$$0 \geq -(x)^2 - (y^2) + 1 + 0.1 * \cos\left(16 \arctan\frac{x}{y}\right), \tag{165}$$

$$\frac{1}{2} \geq (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2. \tag{166}$$

A Figura 57 apresenta o Pareto Ótimo dessa função de teste.

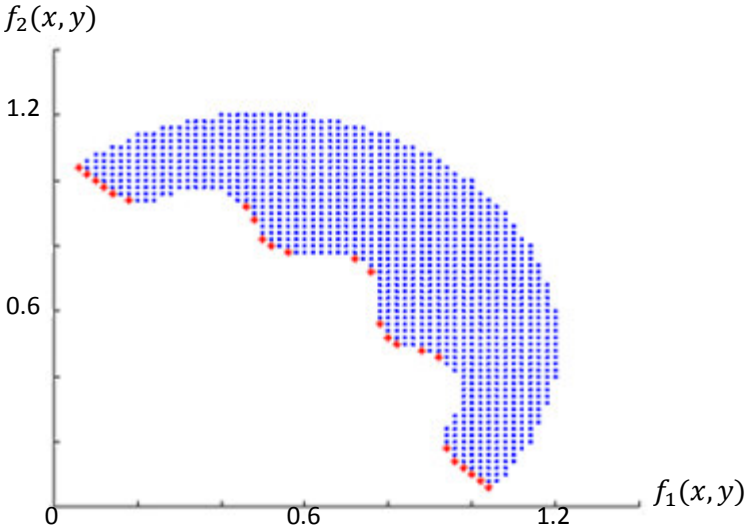

Figura 57. Pareto Ótimo da função de teste Tanaka.

Viennet (1)

A função de teste Viennet (1) foi originalmente proposta em [55] e é representada pela equação (66), onde

$$f_1(x,y) = x^2 + (y-1)^2, \tag{167}$$

$$f_2(x,y) = x^2 + (y+1)^2 + 1, \tag{168}$$

$$f_3(x,y) = (x-1)^2 + y^2 + 2, \tag{169}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$, $f_2(x,y)$ e $f_3(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, que mapeia duas variáveis de decisão para três objetivos. Seu domínio é restringido por

$$-2 \leq x, y \leq 2. \tag{170}$$

Viennet (2)

A função de teste Viennet (2), assim como a Viennet (1), foi originalmente proposta em [55] e é representada pela equação (66), onde

$$f_1(x,y) = \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3, \tag{171}$$

$$f_2(x,y) = \frac{(x+y-3)^2}{36} + \frac{(-x+y+2)^2 - 17,}{8} \tag{172}$$

$$f_3(x,y) = \frac{(x+2y-1)^2}{175} + \frac{(2y-x)^2}{17} - 13, \tag{173}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$, $f_2(x,y)$ e $f_3(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, que mapeia duas variáveis de decisão para três objetivos. Seu domínio é restringido por

$$-4 \leq x, y \leq 4. \tag{174}$$

Viennet (3)

A função de teste Viennet (3), assim como a Viennet (1) e Viennet (2), foi originalmente proposta em [55] e é representada pela equação (66), onde

$$f_1(x,y) = 0.5 * (x^2 + y^2) + \sin(x^2 + y^2), \tag{175}$$

$$f_2(x,y) = \frac{(3x - 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15, \tag{176}$$

$$f_3(x,y) = \frac{1}{(x^2 + x^2 + 1)} - 1.1e^{(-x^2 - y^2)}. \tag{177}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$, $f_2(x,y)$ e $f_3(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, que mapeia duas variáveis de decisão para três objetivos. Seu domínio é restringido pela por

$$-3 \leq x, y \leq 3 \tag{178}$$

Viennet (4)

A função de teste Viennet (4), foi originalmente proposta em [55] e é representada pela equação (66), onde

$$f_1(x,y) = \frac{(x - 2)^2}{2} + \frac{(y + 1)^2}{13} + 3, \tag{179}$$

$$f_2(x,y) = \frac{(x + y - 3)^2}{175} + \frac{(2y - x)^2}{17} - 13, \tag{180}$$

$$f_3(x,y) = \frac{(3x - 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15, \tag{181}$$

onde *x*, *y* representam as variáveis de decisão e $f_1(x,y)$, $f_2(x,y)$ e $f_3(x,y)$ representam os objetivos do problema. A função *F* representa um problema de minimização, que mapeia duas variáveis de decisão para três objetivos. Seu domínio é restringido por

$$-4 \leq x, y \leq 4, \tag{182}$$

$$y < -4x + 4, \tag{183}$$

$$x > -1, \tag{184}$$

$$y > x - 2. \tag{185}$$