OPERADOR DE PESQUISA LOCAL BASEADA EM APROXIMAÇÃO QUADRÁTICA PARA PROBLEMAS DE OTIMIZAÇÃO CONTÍNUA

Universidade Federal de Ouro Preto
Programa de Pós-Graduação em Ciência da Computação

Operador de Pesquisa Local Baseada em Aproximação Quadrática Para Problemas de Otimização Contínua

Felipe de Oliveira Mota

Defesa de dissertação de mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto, como requisito para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Gladston Juliano Prates Moreira

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES/FAPEMIG

Ouro Preto, 20 de Julho de 2018

M856o Mota, Felipe de Oliveira.
Operador de pesquisa local baseada em aproximação quadrática para problemas de otimização contínua [manuscrito] / Felipe de Oliveira Mota. - 2018.
70f.: il.: color; grafs; tabs.

Orientador: Prof. Dr. Gladston Juliano Prates Moreira.

Dissertação (Mestrado) - Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Otimização combinatória. 2. Ferramentas de busca na Web. 3. Algoritmos de computador. I. Moreira, Gladston Juliano Prates. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.4\'416

Catalogação: www.sisbin.ufop.br

Ata da Defesa Pública de Dissertação de Mestrado

Aos 13 dias do mês de agosto de 2018, às 14 horas na Sala de Seminários do DECOM no Instituto de Ciências Exatas e Biológicas (ICEB), reuniram-se os membros da banca examinadora composta pelos professores: **Prof. Dr. Gladston Juliano Prates Moreira (presidente e orientador), Prof. Dr. André Rodrigues Cruz, Profa. Dra. Elizabeth Fialho Wanner e Prof. Dr. Thiago Fontes Santos,** aprovada pelo Colegiado do Programa de Pós-Graduação em Ciência da Computação, a fim de arguirem o mestrando **Felipe de Oliveira Mota**, com o título **"Operadores de Pesquisa Local Baseada em Aproximação Quadrática para Problemas de Otimização Contínua com Restrições de Igualdade"**. Aberta a sessão pelo presidente, coube ao candidato, na forma regimental, expor o tema de sua dissertação, dentro do tempo regulamentar, sendo em seguida questionado pelos membros da banca examinadora, tendo dado as explicações que foram necessárias.

Recomendações da Banca:
(X) Aprovada sem recomendações
( ) Reprovada
( ) Aprovada com recomendações: _____

Banca Examinadora:

Prof. Dr. Gladston Juliano Prates Moreira

Prof. Dr. André Rodrigues Cruz

Profa. Dra. Elizabeth Fialho Wanner

Prof. Dr. Thiago Fontes Santos

Prof. Dr. Joubert de Castro Lima
Coordenador do Programa de Pós-Graduação em Ciência da Computação
DECOM/ICEB/UFOP

Ouro Preto, 13 de agosto de 2018.

Dedicatória à todas as pessoas que, de alguma forma, ajudaram no meu processo de formação como pessoa.
Dedication to all people that, in some way, helped me in my process of formation as a person.

Resumo

Este documento traz o estudo de um operador de busca local que utiliza aproximações quadráticas para formar um algoritmo hibrido e resolver problemas multiobjetivo ou mono objetivo com restrições de igualdade. Muitas vezes, algoritmos evolutivos como o PSO (Eberhart & Kennedy, 1995) conseguem encontrar boas bacias de atração em problemas de otimização, mas explorá-las pode ser complicado. Por isso uma hibridização com potencial para encontrar rapidamente mínimos locais é uma opção amplamente utilizada para acelerar a convergência e melhorar a precisão do processo.

Ao longo da sua execução, os algoritmos evolutivos movem seus pontos de maneira que eles avancem às regiões com as melhores soluções. Os operadores utilizados, chamados aqui de Full-Matrix Quadratic Approximation (FMQA) e Diagonal Quadratic Approximation (DQA), utilizarão pontos das boas regiões encontradas no espaço para gerar funções quadráticas que aproximam as funções originais do problema. Eles diferem apenas em como as matrizes são construídas. Este modelo aproximado pode ser facilmente resolvido, obtendo uma solução que atenderá o problema original e é provavelmente melhor do que os pontos utilizados para fazer tal construção.

O objetivo do trabalho é testar a união destes operadores com o algoritmo evolutivo *Particle Swarm Optimization* (PSO), melhorando seus indivíduos separadamente para resolver problemas com restrições de igualdade. Nós queremos observar suas vantagens e desvantagens quanto a tempo computacional e precisão, quando aplicada aos problemas propostos. Nestes problemas, a dimensão reduzida do espaço de busca torna difícil o trabalho do algoritmo evolutivo puro, e esse operador se mostrou eficiente para auxiliar na busca. Também será estudado como os operadores performam em problemas

multiobjetivo.

Palavras-chave: Aproximações quadráticas, Otimização, Algoritmos Evolutivos Híbridos, Busca Local.

Abstract

This document presents a study proposal of a local search operator using quadratic approximations to build a hybrid algorithm and solve multi-objective problems or single objetive ones with equality constraints. Oftentimes, evolutive algorithms like PSO (Eberhart & Kennedy, 1995) find good attraction basins in optimization problems, but explore them can be complicated. Hence, a hybridization with a potential to quickly find local minima is a widely used option to speed up the convergence and improve the precision of the process.

During its execution, evolutive algorithms constantly moves their points so that they advance towards the regions with the best solutions. The used operators, named here as Full-Matrix Quadratic Approximation (FMQA) and Diagonal Quadratic Approximation (DQA), use points from this already good known regions from the variables space to build a model with only quadratic functions that approximate the original problem functions. They differ only by how the second order terms matrixes are built: in DQA, we have a matriz with only zeros except for the diagonal, that is filled with positive values; on the other hand, FMQA has a complete matrix, with no restrictions. This approximated model can be easily solved, obtaining a solution that will attend the constraints of the original problem and is probably better than the points used to make that construction.

The objective of this work is to test the union of these operators with the evolutionary algorithm *Particle Swarm Optimization* (PSO), improving the individuals separately. We want to see their advantages and disadvantages on time cost and precision, when applied to the proposed problems. In these problems, the reduced dimension in the search space or the higher number

of objectives create difficulties to the pure genetic algorithm's job, and those operators had shown efficiency at helping the search. It will be also studied how the operators perform in multi-objective problems.

Keywords: Quadratic aproximations, Optimization, Hybrid Evolutive Algorithms, Local Search.

Sumário

Lista de Acrônimos

VNS - *Variable-Neighborhood Search*

RBGA -*Real-biased Genetic Algorithm*

MOGA -*Multi-Objective Genetic Algorithm*

PSO -*Particle Swarm Optimization*

CPO - Conjunto Pareto-Ótimo

FPO - Fronteira Pareto-Ótima

CQA -*Constraint Quadratic Approximation*

SOP -*Single-Objective Problem*

MOP -*Multi-Objective Problem*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A otimização é uma das áreas mais estudadas da computação, pela sua utilidade em empresas e organizações financeiras. Vários de seus problemas correspondem à situações do mundo real em que uma melhora na solução pode gerar, por exemplo, redução de custo, tempo, ou algum outro tipo de vantagem. Inicialmente, os modelos de tais problemas retratavam apenas situações com apenas um fator de interesse, trabalhando com funções-objetivo escalares, nos problemas ditos mono-objetivos (SOP's - Single-Objective Problems). Naturalmente, existem casos em que é necessário considerar um conjunto de dois ou mais critérios para, gerando os problemas de otimização multiobjetivo (MOP's - Multi-Objective Problems), em que a função-objetivo é vetorial. É mandatório que estes critérios sejam conflitantes entre si, e neste caso não existe uma única solução que possa ser considerada melhor. Diante disto, as soluções interessantes são aquelas que, quando comparadas com qualquer outra solução, serão mais atrativas em algum critério, e menos atrativas em outros. Estas soluções são chamadas soluções eficientes e, juntas, constituem o chamado *Conjunto Pareto-ótimo* (CPO). Encontrá-lo significa encontrar a resposta do problema.

Muitos problemas da área ainda não possuem uma resposta definitiva e heurísticas têm sido uma das formas de resolução mais estudadas (Coello, 2006). Dentre estas, Algoritmos Evolucionários (EA's) representam uma grande parte dos estudos no assunto (Coello, 2002; Van Veldhuizen

& Lamont, 2000). Tais métodos são usualmente probabilísticos, e inspirados em algum fenômeno observado na natureza. Para citar apenas algumas aplicações, o método aparece em engenharia aeronáutica (Chiba *et al.* , 2005), engenharia estrutural (Greiner *et al.* , 2005), engenharia elétrica (Rivas-Dávalos & Irving, 2005), escalonamento de processos (Hanne & Nickel, 2005) e medicina (Lahanas, 2004). Assim surge a necessidade de que estes algoritmos fiquem cada vez mais precisos, rápidos e robustos.

Dada que a estrutura estocástica dos MOEA's não garante totalmente que as soluções Pareto-ótimas sejam encontradas, uma das suas possíveis modificações é adicionar um operador determinístico no processo. As chamadas hibridizações têm variados procedimentos e aplicações (Goel & Deb, 2002; Deb & Goel, 2001; Côté *et al.* , 2004; Konstantinidis & Yang, 2011). Um bom exemplo é a hibridização sugerida por (Wanner *et al.* , 2005), que desenvolveu um operador de busca local quadrático para restrições (CQA) e mostrou grandes resultados em problemas com restrições de igualdade não-lineares. Com estes dados em mente, é comum pensar que, aproximando funções, possamos criar operadores poderosos que auxiliem os algoritmos evolutivos no processo de busca. Porém, aproximar funções por si só é um vasto campo de estudos, abordado recorrentemente e sendo difícil dizer um campo científico em que não seja utilizada.

Os problemas de tais aproximações podem ser divididos em dois principais grupos. O primeiro é a teoria de aproximação, que procura maneiras de aproximar funções complexas por outras mais simples e fáceis de manipular (Carothers, 1998). O segundo grupo de problemas, que será abordado aqui, surge quando a expressão analítica da função não é explicitada, e uma função aproximada que associe domínio e imagem minimizando o erro pode ser mais útil na prática do que o processo custoso de recuperar a função original. Na estatística, por exemplo, existe o interesse de aproximar as caudas de algumas distribuições (Mumford & Shah, 1989). No ramo de visão computacional, algumas funções precisam ser aproximadas por funções localmente suaves, como por exemplo a intensidade de luz num ponto do plano (Reid, 1996).

Uma instância do problema consiste em um conjunto de pares de pontos

$C = \{(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), ..., (\mathbf{x_n}, \mathbf{y_n})\}$, em que $\{\mathbf{x_i}\}_1^s$ são pontos no domínio (ou de entrada), enquanto $\{\mathbf{y_i}\}_1^s$ são pontos na imagem (ou de saída) da função $f(\mathbf{x})$ desconhecida. Escolhida uma medida de erro $L(\mathbf{x})$, o objetivo é encontrar $g(\mathbf{x})$ tal que minimize esta medida, ou seja:

$$\varepsilon = arg\min_{g} \sum_i^n L(\mathbf{y_i}, g(\mathbf{x_i}))$$

Algumas formas comuns de medir o erro são as funções de erro quadrático $(\mathbf{y} - g(\mathbf{x}))^2$, erro absoluto $|\mathbf{y} - g(\mathbf{x})|$ e log binomial probabilístico $log(1 + e^{-2\mathbf{y}g(\mathbf{x})})$ (Friedman, 2001).

Quando fazemos uma aproximação de alguma função desconhecida, buscamos uma função aproximada que, além de produzir erros pequenos em relação ao conjunto de imagens fornecido, seja simples e tenha propriedades que facilitem o seu processamento. Assim, é comum esperar que tal função seja polinomial, exponencial, trigonométrica ou uma combinação destas. Este trabalho é focado na aproximação utilizando funções quadráticas.

A forma geral de uma função quadrática é dada por:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}x_i x_j + \sum_{k=1}^{n} b_k x_k + c \ , \tag{1.1}$$

em que ao menos um dos $a_{i,j}$ é não-nulo (Jankovic, 2005). Para se obter uma aproximação dessa forma pode-se, por exemplo, utilizar conjuntamente os métodos de Newton para encontrar os zeros da função e o método de Cauchy em seguida, para aproximar a série de Taylor da função por uma fração racional (Shafer, 1974), ou aplicar o algoritmo de *Lower Bound* (LB) (Böhning & Lindsay, 1998).

Esta aproximação é eficaz quando utilizada em conjunto com algoritmos evolutivos, melhorando a convergência destes. Ela auxilia guiando o processo em regiões de difícil exploração, como regiões de volume zero, em que a dimensão é inferior ao contra-domínio, e tratando problemas com variados tipos de restrições (Wanner, 2006). Esta estratégia já foi testada previamente em algoritmos genéticos (Deep & Das, 2008; Fonseca *et al.* , 2012), *Bilevel Evolutionary Algorithm based on Quadratic Approximations* (BLEAQ) (Sinha *et al.* , 2013) e *Self-organizing Migration Algorithm with*

Quadratic Approximation (SOMAQI) (Singh & Agrawal, 2015). O objetivo estudado pelo autor é aprimorar o operador quando utilizado para resolver SOP's com restrições de igualdade e MOP's irrestritos.

1.1 Estado da Arte

É importante destacar aqui alguns trabalhos feitos anteriormente, com avanços significantes para a área de estudos. Serão citados metodologias que atacam o mesmo tipo de problema, usam aproximação quadrática como método para buscas locais, ou para tentar melhorar o PSO de outra maneira.

1.1.1 Aproximação Quadráticas de Restrições

O operador *Constraint Quadratic Approximation* (CQA) foi sugerido inicialmente em (Wanner *et al.* , 2005), onde foi acoplado ao RBGA. Este operador utiliza um ponto $\mathbf{x}$ e pontos na vizinhança $V(\mathbf{x})$ já obtidos no algoritmo genético (GA) para formar um problema quadrático aproximado, sob a condição de que o número de pontos disponível é suficiente para se obter a aproximação. Resolve-se então o problema aproximado com métodos monoobjetivo, e o ponto encontrado substituirá o pior indivíduo da população original.

No documento citado, a hibridização foi testada apenas em alguns problemas analíticos. Em (Wanner, 2006), o operador foi acoplado ao MOGA e ao SPEA2 para problemas multiobjetivo com funções objetivo e restrições não lineares. Testes com até 3 restrições de desigualdade resultaram em uma melhora significativa quando comparado com o algoritmo evolutivo sem o operador (Wanner *et al.* , 2008), e também foi utilizado em SOP's com duas (Fonseca & Wanner, 2016) ou mais restrições de igualdade (Peconick *et al.* , 2007).

1.1.2 Aproximação da diagonal

É possível que uma matriz com apenas elementos na sua diagonal principal funcione como uma aproximação da Hessiana, e possamos usá-la na

função quadrática aproximadora da busca local. Essa é a ideia exposta em (da Cruz *et al.* , 2011), que utiliza um otimizador linear para minimizar o erro de norma-1 e obter a aproximação com pontos da população principal. Este método possui, entre outras vantagens, um número reduzido de icógnitas e robustidade com relação a outliers. As ideias deste artigo foram utilizadas aqui, com a diferença de utilizar o PSO como algoritmo principal.

1.1.3 Outros operadores para o PSO

Em um outro trabalho, foi estudado como auxiliar o PSO com uma poderosa ferramenta de busca local: o *Variable Neighborhood Search* (VNS) (Mota *et al.* , 2018). Utilizando o melhor ponto como centro, vizinhanças crescentes eram pesquisadas em busca de um ponto melhor que o central. O trabalho atacou SOP's com restrições de igualdade, foi comparado com o PSO puro, além do operador de busca local BFGS (Luenberger, 2003) e mostrou-se mais preciso que os dois outros métodos na busca pela solução ótima do problema.

Uma outra abordagem com semelhanças na base da aproximação também pode ser vista em (Deep & Bansal, 2009), na qual é anexado ao PSO um operador baseado em aproximações quadráticas. No caso, separam a população em dois enxames, e melhoram um encontrando o valor de mínimo da superfície quadrática formada a partir de 3 pontos.

1.2 Objetivos

O trabalho teve como objetivo principal avaliar a melhora de desempenho em relação à tempo, precisão e diversidade (esta última apenas no caso multiobjetivo) do PSO quando munido dos operadores FMQA e DQA utilizados. Assim, ficará evidente o potencial das hibridizações de buscar boas soluções em problemas contínuos com restrição de igualdade. Outros objetivos traçados foram:

- Computar o tempo gasto por cada operador, e analisar ganho de eficiência *versus* custo de tempo nos SOP's restritos.

- Avaliar do modelo proposto em MOP's, levando em consideração a diversidade.
- Observar o quanto cada restrição é violada no fim da rotina, calculando o maior valor absoluto obtido pelos melhores pontos encontrados nas restrições do problema.
- Concluir qual dos dois operadores é melhor para os problemas analisados, e se possível indicar quando cada um é mais vantajoso.

1.3 Organização do texto

Após esse capítulo introdutório, a dissertação se organiza desta maneira: O Capítulo 2 traz o conhecimento considerado imprescindível para melhor compreensão da metodologia e dos resultados. O Capítulo 3 exibe a metodologia escolhida para se atacar o problema. No Capítulo 4 é explicado como foram feitos os teste escolhidos e apontado qual foi o desempenho alcançado no trabalho. Concluímos o trabalho e mostramos as direções a serem seguidas no futuro no Capítulo 5.

Capítulo 2

Fundamentação Teórica

2.1 Otimização

Otimização é um ramo da matemática computacional que basicamente procura encontrar mínimos e máximos de uma função, podendo haver ou não restrições. A formulação geral pode ser escrita como:

$$\begin{aligned} \mathbf{x}^* = \min_{\mathbf{x} \in D} \quad & f(\mathbf{x}) \\ \text{sujeito à : } & \begin{cases} g_i(\mathbf{x}) \leq 0, \ i = 1, ..., l \\ h_j(\mathbf{x}) = 0, \ j = l, ..., m \end{cases} \end{aligned} \tag{2.1}$$

O ponto $\mathbf{x}^*$ precisa ser encontrado, f é chamada função-objetivo, D é o domínio da função, e g_i e h_j restringem o espaço de busca $\forall i, j$. É válido notar que um problema de maximização pode ser transformado em minimização com uma simples troca de sinal na função-objetivo. De forma análoga, pode-se aplicar esta troca nas restrições para alterar o sinal de desigualdade. Cada ponto $\mathbf{x}$ pertencente ao domínio é um vetor de variáveis de decisão. Cada coordenada do vetor é independente e pode pertencer a diferentes conjuntos, como os reais, imaginários, binário, inteiros, dentre outros.

2.1.1 Otimização Mono-Objetivo

A otimização se diz mono-objetivo quando a função-objetivo mapeia o conjunto das variáveis de decisão em um escalar, ou seja, $f : \mathbb{R}^n \mapsto \mathbb{R}$. Este tipo de problema é de muita aplicabilidade, já que muitas das vezes, modelos reais têm como fim melhorar um aspecto isolado, como minimizar o custo ou maximizar o lucro.

As técnicas de otimização mono-objetivo podem servir de base ou até trabalhar em conjunto com as técnicas multiobjetivo. Portanto é importante conhecer a ideia por trás de algumas delas. Como trabalharemos principalmente com heurísticas, é válido citar aqui métodos populacionais (GA, PSO, etc) e de exploração de vizinhanças (*Iterated Local Search* (ILS) (Lourenço *et al.* , 2003), Variable Neighborhood Search (VNS) (Mladenović & Hansen, 1997),etc).

2.1.2 Vizinhança

O conceito de vizinhança é diretamente associado ao conceito de distância. Dependendo da distância escolhida, podemos considerar pontos interiores ou apenas na borda, observar vizinhanças contínuas, discretas, circulares, entre outras formas. Por exemplo, em um problema com apenas variáveis binárias, podemos definir a distância entre duas soluções como o número de bits diferentes entre elas. E a partir desta distância, podemos definir uma vizinhança de um ponto como o conjunto de soluções que distam no máximo d dele. Muitos algoritmos utilizam este conceito, então é importante explicá-lo um pouco melhor.

Considere a família de operações $A = \{\Phi_i\}_1^k$ em que cada i-ésimo elemento aplica uma alteração diferente no ponto. Vamos definir a n-ésima vizinhança $V_d(\mathbf{x})$ como:

$$\begin{aligned} &y \in V_d(\mathbf{x}). \iff \mathbf{y} = \Phi_{q_1}(\Phi_{q_2}...(\Phi_{q_d}(\mathbf{x}))...) \\ &q_j \in \{1, 2, ..., k\} \ \forall \ j \text{ e } j \neq l \rightarrow i_j \neq i_l \end{aligned} \tag{2.2}$$

Em resumo, se o ponto $\mathbf{y}$ tem distância d do ponto $\mathbf{x}$, então ele foi alterado por exatas d operações da família A, e $V_d(\mathbf{x})$ é constituída de todos os pontos que distam d de $\mathbf{x}$.

2.1.3 Soluções ótimas

Existem dois tipos de pontos ótimos, o local e o global. O ponto de ótimo global é o ponto que gera o melhor valor (seja mínimo ou máximo) de função dentro de seu domínio D. O ótimo local só garante que não existem soluções melhores dentro de uma determinada vizinhança $V(\mathbf{x})$. Os dois podem ser vistos na Figura 2.1.

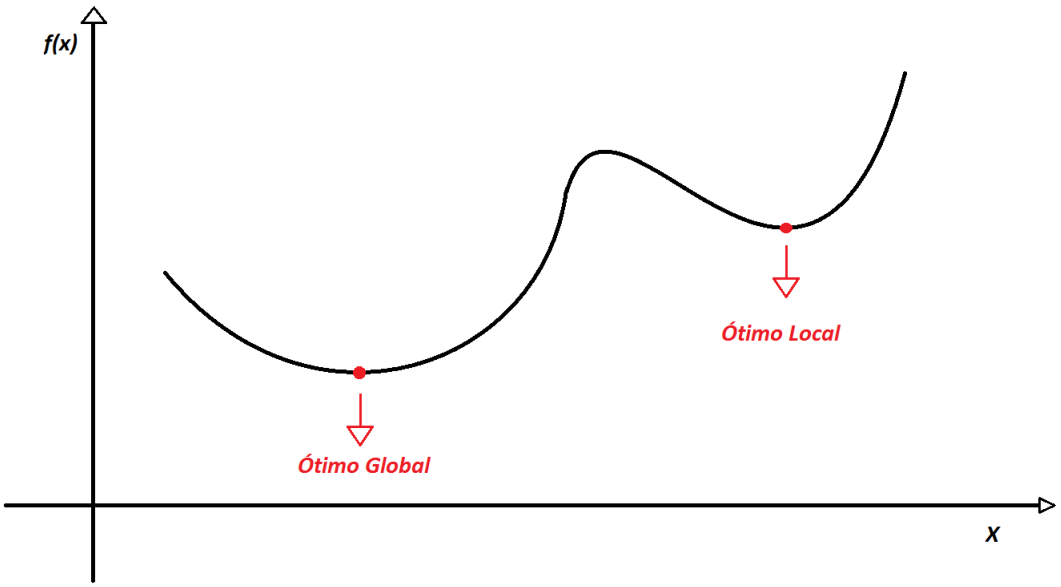

Figura 2.1: Ótimo local e ótimo global.

A definição formal destes dois objetos em um problema de minimização é:

$$\mathbf{x} \text{ é ótimo global} \iff \nexists\, \mathbf{y} \in D \mid f(\mathbf{x}) > f(y)$$
$$\mathbf{x} \text{ é ótimo local} \iff \nexists\, \mathbf{y} \in V(\mathbf{x}) \mid f(\mathbf{x}) > f(\mathbf{y})$$

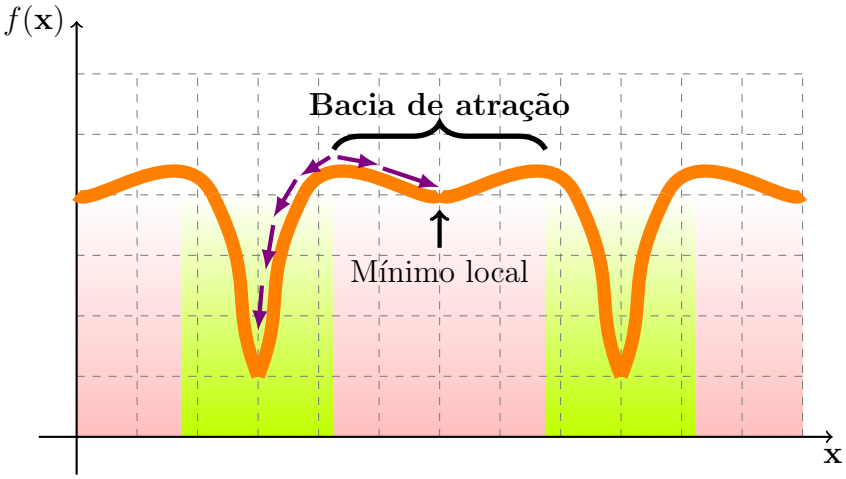

Figura 2.2: Bacias de atração separadas por cor (Rutten, 2011).

2.1.4 Bacias de atração

Visualizar uma bacia de atração é muito mais simples que defini-la. Em um gráfico de minimização, uma bacia de atração pode ser vista nas cavidades com apenas um "fundo", como na Figura 2.2.

Uma definição mais rigorosa exige os conceitos de conexidade e unimodalidade. Uma região C é dita conexa quando qualquer par de pontos $\mathbf{x}, \mathbf{y} \in C$ pode ser unido por uma curva contínua. Já uma função f é unimodal quando dado qualquer $\alpha \in \mathbb{R}$, os pontos do domínio de f tal que $f(\mathbf{x}) \leq \alpha$ definem uma região conexa. Posto isso, dado um ponto $\mathbf{x_0}$, a bacia de atração deste ponto é a maior região conexa C contendo $\mathbf{x_0}$ tal que $f\mid_C$ (lê-se f restrita à C) é unimodal.

2.1.5 Otimização Multiobjetivo

A definição de um MOP é bem semelhante ao que foi apresentado na Equação 2.1, com uma mudança no contra-domínio: $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^m$. Problemas deste tipo são vistos em várias situações reais, quando se tem dois ou mais objetivos conflitantes. Como exemplo, pode-se supor que, em uma determinada tarefa, quer-se reduzir simultaneamente tempo e custo, em que estes dois objetivos são inversamente proporcionais de alguma forma.

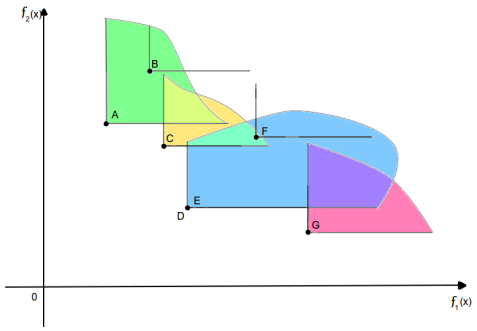

Figura 2.3: Imagem das funções em um problema com dois objetivos.

Esta mudança na dimensão da imagem trás diferenças gigantescas para o problema, pois é um espaço parcialmente ordenado. Isto quer dizer que existem soluções distintas tais que é difícil escolher a melhor delas, olhando apenas para os valores de função. Porém, nos casos em que é possível distinguir quando uma solução é melhor que as outras, precisamos eliminar as piores. E para isso é necessário entender o conceito de relação de dominância.

Relação de dominância e soluções eficientes

Dada uma função $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})) : \mathbb{R}^n \mapsto \mathbb{R}^m$ a ser minimizada, definiremos:

$$\mathbf{f}(\mathbf{a}) \prec \mathbf{f}(\mathbf{b}) \iff (\exists\ i \mid f_i(\mathbf{a}) < f_i(\mathbf{b})) \wedge (f_j(\mathbf{a}) \leq f_j(\mathbf{b})\ \forall\ j = 1, ..., n)$$

Aqui, o símbolo $\prec$ pode ser lido como "domina". Quando uma solução a domina outra solução b, a primeira tem pelo menos um valor de função-objetivo melhor que a segunda, enquanto os outros são menores ou iguais. Na Figura 2.3 podemos ver algumas relações de dominância entre os pontos no espaço dos objetivos: A, C, D, E e G são não dominados; B é dominado apenas por A; F é dominado por C, D e E. As regiões coloridas são parte do domínio dos 5 pontos não-dominados.

Uma solução eficiente é aquela que não é dominada por qualquer outra solução. Nenhuma solução eficiente pode ser dita, à priori, melhor ou pior que outras soluções eficientes. É necessário que seja feita uma análise dos

dados obtidos, e escolher as melhores soluções de acordo com o escopo do estudo.

Conjunto Pareto-Ótimo e Fronteira Pareto-Ótima

O conceito de Pareto-Otimalidade foi introduzido por Vilfredo Pareto (Ehrgott, 2012). É um conceito econômico, e diz que se uma certa alocação de recursos é Pareto-Ótima, então não há como alterá-la para melhorar um critério sem piorar algum outro. Ainda na Figura 2.3, o conjunto-Pareto dentre os pontos fornecidos seria A, C, D, E e G.

Em computação, este conceito foi adaptado para problemas de otimização multiobjetivo. No espaço das variáveis, a união de todas as soluções eficientes constitui o CPO, e a fronteira Pareto-Ótima (FPO) é a imagem deste conjunto pela função pré-definida (Van Veldhuizen & Lamont, 1998). Dada uma solução da fronteira e escolhido um dos objetivos separadamente, não se pode escolher outra solução que gere ganho neste objetivo sem perder nos outros.

2.2 Indicadores de Desempenho

Diferentemente dos SOP's, comparar soluções de MOP's que não tem relação de dominância não é trivial. Além disso, podem haver infinitas soluções no CPO, e dois algoritmos distintos ou até duas execuções de um mesmo algoritmo não-determinístico podem ter resultados bem diferentes, e ainda assim serem equivalentes em eficiência. Assim, quando se tratam de MOP's, comparar dois métodos tem dois importantes fatores a se avaliar: o quão perto chegaram da FPO (precisão); e se suas soluções estão bem distribuídas no espaço (diversidade) (Knowles & Corne, 2002).

2.2.1 *Spacing*

Criada em (Schott, 1995), esta medida diz o quão bem distribuídos no espaço dos objetivos estão as soluções eficientes retornadas pelo algoritmo (diversidade do algoritmo). Ela avalia a variância entre as distâncias de cada

ponto para a solução eficiente mais próxima. Pode ser calculada pela seguinte fórmula:

$$
\begin{aligned}
f_{spacing} &= \sqrt{\frac{1}{e-1}\sum_{i=1}^{e}(d_i - \overline{d})^2} \\
\text{em que } d_i &= \min_j \sum_{k=1}^{m} |f_k^i - f_k^j| \\
\text{e } \overline{d} &= \frac{1}{e}\sum_{i=1}^{e} d_i
\end{aligned}
\tag{2.3}
$$

Em 2.3, que f_k^i é o valor da k-ésima função-objetivo no i-ésimo ponto, d_i é a distância para o vizinho mais próximo deste ponto, e $\overline{d}$ é a média destas distâncias. Quanto menor for esta medida, melhor distribuídos estão as soluções eficientes, e se ela for nula, então eles se encontram igualmente distribuídos no espaço.

2.2.2 Hipervolume

Esta medida bem intuitiva foi proposta em (Zitzler, 1999), e avalia tanto a precisão (ou o quanto as soluções avançaram em direção à FPO) quanto a diversidade do algoritmo. Ela mede, basicamente, a área dominada pelas soluções retornadas no algoritmo. Ela é independente de escala, mas requer um ponto fixo de referência, normalmente um limite superior para cada função-objetivo. A partir deste ponto, é calculada a área do hipervolume entre ele e os pontos fornecidos pelo resolvedor, e quanto maior for este valor, mais eficiente é o algoritmo. Um exemplo pode ser visto na Figura 2.4.

2.3 Algoritmos Evolutivos

Nas seções anteriores, foi visto que a natureza dos MOP's faz com que seja importante encontrar muitas soluções distintas. Algoritmos evolutivos são heurísticas que trabalham com um conjunto de soluções (população) explorando o espaço de busca simultaneamente, abrindo a possibilidade de

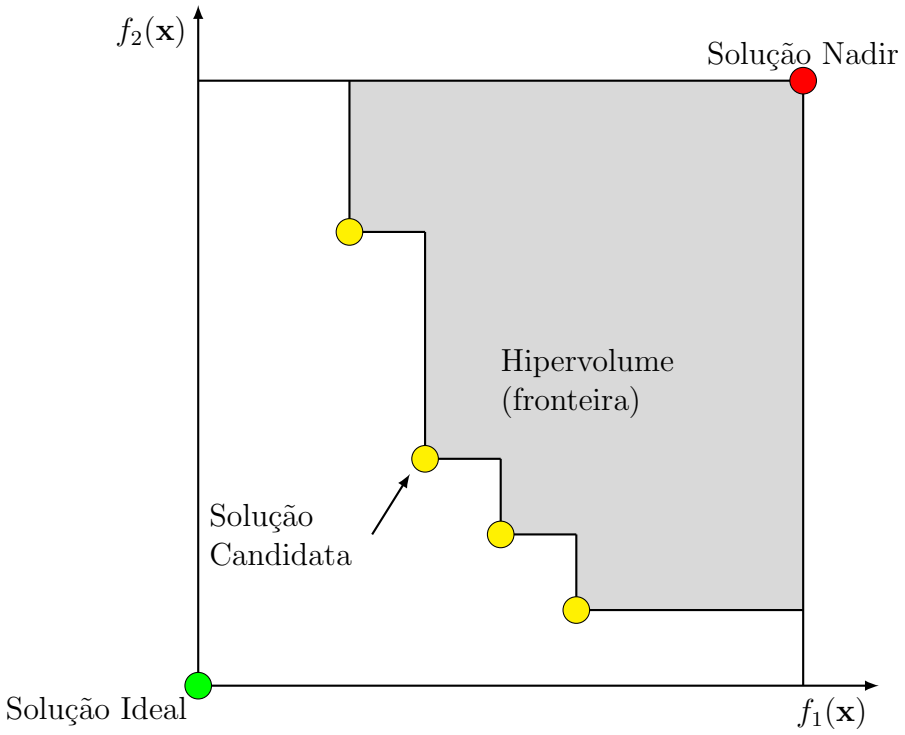

Figura 2.4: Hipervolume produzido a partir de 4 soluções eficientes e uma solução Nadir(com os piores valores de cada função-objetivo) (Adra *et al.* , 2009)

.

se encontrar várias soluções eficientes na mesma execução. A capacidade destes algoritmos de lidar com problemas tão complexos, faz com que esta seja uma forte opção no ataque à tais problemas (Fonseca & Fleming, 1995).

A modelagem do MOEA tem dois objetivos principais: representar as variáveis de entrada (decisão) em indivíduos e de saída (objetivos) em sua aptidão. A primeira representação dá forma aos cromossomos, que são os conjuntos de informação de cada indivíduo. É usual que seja utilizado um vetor para isto, onde cada posição representa um valor de variável. Já a segunda representação é mais complexa, uma vez que é necessário criar uma função de avaliação, que tem como finalidade comparar e distinguir melhores e piores indivíduos, algo que nem sempre é fácil em um MOP (Coello *et al.* , 2007)

Modelado o problema, o processo restante dos algoritmos evolutivos têm uma estrutura semelhante para resolver o problema. Soluções iniciais são criadas sob algum critério (aleatoriedade, por exemplo), e, até um critério de parada ser atendido, elas são avaliadas segundo uma função pré-definida, combinadas e modificadas com o intuito de tentar direcioná-las para a região com as melhores soluções.

Ao final da execução, pode-se calcular medidas de desempenho sobre os resultados do algoritmo para avaliá-lo. As medidas mais comuns são de precisão e diversidade. Se os resultados não forem satisfatórios, pode ser feita uma mudança nos parâmetros, operadores, ou, em casos extremos, na modelagem ou mesmo no algoritmo.

2.3.1 PSO

Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) é um algoritmo evolutivo que melhora sua população baseando-se na cooperação, ao contrário de algoritmos como o genético, que se utiliza de competição. Ele foi criado inspirado nos movimentos de conjuntos de animais, como cardumes de peixes e bandos de gaivotas.

O PSO tem por base suas partículas, que inicialmente recebem alguma posição e velocidade para iniciarem a busca. A partir disto, elas começam a

se movimentar, baseando-se na combinação de informações obtidas por ela mesma, pelo resto das partículas e por fatores aleatórios. Fazendo com que a tendência da velocidade das partículas seja direcionar-se para onde estão as melhores informações, é esperado que a população evolua após repetidos movimentos de cada uma delas.

Esta ideia foi explorada em inúmeras aplicações (Poli, 2008), comumente obtendo resultados positivos. Porém, alguns estudos mostraram dificuldades e limitações no método, relacionadas à convergência de vários tipos e a invariância por transformações (Bonyadi & Michalewicz, 2017).

2.3.2 Operadores

Operadores são agentes que podem inserir, alterar ou retirar indivíduos da população. São, na maioria das vezes, estocásticos, de modo que apenas o uso repetido destes garantirá o retorno de soluções melhores ao fim do processo. Três operadores básicos aparecem nos algoritmos em geral: seleção, recombinação e mutação.

A seleção é o processo que tem a função de eliminar os piores indivíduos e manter os melhores a cada geração, aproximando a população do CPO. Se as melhores soluções obrigatoriamente passarem pela seleção, o método é dito também elitista. Uma descrição de vários métodos com diferentes critérios pode ser encontrada em (Hancock, 1994).

Na recombinação, informação de dois ou mais indivíduos é misturada na tentativa de se obter indivíduos que carreguem o melhor de seus precedentes. No espaço de busca, a solução obtida estará numa região entre as originais, fazendo com que este operador seja responsável pela exploração da região delimitada pelos indivíduos extremos da população.

A mutação é o operador que modifica um único indivíduo, reposicionando-o no espaço de busca. Isto garante que a população não fique presa apenas nas mesmas regiões, já que explorá-las poderia encontrar apenas ótimos locais distantes do global.

2.3.3 Hibridização

Apesar das várias vantagens que os algoritmos evolutivos têm à oferecer, eventualmente haverão casos em que a heurística populacional não irá convergir satisfatoriamente. Para garantir resultados mais precisos, é possível combinar os procedimentos destes algoritmos com outras estratégias de otimização. Este processo é conhecido como hibridização.

Uma das opções de hibridização, por exemplo, é inserir procedimentos determinísticos no algoritmo. Procurando nas bacias de atração com processos exatos de busca local, pode-se garantir que mínimos locais sejam encontrados, apesar do processo ser custoso. Utilizando heurísticas na hibridização pode ser rápido, mas ineficiente dependendo de como forem gerados os pontos. São precisos muitos testes para dizer quais e quantas hibridizações serão positivas para auxiliar o algoritmo evolutivo.

2.3.4 Busca ou Pesquisa Local

A busca local é um método de otimização que procura melhorar uma solução atual $\mathbf{x}$ comparando-a com soluções em uma vizinhança $V(\mathbf{x})$ com critério pré-definido. Se houver uma solução $\mathbf{y} \in V(\mathbf{x})$ tal que $f(\mathbf{x}) > f(\mathbf{y})$, então $\mathbf{y}$ assume o lugar de $\mathbf{x}$ como solução atual, e o processo continua (Pirlot, 1996). Eles são muito úteis quando se quer trabalhar num espaço unimodal do problema, encontrando mínimos locais.

Como nos piores casos o processo é exaustivo, em grandes dimensões ele pode se tornar lento e custoso, e examinar muitas vizinhanças se torna inviável. Assim é interessante aplicá-lo em pequenas sub-regiões, enquanto outros algoritmos exploram o domínio por inteiro.

2.3.5 Algoritmos Meméticos

Ao longo da vida, um ser vivo ganha conhecimento antes de se reproduzir. Baseando-se nessa ideia, alguns indivíduos da população podem ser melhorados por meio de busca local antes de avançar e fornecer suas informações. Esse princípio dá origem aos chamados algoritmos meméticos (Radcliffe &

Surry, 1994). Sendo uma hibridização, esse tipo de algoritmo depende muito da performance da busca local, e isto motiva o estudo e desenvolvimento do operador citado em (Wanner *et al.* , 2008).

2.4 Matriz Hessiana

Seja $f : \mathbb{R}^n \mapsto \mathbb{R}$ duas vezes diferenciável. A matriz Hessiana de f no ponto $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ é dada por:

$$H(f(\mathbf{x})) = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_2}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Na matriz em questão, $\frac{\partial f}{\partial x_i}$ é a derivada parcial de f em relação à x_i. Em resumo, é constituída de todas as derivadas parciais de segunda ordem de f, e pode ser utilizada para aproximar funções quadraticamente na vizinhança do ponto escolhido.

2.5 Convexidade

Aqui definiremos a convexidade de funções e conjuntos.

2.5.1 Conjunto Convexo

Um conjunto D é dito convexo se:

$$\mathbf{x_1}, \mathbf{x_2} \in D, \lambda \in (0, 1) \implies \lambda \mathbf{x_1} + (1 - \lambda)\mathbf{x_2} = \mathbf{y} \in D$$

Mais claramente, se o conjunto é convexo, dado dois pontos nele contidos, qualquer ponto pertencente ao segmento de reta que liga os dois também está no conjunto, como o conjunto verde da Figura 2.5.1.

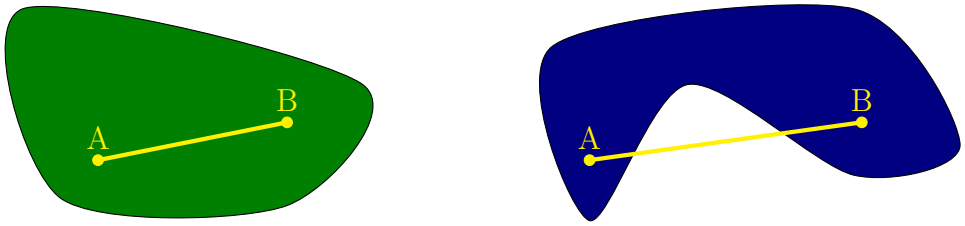

Figura 2.5: Conjunto convexo em verde e não-convexo em azul. Apenas no primeiro conjunto temos quaisquer segmentos de reta ligando dois pontos do conjunto inteiramente contidas nele.

2.5.2 Função Convexa

Seja $D \subset \mathbb{R}^n$ um domínio convexo e a função $f : D \to \mathbb{R}$. Então, dado $\mathbf{x_1}, \mathbf{x_2} \in D$ e $\lambda \in [0, 1]$, temos que:

$$f \text{ é convexa } \iff f(\lambda \mathbf{x_1} + (1 - \lambda)\mathbf{x_2}) \leq \lambda f(\mathbf{x_1}) + (1 - \lambda) f(\mathbf{x_2})$$

A fórmula acima afirma apenas que uma avaliação de f em qualquer combinação linear entre $\mathbf{x_1}$ e $\mathbf{x_2}$ resultará em um valor de função menor ou igual do que a mesma combinação de $f(\mathbf{x_1})$ e $f(\mathbf{x_2})$. A Figura 2.6 mostra a forma de uma função convexa.

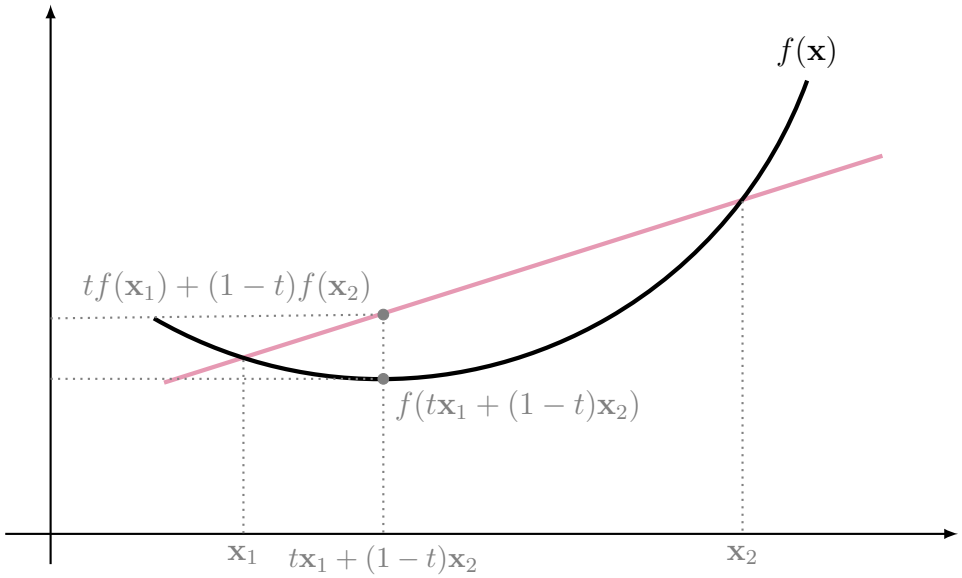

Figura 2.6: Exemplo de função convexa (Freiberger, 2011).

Capítulo 3

Metodologia

3.1 Operadores quadráticos utilizados

O processo de melhorar a população através de busca local tem duas fases principais: a construção da função aproximada e a pesquisa local (Wanner *et al.* , 2008). Suponha agora que tenhamos um problema da forma:

$$\begin{aligned} &\mathbf{x}^* = \min f(\mathbf{x}) \\ &\text{sujeito a } g_i(\mathbf{x}) = 0, i = 1, 2, ..., m \end{aligned} \tag{3.1}$$

Nosso objetivo começa por achar funções f^A e g_i^A que aproximem f, g_i, $i \in 1, 2, .., m$ e sejam da forma:

$$\begin{aligned} f_i^A(\mathbf{x}) &= 0.5\mathbf{x}' H_0 \mathbf{x} + r_0' \mathbf{x} + \gamma_0 \\ g_i^A(\mathbf{x}) &= 0.5\mathbf{x}' H_i \mathbf{x} + r_i' \mathbf{x} + \gamma_i, \ i = 1, 2, ..., m \end{aligned} \tag{3.2}$$

Dado que $\mathbf{x} \in \mathbb{R}^n$ (e n considerado o mesmo para o restante deste trabalho), H_i são matrizes $n \times n$, r e f_i vetores $n \times 1$, e γ_i constantes.

As matrizes H_i possuem o maior grau na equação, e assim determinam o comportamento assintótico e representam as matrizes Hessianas da função aproximada. Propusemos construí-las de duas formas: uma preenchendo-a completamente, com nenhuma restrição quanto à simetria ou concavidade; e

outra fazendo-a uma matriz diagonal apenas com termos positivos, forçando a convexidade da função.

3.1.1 Método 1: Matriz completa

Seja f a função a ser aproximada e $\mathbf{x^a}$, $a \in \{1, 2, ..., s\}$ o conjunto de pontos no espaço das variáveis, cujos respectivos valores $f(\mathbf{x^a}) = y_a$ são conhecidos. Se queremos obter funções do tipo 3.2, desenvolvendo termo a termo, podemos montar o sistema de equações:

$$\begin{gathered}
\sum_{i=1}^{n}\sum_{j=1}^{n} 0.5h_{ij}x_i^1x_j^1 + \sum_{k=1}^{n} r_k x_k^1 + \gamma - f(x^1) = 0 \\
\sum_{i=1}^{n}\sum_{j=1}^{n} 0.5h_{ij}x_i^2x_j^2 + \sum_{k=1}^{n} r_k x_k^2 + \gamma - f(x^2) = 0 \\
\vdots \\
\sum_{i=1}^{n}\sum_{j=1}^{n} 0.5h_{ij}x_i^sx_j^s + \sum_{k=1}^{n} r_k x_k^s + \gamma - f(x^s) = 0
\end{gathered}$$

Temos que h_{ij} representa o termo na i-ésima linha e j-ésima coluna da matriz H, e r_k o k-ésimo elemento de r. Temos então $N = n^2 + n + 1$ icógnitas para serem encontradas, ignorando o fato de que a matriz deveria ser simétrica, por comutatividade do produto. Isso acarreta num custo computacional maior, já que teremos de encontrar $(n^2 - n)/2$ variáveis a mais, mas dá mais liberdade ao resolvedor para distribuir o valor entre 2 posições na matriz e adiciona equações e pontos para melhor determinar a quadrática. Como o sistema é linear, é possível que existam infinitas soluções, quando o número de equações for inferior à N. A aproximação será mais precisa à medida que o número de equações se aproximar de N, e portanto este será o número de pontos passados ao algoritmo de aproximação. Para resolver o sistema, utilizamos o procedimento *fsolve* do Matlab, um resolvedor não linear que retorna valores encontrados para cada icógnita do sistema. Este método utiliza principalmente dois algoritmos para tal tarefa: "trust-region"(Coleman & Li, 1996) e "Levenberg-Marquadt"(Moré, 1978).

Na teoria, utilizar o operador aproximando a matriz inteira tem uma série de vantagens: o solucionador é rápido (mesmo com mais icógnitas) e considera-se a correlação entre as variáveis. Porém, a convexidade não é

garantida, podendo levar a problemas de convergência, já que a superfície pode apresentar inflexões.

3.1.2 Método 2: Matriz diagonal

Para obter a matriz diagonal (ou seja, apenas elementos em sua diagonal são não-nulos), utilizamos um processo diferente de apenas resolver um sistema. Primeiramente, iremos exigir a convexidade da função (que todos elementos não-nulos sejam positivos), para que a função tenha um mínimo no interior do domínio. Além disso, por vários motivos já citados e melhor descritos em (da Cruz *et al.* , 2011), a norma-1 é muito indicada para fazer aproximações desse tipo. Então, utilizando novamente a Equação 3.2 e desenvolvendo-as termo a termo, chegamos ao seguinte modelo:

$$\min \sum_{k=1}^{s} f(x^k) - \left(\frac{1}{2} \sum_{i=1}^{n} h_i x_i^k + \sum_{j=1}^{n} r_j x_j^k + \gamma \right) \tag{3.3}$$

$$\text{s.a. } \frac{1}{2} \sum_{i=1}^{n} h_i x_i^k + \sum_{j=1}^{n} r_j x_j^k + \gamma \leq f(x^k), \quad k = 1, 2, .., s \tag{3.4}$$

$$-h_i^k < 0 \ \forall \ k = 1, 2, ..., s \tag{3.5}$$

Com os mesmos índices do método anterior.

Temos então um problema de otimização linear contínua, com $N = 2n+1$ variáveis a serem encontradas, utilizando s pontos. Foi utilizada a função do Matlab *linprog* neste caso. Ela encontra os valores das variáveis através dos algoritmos "Dual-Simplex"e "Interior-Point-Legacy".

Essa abordagem garante o ponto de mínimo local no interior do espaço de busca, diferentemente do método anterior. Porém, o número bem reduzido de variáveis pode perder informação contida nos pontos, e o otimizador linear é bem mais lento, principalmente se a região onde se encontram os pontos for muito irregular.

3.1.3 O resultado da busca

Após obter todas as matrizes Hessianas, os termos lineares e as constantes, produzimos funções aproximadas de segundo grau. Queremos encontrar um ponto que atenda às aproximações das restrições (incluindo os limites de variável) e minimize a aproximação da função-objetivo. Para obter tal ponto, utilizamos a função *fmincon* do MatLab, passando as funções aproximadas e o ponto inicial da busca na origem.

Novamente, o resolvedor possui algumas opções de algoritmos a serem utilizados nessa procura, e selecionamos o "Interior-Point"para conseguir a solução. Terminado o procedimento do resolvedor, o ponto que a função retornar será aquele produzido pelo operador e que será utilizado pelo algoritmo principal.

3.2 O PSO como algoritmo principal

O procedimento feito pelo PSO pode ser dividido em algumas etapas básicas, como mostrado na Figura 3.2.

Podem haver variações destas etapas, visando corrigir os problemas e limitações mais frequentes no algoritmo. Possibilidades são citadas em (Bonyadi & Michalewicz, 2017). Nas próximas seções, mostraremos alguns detalhes sobre o PSO implementado no trabalho.

3.2.1 Inicialização

Dado um problema, como apresentado pela Equação 3.1, a população será representada por duas matrizes $P, V \in \mathbb{R}^{p \times n}$, sendo p o número de indivíduos e n o número de variáveis (ou dimensão do problema). A i-ésima linha delas representa, respectivamente, a posição e velocidade do indivíduo, sendo as colunas responsáveis por cada uma das variáveis do problema. Ambas as matrizes são inicializadas aleatoriamente, de maneira uniforme, com valores pertencentes aos limites de cada variável.

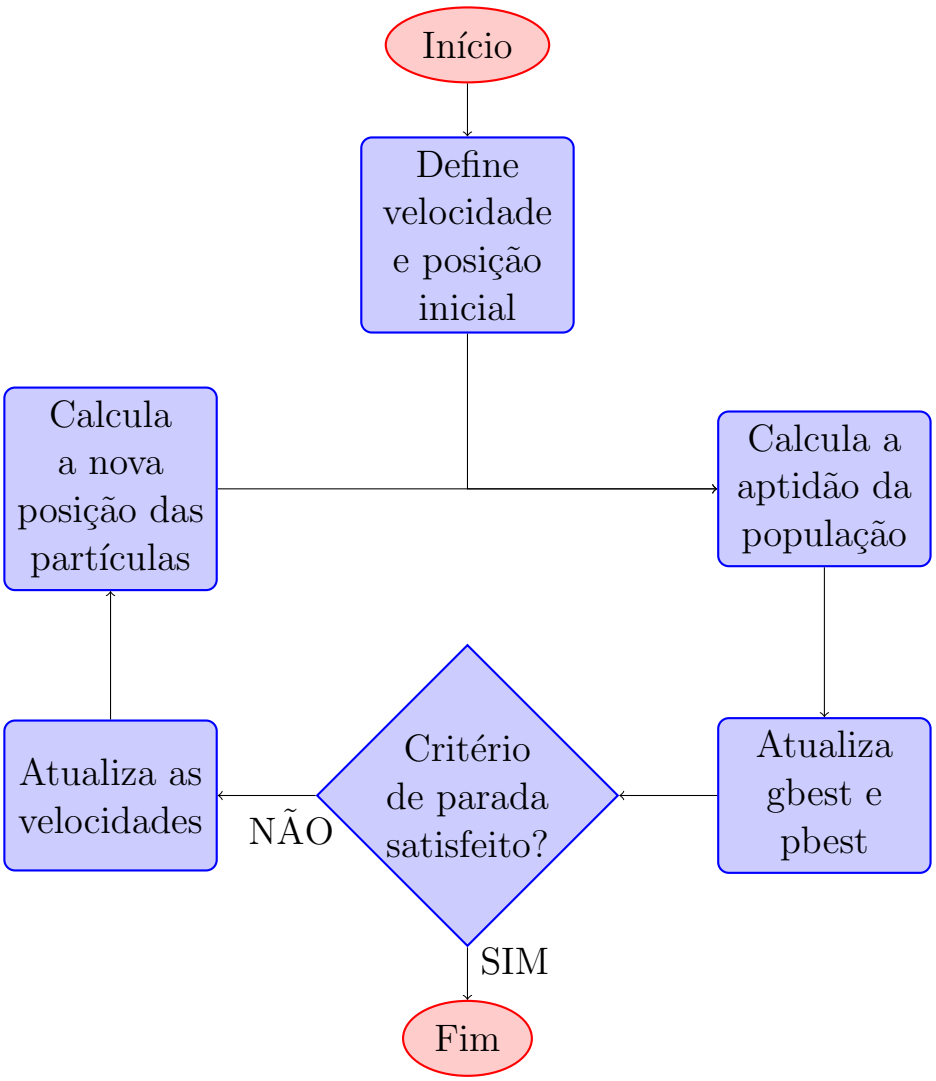

Figura 3.1: Procedimentos básicos do PSO.

3.2.2 Avaliação e atualização dos melhores valores

A função de aptidão avalia cada indivíduo baseando apenas na posição. Como se tratam de problemas restritos, qualquer violação da restrição levará a uma piora no valor $\alpha(\mathbf{x}_i)$ de aptidão do indivíduo. Este tipo de abordagem torna problemas restritos em "irrestritos"sem alterar a ênfase nos pontos factíveis. O tipo mais comum de penalização é o aditivo (Yeniay, 2005), e o utilizaremos neste trabalho. Em termos exatos, teremos:

$$\alpha(\mathbf{x}) = \begin{cases} f(\mathbf{x}) + \phi \sum_{i=1}^{m} |g_i(\mathbf{x})|^2 \ \forall \ i : |g_i(\mathbf{x})| > \epsilon \\ f(\mathbf{x}) \text{ se } |g_i(\mathbf{x})| \leq \epsilon, \ i = \{1...m\} \end{cases} \tag{3.6}$$

Em que ϕ é o fator de penalidade, ϵ é a tolerância, e $g_i(\mathbf{x})$ são as restrições do problema.

Após esta avaliação é possível que sejam atualizados o *pbest* e o *gbest*. O primeiro diz respeito à melhor posição já alcançada por cada partícula individualmente. O segundo é a melhor solução já encontrada pela população. Estes são os fatores responsáveis pela informação colaborativa e individual utilizado pelo sistema no processo de evolução, e são alterados toda vez que um valor de solução melhor é encontrado, local ou globalmente.

3.2.3 Atualização de velocidade

No PSO, a velocidade é o fator responsável por alterar a posição das partículas, uma vez a cada passo evolutivo. A nova velocidade é calculada considerando-se a velocidade antiga e os atuais *gbest* e *pbest*, como pode ser visto na Figura 3.2.

A velocidade é modificada segundo a seguinte fórmula:

$$\mathbf{v}_i^k(t) = \mathbf{w}(t)\mathbf{v}_i^k(t-1) + c_1\Gamma_{1i}(\mathbf{pbest}_i^k - \mathbf{x}_i^k(t-1)) + c_2\Gamma_{2i}(\mathbf{gbest}_i^k - \mathbf{x}_i^k(t-1))$$

Na qual temos que $\mathbf{v}_i^k(t)$ é a k-ésima componente da velocidade do i-ésimo indivíduo, no instante (ou geração) t, $\mathbf{x}_i^t(t)$ diz respeito à posição nas mesmas condições. Sugerido em (Shi & Eberhart, 1998), $\mathbf{w}(t)$ é o fator de inércia, que diz qual peso terá a velocidade anterior. Em contrapartida, c_1 e c_2 são

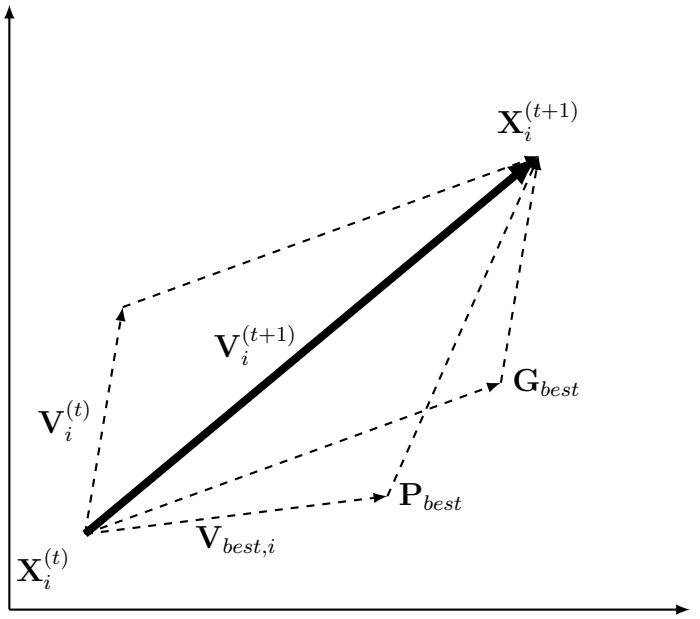

Figura 3.2: Vetores que têm influência no cálculo da nova velocidade (Moghaddam *et al.* , 2011).

fatores sociais, que indicam a tendência das partículas de seguirem o *pbest* ou o *gbest*. Finalmente, Γ_1 e Γ_2 são fatores de aleatoriedade, que podem dar variações positivas ao processo.

3.2.4 Atualização de posição e condição de parada

Com a velocidade atualizada, as partículas se movem simplesmente seguindo a fórmula:

$$x_i^k(t) = x_i^k(t-1) + v_i^k(t)$$

Seguindo esta fórmula sem qualquer tipo de prevenção, é possível que indivíduos saiam do intervalo previamente estabelecido para cada variável. Se isso ocorrer, podemos fazer uma reflexão deste indivíduo segundo o limite que foi violado. A reflexão é dada por:

$$x_r^k = x_L^k + |x^k - x_L^k| \text{ or } x_r^k = x_U^k - |x_U^k - x^k|$$

em que x_r é o resultado da reflexão, e x_L e x_U os limites inferiores e superiores das variáveis, respectivamente.

Quanto ao critério de parada, o único utilizado foi o número de gerações, que é equivalente ao número de avaliações de função, uma vez que este número não muda com o uso de operador ou entre gerações.

3.3 Processo de Hibridização

É esperado que a metodologia utilizada irá melhorar indivíduos isoladamente, à parte do processo evolutivo, entrando então na classe dos algoritmos meméticos. O operador será acoplado da seguinte maneira:

1. A cada cinco gerações, pega-se um determinado conjunto de N pontos da população que será passado ao operador para serem feitas as aproximações. Serão escolhidas e passadas como parâmetro as melhores soluções daquela geração. O tamanho escolhido do conjunto é o que favorece a aproximação mais precisa de cada operador:

    a - Na aproximação da matriz inteira, o sistema fica unicamente determinado com o número de equações igual ao de icógnitas, ou seja, $N = n^2 + n + 1$[1].

    b - Quando se quer apenas a diagonal da matriz, o otimizador pode se beneficiar de mais pontos (restrições). Resolvemos então seguir o que foi feito em (da Cruz *et al.* , 2011), e utilizar $N = 4n + 2$.

2. Se algum dos solucionadores não obtiver sucesso em aproximar alguma função ou resolver o problema aproximado, então o operador retornará um ponto aleatório à população. Assim, mesmo não melhorando a população, ainda aumentará a diversidade e pode evitar uma convergência prematura.

3. Após resolvido o problema aproximado, o ponto resultante é inserido na população, no lugar do pior indivíduo.

[1]No problema 11, seriam necessários 507 pontos para conseguir N, então foram utilizados apenas 200 pontos

Como o ponto substituído é o pior ponto da população, é esperado que, em média, este operador não piore o desempenho do PSO utilizado como base da otimização, mesmo que as os resolvedores falhem repetidamente.

Capítulo 4

Experimentos e resultados

4.1 Parâmetros

Enquanto as aproximações foram praticamente livre de parâmetros, o PSO foi calibrado empiricamente, e os valores encontrados que melhor se adaptavam aos problemas foram:

- População: 200[1].
- Gerações: 100.
- Fatores sociais da velocidade: $c_1 = c_2 = 2.05$.
- Fatores aleatórios: $0 < \Gamma_1, \Gamma_2 < 1$.
- Fator de inércia: $w = (0.1) * ((100 - nger)/100) + 0.2$, sendo $nger$ o número de gerações passadas até aqui.
- Fator de penalidade (SOP's): $\phi = 100$.[2]
- Tolerância (SOP's): $\epsilon = 10e - 4$.
- Total de avaliações de objetivo: 20.000 (30.000 para P11).

[1]*Exceto para o problema P11, que possui 22 variáveis. Nele temos população de 300 indivíduos.

[2]Exceto para o problema P4, em que $\phi = 10^6$.

4.2 Problemas Teste

Os problemas testados no trabalho vieram de algumas fontes diferentes. Dentre os SOP's encontramos os problemas P1, P2 e P3 em (Wanner *et al.* , 2005). Os dois primeiros são considerados os mais fáceis, por serem polinomiais, unimodais, com poucas varáveis, enquanto o terceiro é multimodal, mas não apresenta muito aumento na dificuldade devido às dimensões também reduzidas.

P6, P7 e P8 foram retirados de (Peconick *et al.* , 2007). Eles possuem características semelhantes aos 3 primeiros problemas, mas com um número superior de variáveis, aumentando o espaço de busca, e de restrições, definindo uma região factível difícil de explorar.

Os problemas P4, P5 e P10 correspondem, respectivamente aos problemas g02, g11 e g14 do benchmark do *Congress on Evolutionary Computation* (CEC) 2006 (Liang *et al.* , 2006). P9 e P11 são adaptações dos problemas g21 e g22 do mesmo congresso, com as desigualdades ativas tornadas igualdades.

Passando para os MOP's, temos os problemas propostos por Zitzler, Deb e Thiele (Zitzler *et al.* , 2000), ZDT 1, 2, 3 e 4, que são referenciados respectivamente como MP2-5. Já MP1 e MP6 foram retirados isoladamente de (Binh & Korn, 1996) e (Jin *et al.* , 2001), respectivamente. Todos eles possuem diferentes caraterísticas e níveis de dificuldades, melhores explicados em (Huband *et al.* , 2006), e foi definido que sua dimensão seria $p = 10$. Estes problemas não possuem restrições, mas ter mais de um objetivo é dificuldade suficiente para evidenciar a importância do operador.

A forma analítica de todos os problemas testados pode ser vista no Apêndice A.

4.3 Experimentos

Cada algoritmo para os problemas restritos mono-objetivo foi executado 30 vezes, como sugerido na literatura para convergência à média (Student, 1908), e os resultados podem ser vistos na Tabela 4.2, em que P = PSO puro, D = PSO com aproximação diagonal, e FM = PSO com aproximação cheia

Prob	T-P	T-D	T-FM	SP-P	SP-D	SP-FM	HV-P	HV-D	HV-FM
MP1	**41.20**	126.03	722.98	**<10e-05**	**<10e-05**	**<10e-05**	**1986.60**	**1986.60**	**1986.60**
MP2	**56.09**	112.62	919.26	0.0092	0.0033	**0.0031**	2.1308	3.7294	**3.7309**
MP3	**56.99**	107.44	846.50	0.0037	**5.1383e-06**	1.3683e-05	2.1086	**3.6454**	**3.6454**
MP4	**55.19**	122.69	1059.10	0.0456	**0.0098**	0.0103	2.9649	**4.3627**	4.0797
MP5	**61.34**	176.4626	3724.80	0.0576	**0.0330**	0.3353	4.0558	13.1432	**14.9941**
MP6	**55.81**	113.30	998.66	0.0045	3.7396e-04	**8.6488e-04**	2.3188	2.9593	**3.0330**

Tabela 4.1: Tabela com os resultados dos testes de cada algoritmo nos problemas multiobjetivo selecionados.

na matriz. Quanto às medidas, T é o tempo total gasto em cada algoritmo, VR é o valor máximo de violação dentre todas as restrições do problema, B é o melhor valor de função-objetivo (penalizado) encontrado, e S representa a média de gerações a partir da última melhora de *gbest*. Em negrito estão os valores considerados melhores para cada medida. Podemos também observar como foi a progressão dos algoritmos ao longo das gerações nas Figuras 4.8-4.18. Alguns problemas tiveram que ser ampliados para que ficassem visíveis as diferenças entre os gráficos.

No caso multiobjetivo, como utilizamos um método de pesos, cada execução do algoritmo é feita em um peso diferente, segundo a fórmula:

$$Wv(1) = (j-1)/Ntestes$$
$$Wv(2) = 1-(j-1)/Ntestes$$
$$f(x, Wv) = f_1(x)Wv(1) + f_2(x)Wv(2)$$

Em que $Ntestes = 60$ e $j = \{1, 2, ..., 60\}$ é o número da execução atual. Daí temos a Tabela 4.1, com os resultados obtidos com estes parâmetros, e temos que T é o tempo médio gasto em cada algoritmo, SP e HV são as métricas de espaçamento e hipervolume, respectivamente. Os valores considerados melhores em cada medida estão em negrito. O ponto de referência do hipervolume foi o ponto com os piores valores de cada função-objetivo dentre todos os pontos produzidos pelos três algoritmos. Além dos dados, construímos as Figuras 4.19-4.24 com os pontos retornados em cada procedimento (sejam eles eficientes ou não), e se possível a FPO do problema.

Problemas	T-P(s)	T-D(s)	T-FM(s)	VR-P	VR-D	VR-FM	B-P	B-D	B-FM	S-P	S-D	S-FM
P1	**23.87**	51.95	97.48	**0.0680**	**0.0680**	**0.0680**	**2.3614**	**2.3614**	**2.3614**	**38.10**	39.47	40.37
P2	**24.43**	50.67	66.95	0.0169	0.0169	**0.0165**	**1.9184**	**1.9184**	**1.9184**	94.37	74.87	**73.53**
P3	**38.70**	71.43	107.43	**10e-04**	**10e-04**	**10e-04**	**-12.0198**	**-12.0198**	**-12.0198**	4.17	**3.93**	4.17
P4	**31.29**	102.32	926.98	0.0087	**<10e-05**	**<10e-05**	74.6151	**<10e-10**	**<10e-10**	99.00	91.5	**41.83**
P5	**24.30**	51.63	126.38	**0.0050**	**0.0050**	**0.0050**	**0.7475**	**0.7475**	**0.7475**	**37.17**	38.53	38.33
P6	**25.08**	135.42	6317.88	0.0058	**0.0018**	0.0045	74.6466	47.9670	**46.7983**	**0.00**	**0.00**	**0.00**
P7	**26.28**	136.92	937.22	**1.3917**	1.5136	1.5553	**-8.7152e+04**	-8.7969e+04	-8.7172e+04	**0.63**	7.23	6.73
P8	**26.42**	77.74	292.02	**0.0049**	0.0052	0.0072	53.5667	53.5722	**52.7022**	**0.00**	**0.00**	**0.00**
P9	**27.71**	382.07	983.91	8.8082	8.8082	**0.2284**	8.8954e+03	8.8954e+03	**208.1127**	49.13	48.67	**36.57**
P10	**30.27**	87.49	1859.50	0.1032	**0.0747**	0.0907	-42.0007	-48.8000	**-48.8115**	99.00	40.33	**5.60**
P11	**51.40**	2491.10	138110.00	3.4678e+07	67.8907	**5.4762**	2.2645e+19	**6.1748e+16**	6.3467e+16	99.00	40.63	**34.97**

Tabela 4.2: Tabela com os resultados dos testes de cada algoritmo nos problemas mono-objetivo selecionados, contando as 30 execuções.

4.4 Análise dos Resultados

Com base nos dados e gráficos observados na seção anterior, vamos discutir o que pode ser afirmado em relação ao método estudado. Dividiremos a análise entre os problemas mono e multiobjetivo.

4.4.1 Resultados nos SOP's

Os eventos observados na Tabela 4.2 remetem à um resultado frequente na área de otimização: o ganho de desempenho vem acompanhado de um custo maior no tempo de execução. Como já era esperado, o algoritmo do PSO, sem operador, retorna bem mais rápido do que utilizando operadores em todos os problemas, e acompanha o desempenho deles em alguns problemas mais simples, como P1, P2 e P3. Existiu ainda um caso excepcional, P7, em que o PSO puro obteve uma execução mais rápida e mais eficiente do que os demais, mesmo o problema sendo quadrático e de dimensão avançada, teoricamente apropriado para os operadores quadráticos. Porém, a aproximação pode ter falhado repetidas vezes, inserindo pontos aleatórios demais na população e freando a progressão do algoritmo evolutivo. Este pode ser considerado um caso à parte e, na média, os gráficos de convergência mostram que isto não acontece. Por isso, o PSO puro só seria aconselhável em problemas pouco complexos.

O valor de função-objetivo é muitas vezes considerado o foco da otimização, e em problemas com restrições este valor está diretamente ligado à satisfação destas. Porém os experimentos mostram várias situações nas quais o algoritmo que teve a menor máxima violação não foi o algoritmo que atingiu o melhor valor de função. Isto ocorre devido às folgas de restrição e problemas de escala, dado que a penalidade pode não acompanhar o ganho na função-objetivo. É difícil calcular exatamente qual a penalidade mais apropriada para cada problema, e mesmo que ela seja encontrada, pode ser que isso restrinja muito o processo de busca. A única informação inferida apenas dos dados é de que os operadores melhoram ambas as violações e os valores de função, nem sempre com uma correlação entre elas. Entretanto, neste aspecto o gráfico médio do melhor indivíduo é mais conclusivo: Ve-

mos que execuções de alto desempenho em problemas maiores são raridade sem os operadores, indicando que o algoritmo puro fica concentrado na primeira solução factível que ele encontra, melhorando pouco depois disso e dependendo da aleatoriedade. Já os algoritmos que utilizam aproximação se alternam na liderança pelo melhor desempenho, sendo difícil dizer a priori qual é mais eficiente. Aliando tudo isto ao fato de que apenas P11 teve um gap (diferença entre valor ótimo e valor encontrado) maior que 10%, pode-se afirmar que os operadores são de suma importância quando se quer buscar ótimos factíveis em problemas complexos.

A última característica a ser analisada é o que foi chamado "estagnação", Referindo-se à quantas gerações passaram-se desde que o *gbest* foi modificado pela última vez. Esta medida indica pode dizer muito sobre a capacidade do algoritmo de sair de ótimos locais. Neste caso, pode-se ponderar os dados de duas formas: Ou valores grandes indicam facilidade convergência para o ótimo, enquanto pequenos mostram dificuldade; Ou valores grandes indicam pouco avanço, enquanto os pequenos significam uma consistente melhora no melhor indivíduo. A primeira se transforma na segunda à medida que o problema vira mais complexo, então é necessário uma análise do benchmark primeiro. Nos problemas simples, o operador encontra rapidamente o ótimo, e o PSO sozinho necessita mais gerações para compensar a falta de busca local. Nos problemas mais complexos, como já apontado anteriormente, o PSO puro estagnava nas primeiras bacias de atração que passa, convergindo prematuramente, e os algoritmos com operador continuam a busca, principalmente quando se tem a matriz cheia. Também se observa isto facilmente nos gráficos, em que o algoritmo mais simples tem uma média de melhor indivíduo praticamente paralela ao eixo das abscissas. Então, o que é extraído desta parte dos experimentos é que, sem perda de desempenho, o tempo dos algoritmos com operadores pode ser reduzido se a quantidade de gerações não for um valor padronizado.

4.4.2 Resultado nos MOP's

Novamente, fixando o número de gerações, o PSO puro vai ser sempre mais rápido e o operador com matriz cheia mais lento, já que este último possui a maior ordem de complexidade. Porém, nessa base de problemas, ele só foi eficiente no MP1, tendo um comportamento bem próximo de aleatório nos outros.

Os algoritmos híbridos, por sua vez, mostraram bastante variação no desempenho. Em MP2 e MP3, conseguiram bons valores de métricas, mas isto não é sustentado pelo gráfico de dispersão em conjunto com a FPO do terceiro problema-teste: os pontos se acumularam nos extremos. A medida de *spacing* foi próxima de zero, como se deseja, mas é pela aglomeração pontual das soluções. Se houvesse uma terceira métrica que considerasse os extremos, os valores desta seriam pouco satisfatórios. O fato da FPO ser côncava pode ter sido o motivo de tal comportamento.

Os problemas MP4 e MP5 têm a multimodalidade como agravante, e os algoritmos não foram muito precisos nestes. No primeiro deles, o operador com matriz diagonal teve uma melhor distribuição e atingiu alguns pontos na FPO. No segundo, o método com matriz completa fez esse papel. No quesito distribuição no espaço, os dois mantiveram um espaçamento satisfatório. O último problema é semelhante aos problemas ZDT, mas não é informado a FPO. Como é um problema unimodal, e os dois algoritmos chegaram à resultados semelhantes, provavelmente muitos pontos são parte do ótimo.

4.4.3 Significância Estatística

Graficamente e com os dados da melhor execução, é difícil dizer se o tempo a mais gasto pelo operador de matriz completa é mais interessante que o apenas com a diagonal. Para tentar comparar melhor os algoritmos, foram feitos testes estatísticos de alguns problemas que mostraram, já na curva de convergência, diferença no desempenho. Os métodos serão comparados segundo as hipóteses:

H_0: Os dois algoritmos têm, em média, precisão equivalente.

H_1: O método que utiliza FMQA é mais preciso do que aquele munido de DQA.

No intuito de descobrir se devemos rejeitar a hipótese nula, vamos executar testes de randomização (Dwass, 1957) para construir as distribuições da comparação entre medianas. O ponto negro no eixo das abscissas nos diz o resultado do teste em cada problema: Se estiver na direita da distribuição, o primeiro algoritmo (diagonal), é significantemente melhor do que o outro; Se ela estiver na esquerda, o segundo (matriz cheia) é significante; Se estiver no centro, não há diferença significativa. As Figuras 4.1-4.7 mostram os resultados dos testes estatísticos.

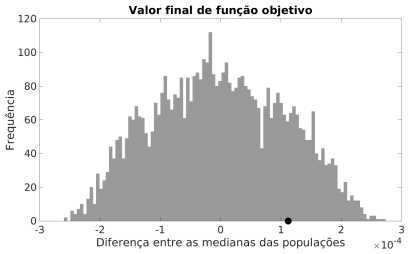

Figura 4.1: Teste estatístico do problema P2.

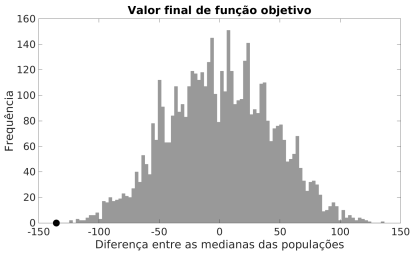

Figura 4.2: Teste estatístico do problema P6.

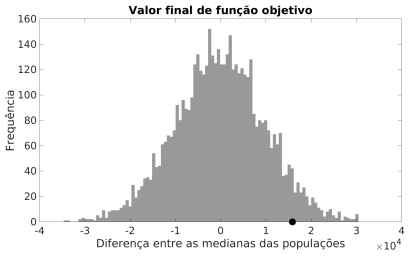

Figura 4.3: Teste estatístico do problema P7.

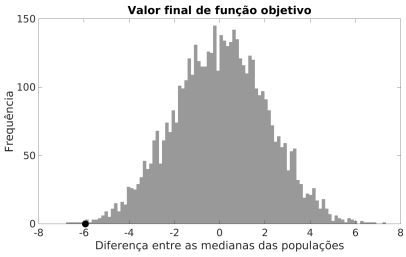

Figura 4.4: Teste estatístico do problema P8.

Dos 7 problemas analisados estatisticamente, o operador quadrático diagonal foi significantemente melhor em quatro. É um número equilibrado de problemas, porém, como o tempo que ele gasta é muito inferior, preencher completamente a matriz só é indicado se o tempo não for importante, e

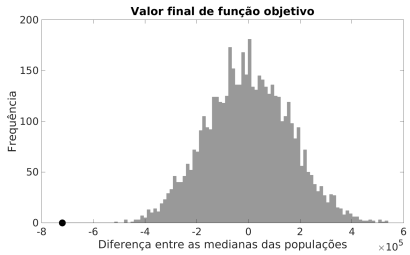

Figura 4.5: Teste estatístico do problema P9.

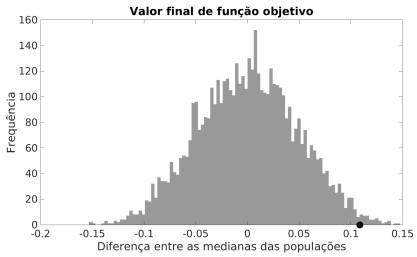

Figura 4.6: Teste estatístico do problema P10.

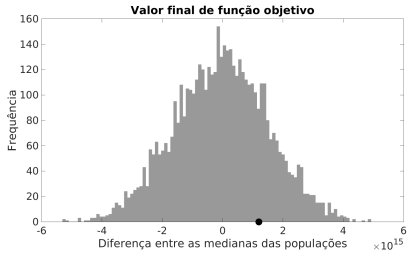

Figura 4.7: Teste estatístico do problema P11.

mesmo nestes casos, é mais importante testar os dois.

4.5 Saída Gráfica dos Problemas-Teste

Através das Figuras 4.8 a 4.24, seguem os resultados dos experimentos, graficamente. o valor de aptidão se refere ao calculado na Equação 3.6.

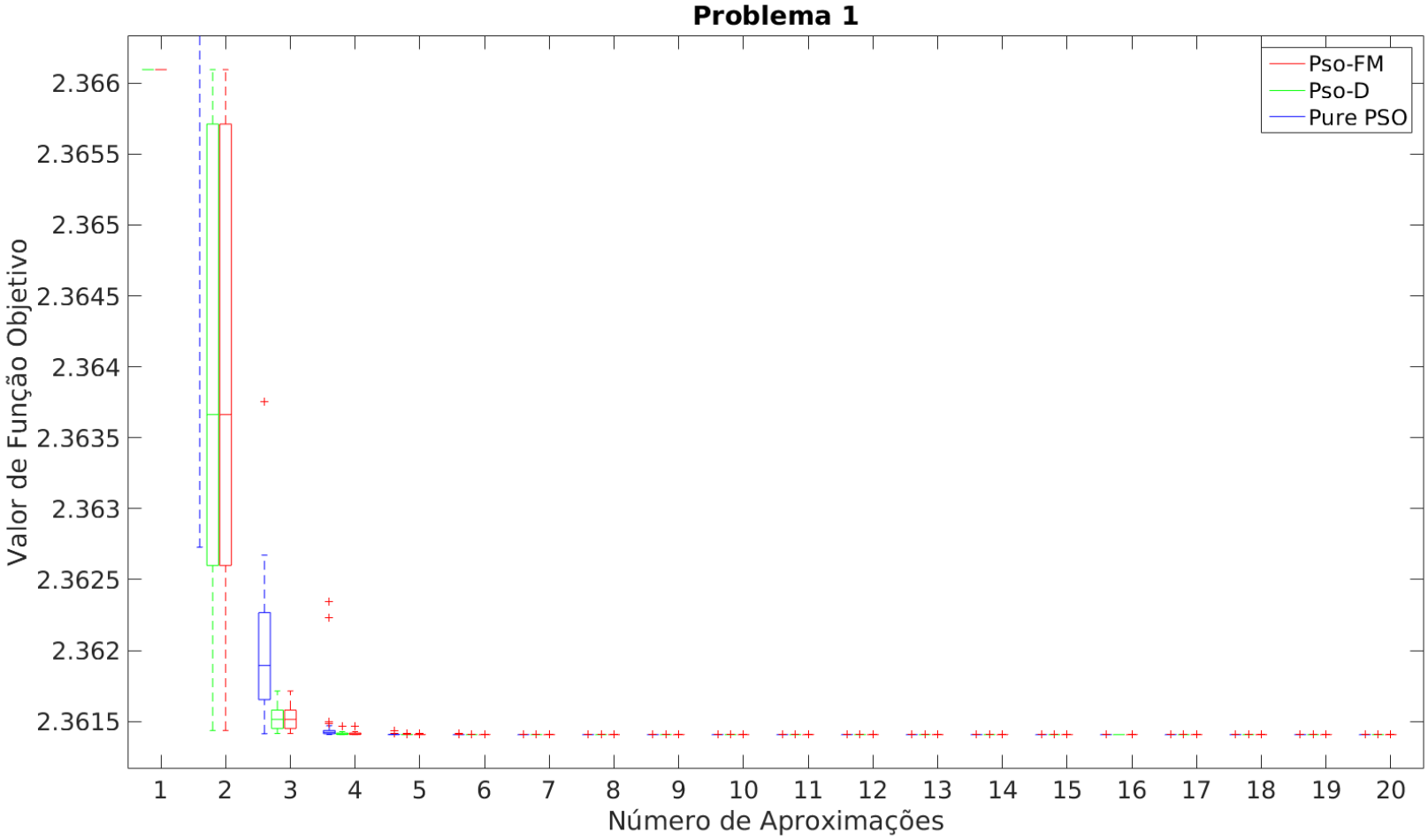

Figura 4.8: Box-plot da aptidão do melhor indivíduo nas execuções do problema P1.

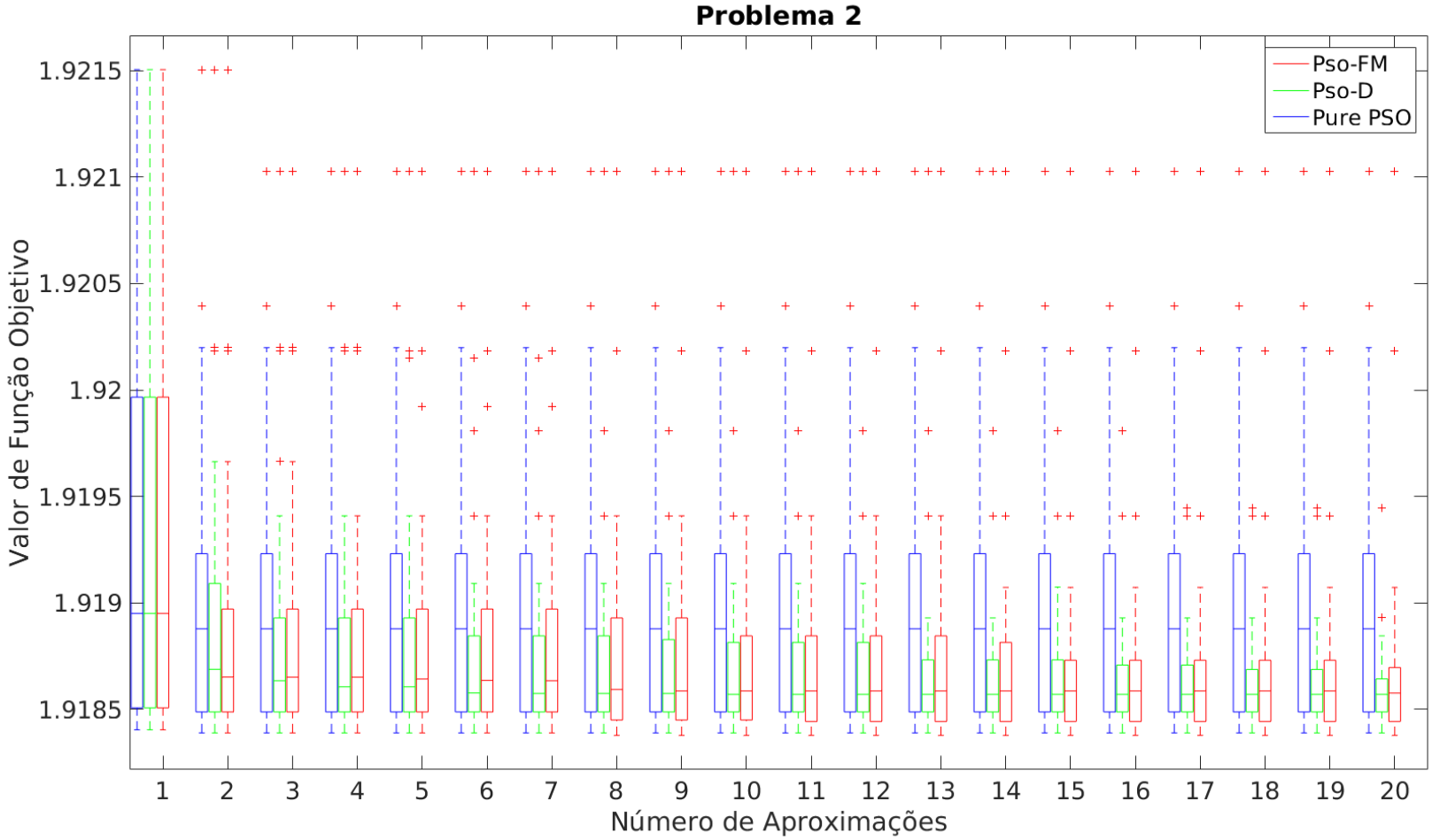

Figura 4.9: Box-plot da aptidão do melhor indivíduo nas execuções do problema P2.

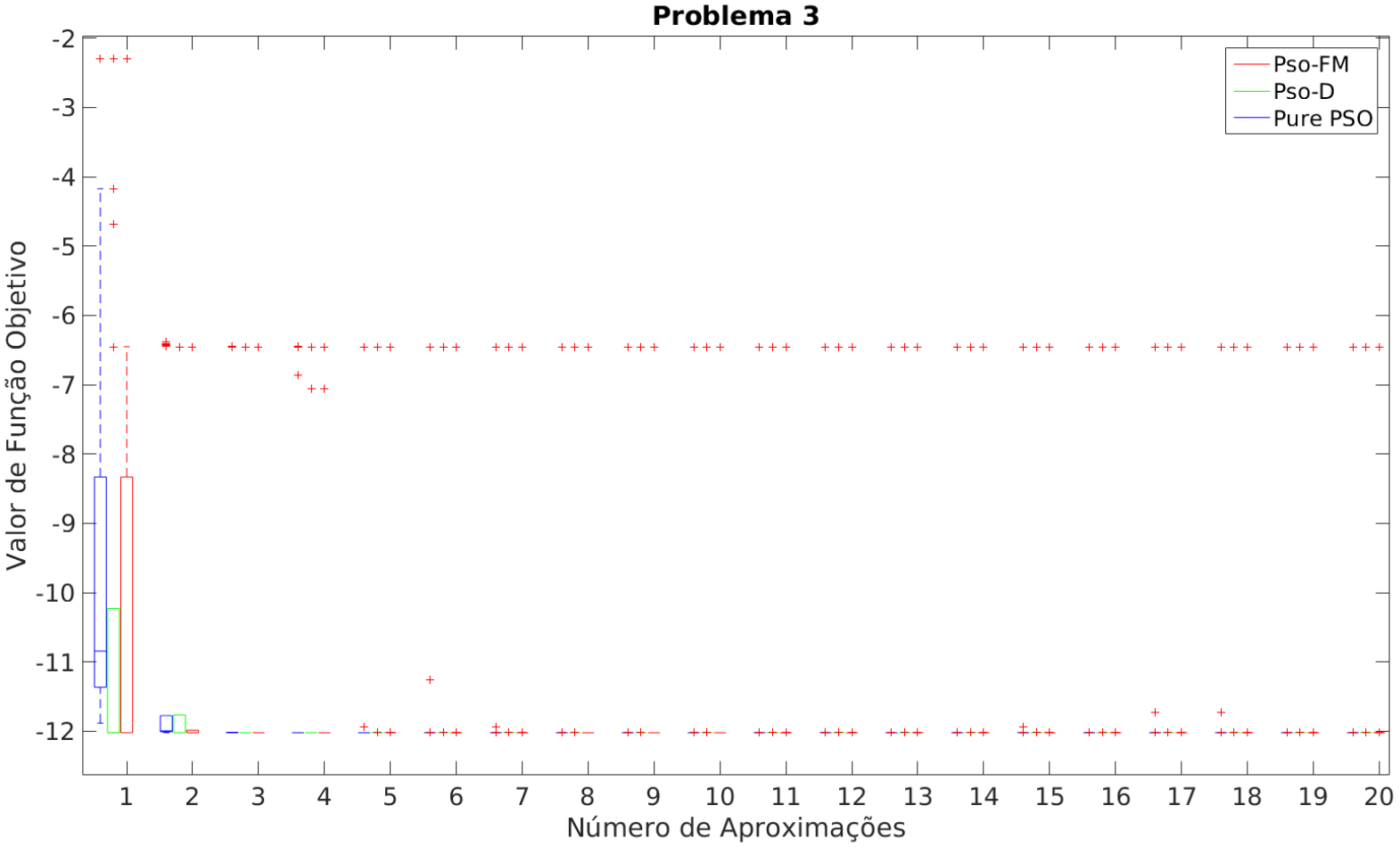

Figura 4.10: Box-plot da aptidão do melhor indivíduo nas execuções do problema P3.

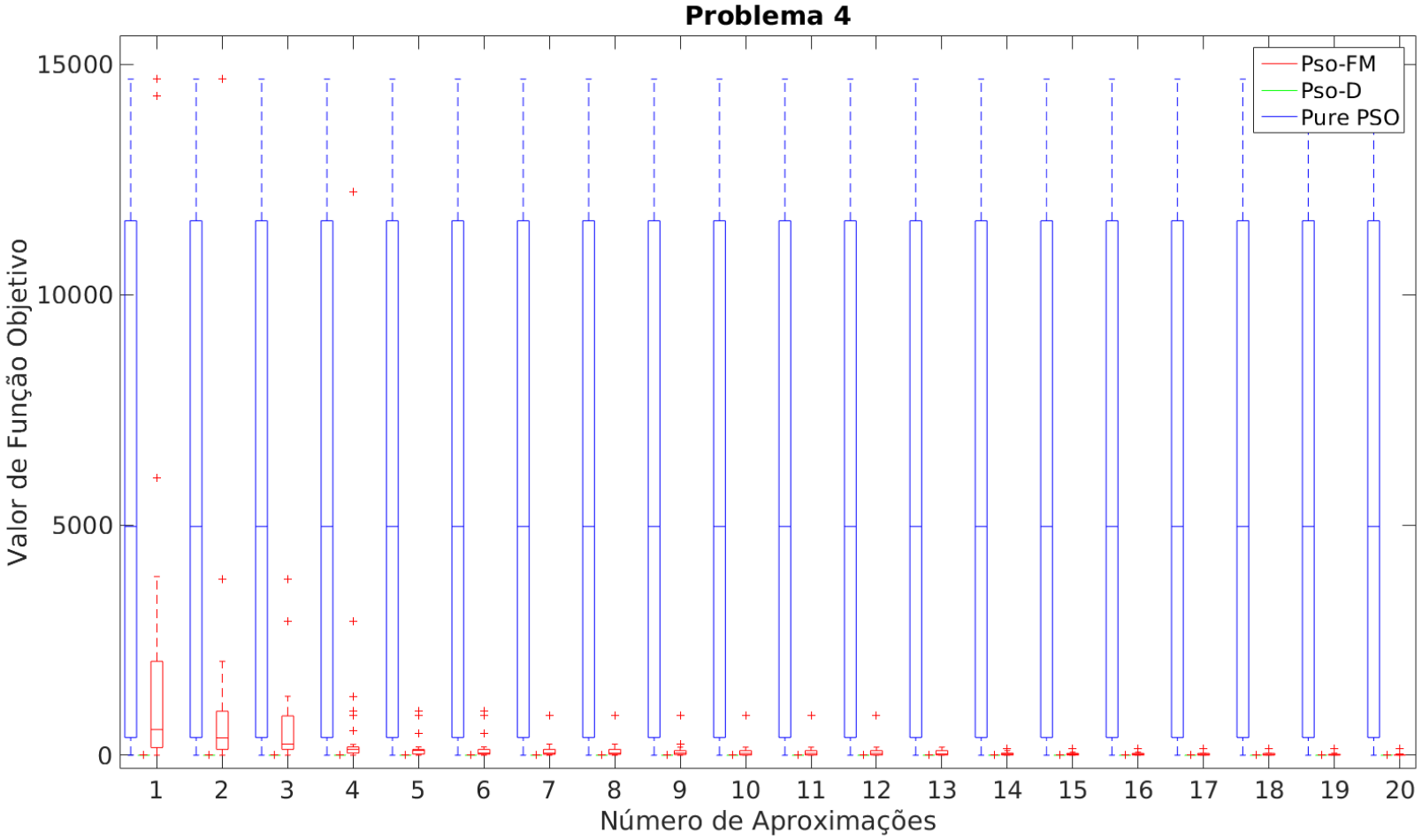

Figura 4.11: Box-plot da aptidão do melhor indivíduo nas execuções do problema P4.

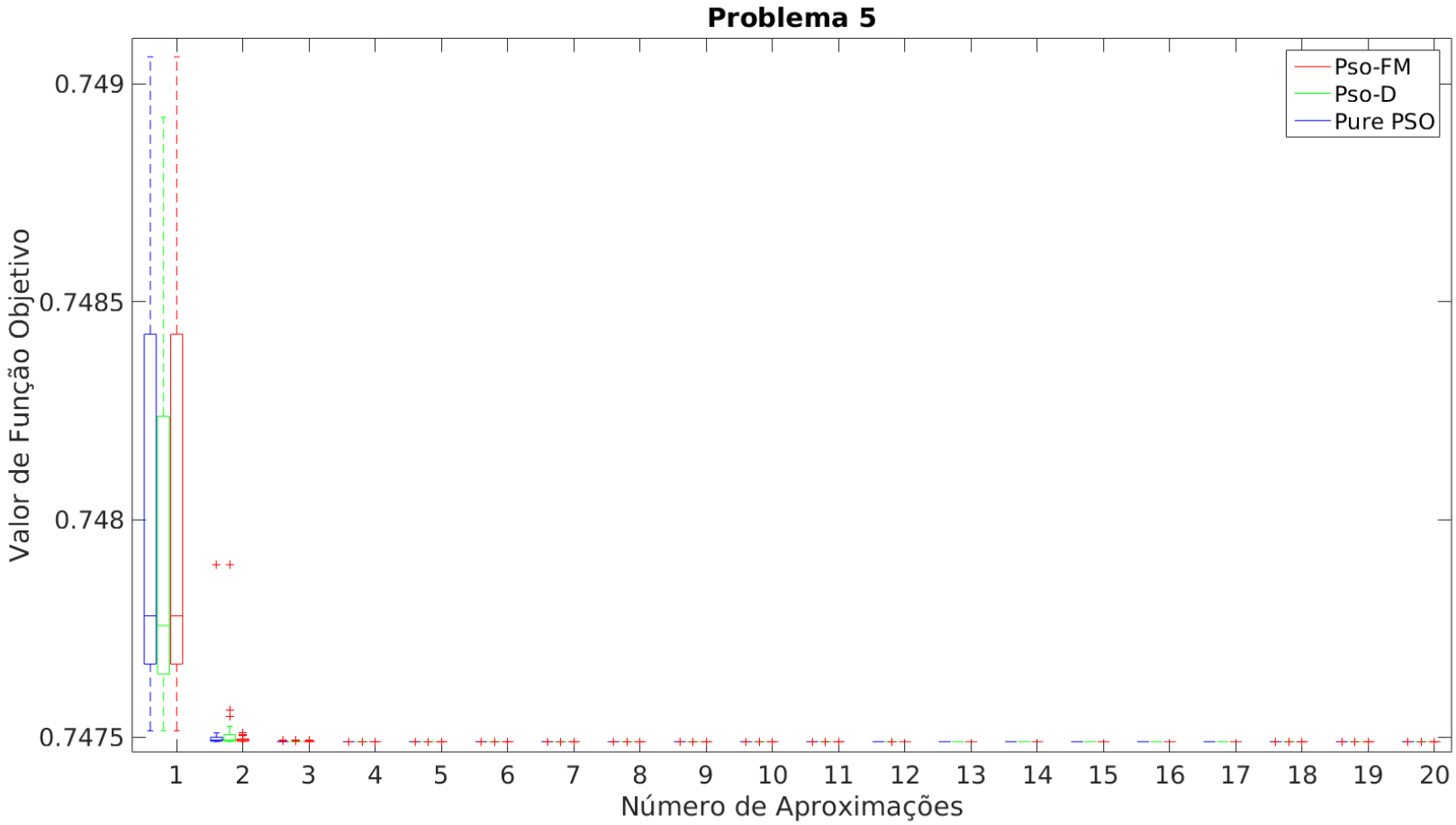

Figura 4.12: Box-plot da aptidão do melhor indivíduo nas execuções do problema P5.

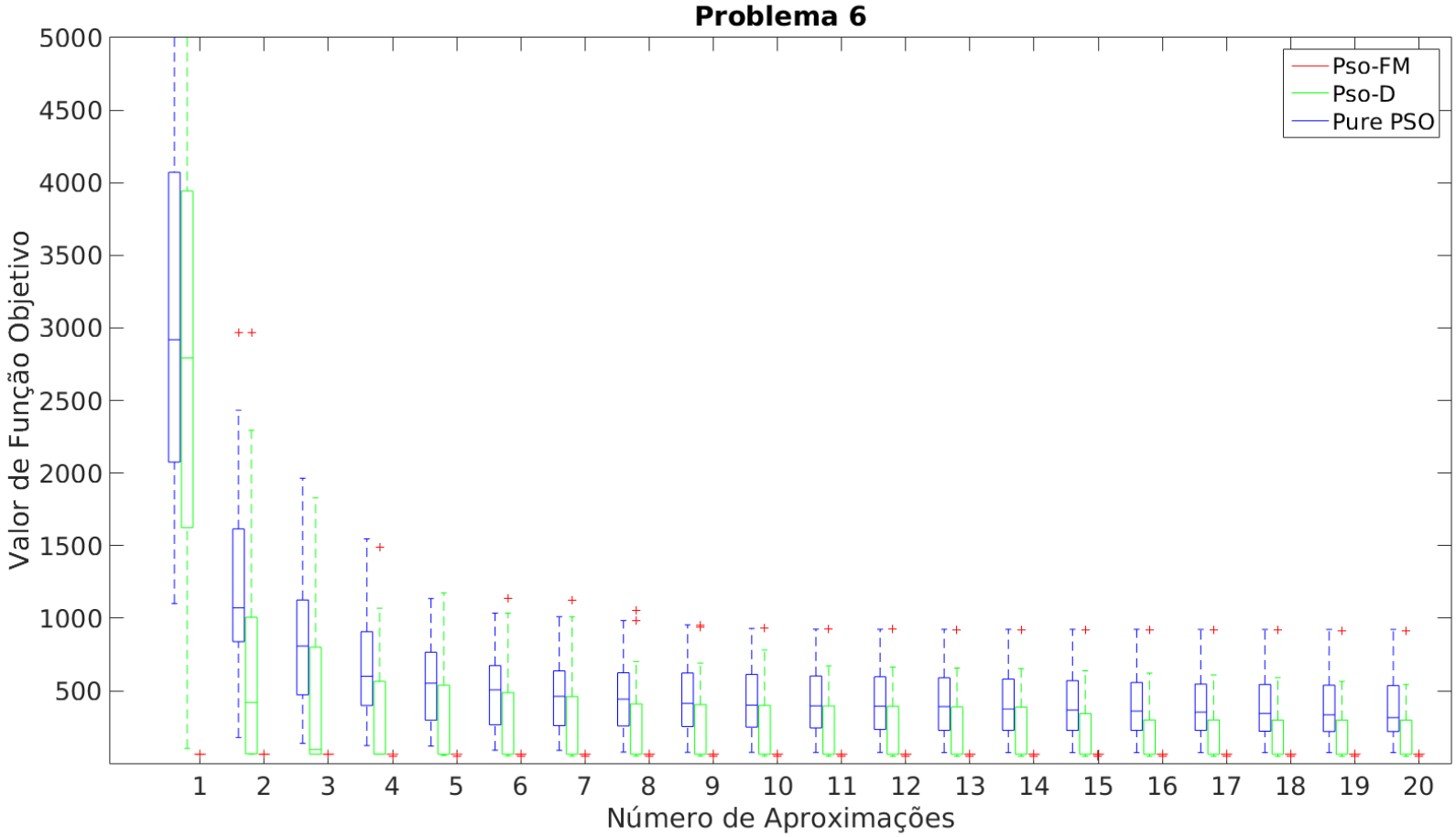

Figura 4.13: Box-plot da aptidão do melhor indivíduo nas execuções do problema P6.

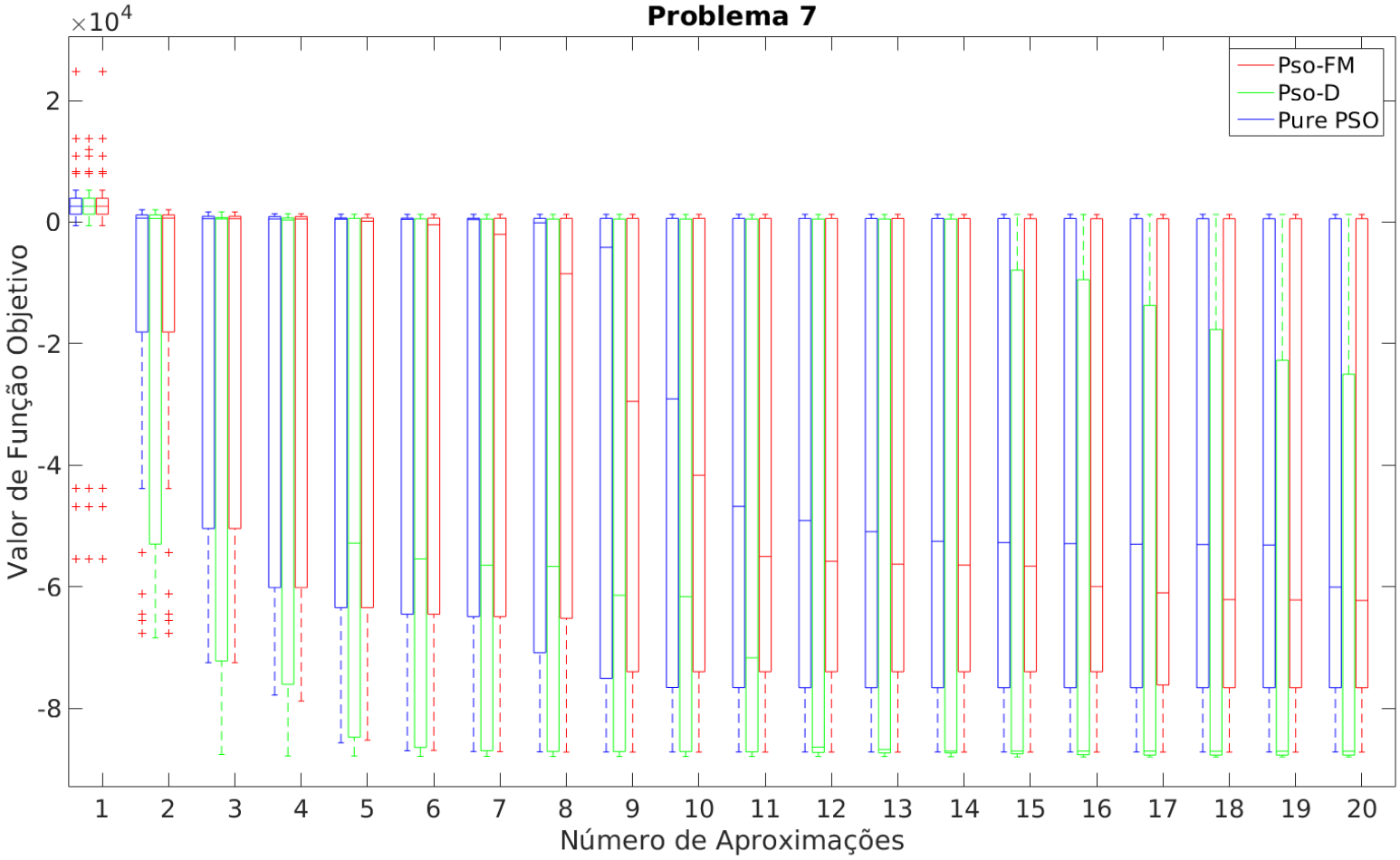

Figura 4.14: Box-plot da aptidão do melhor indivíduo nas execuções do problema P7.

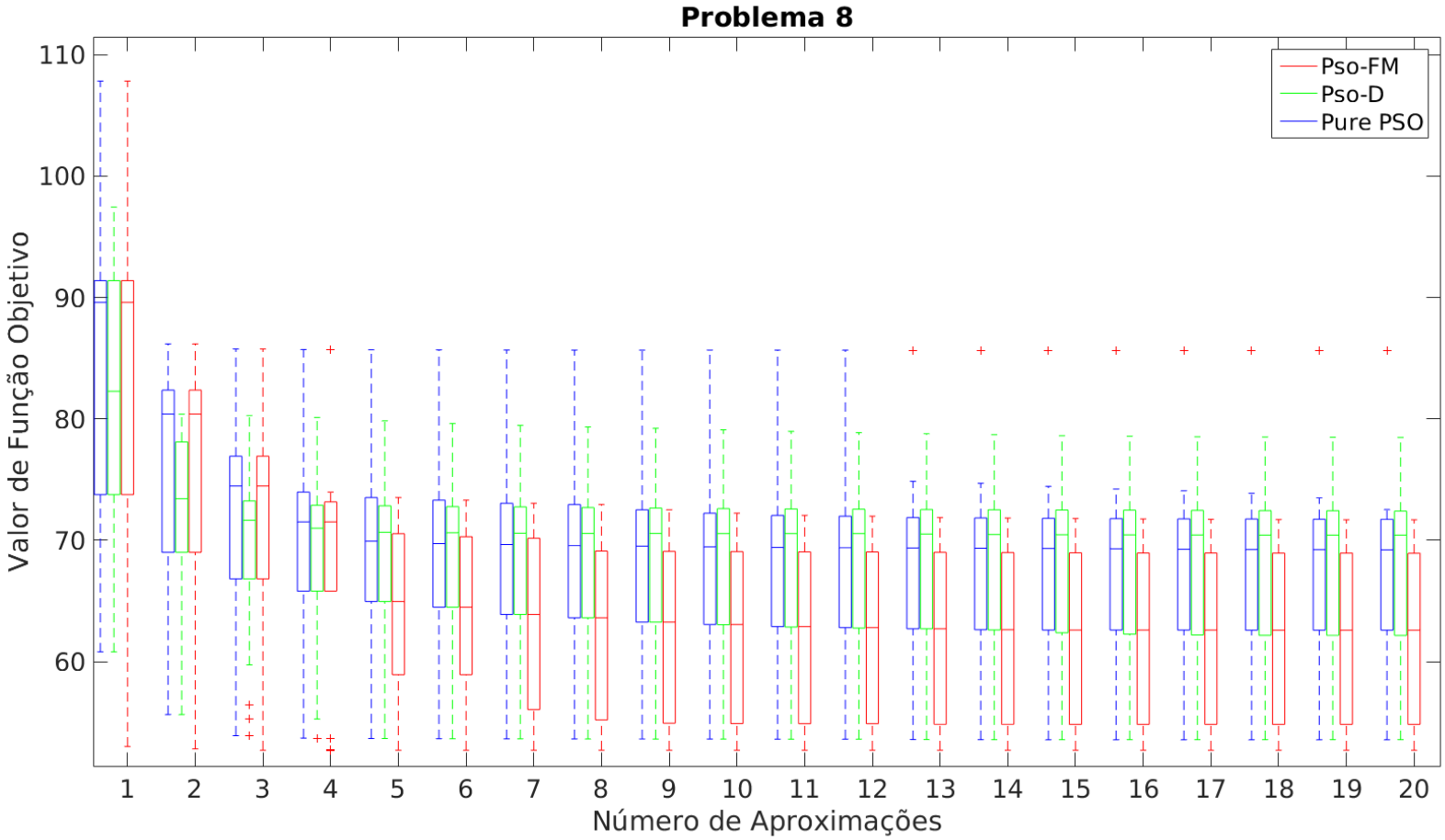

Figura 4.15: Box-plot da aptidão do melhor indivíduo nas execuções do problema P8.

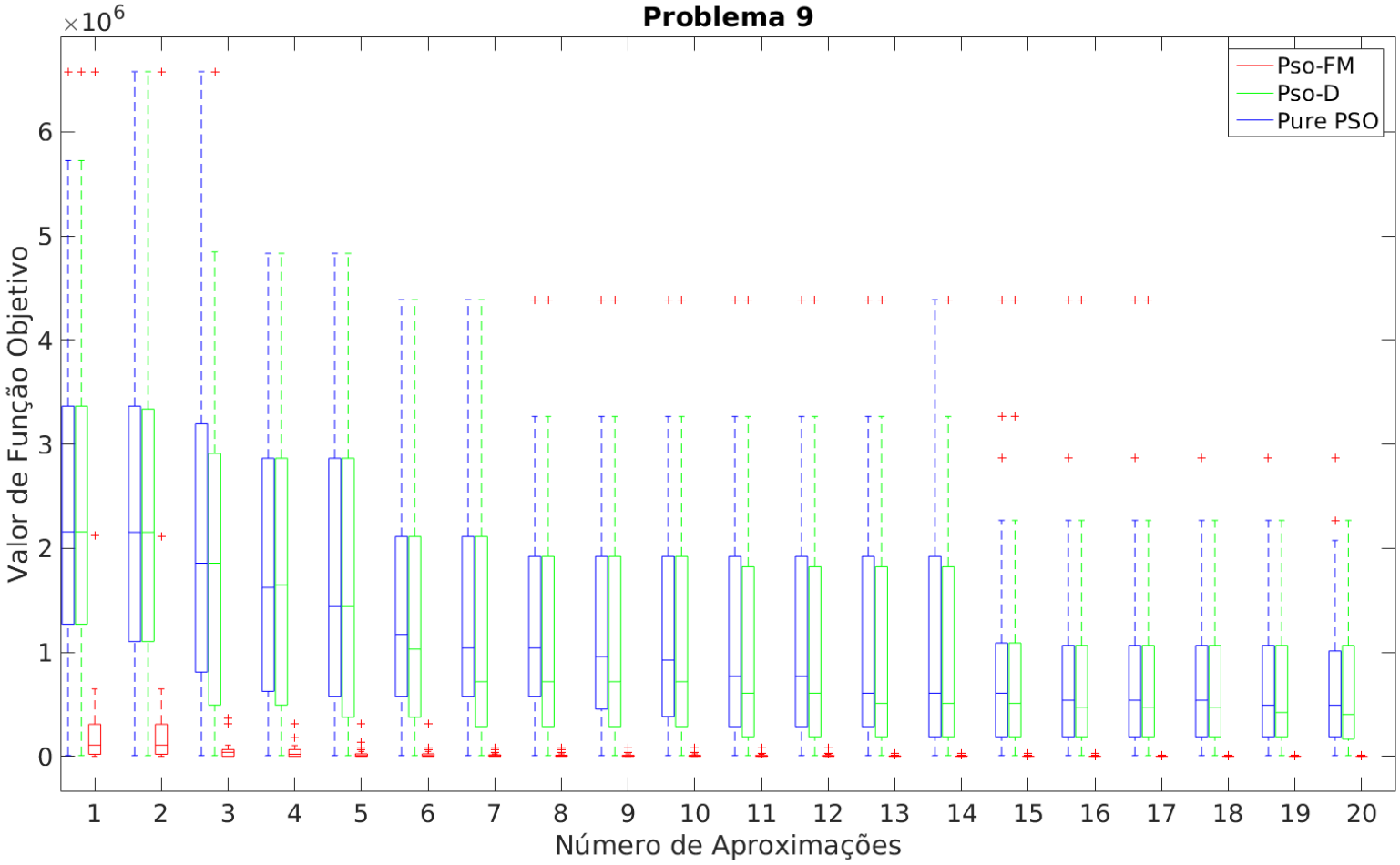

Figura 4.16: Box-plot da aptidão do melhor indivíduo nas execuções do problema P9.

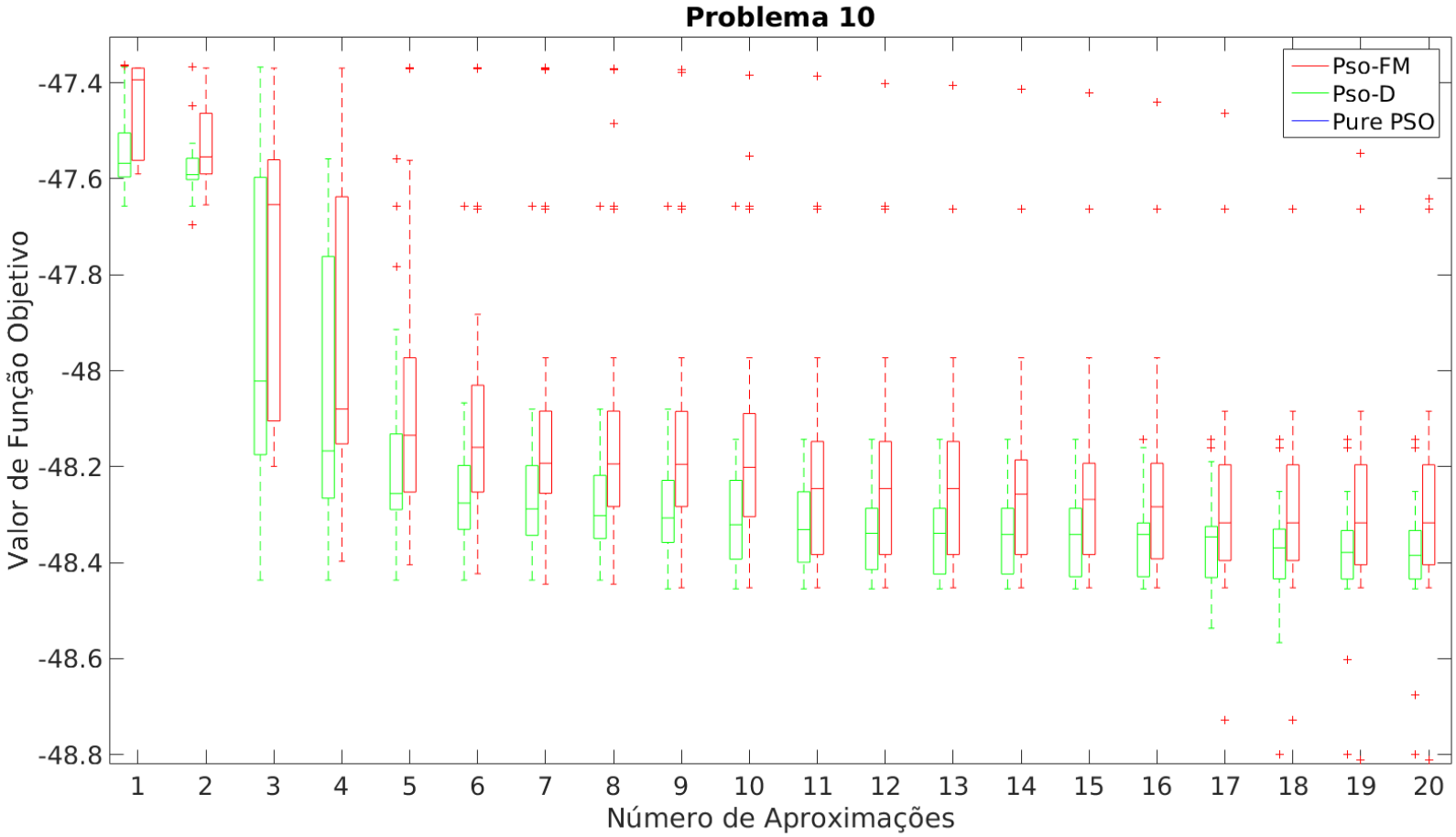

Figura 4.17: Box-plot da aptidão do melhor indivíduo nas execuções do problema P10.

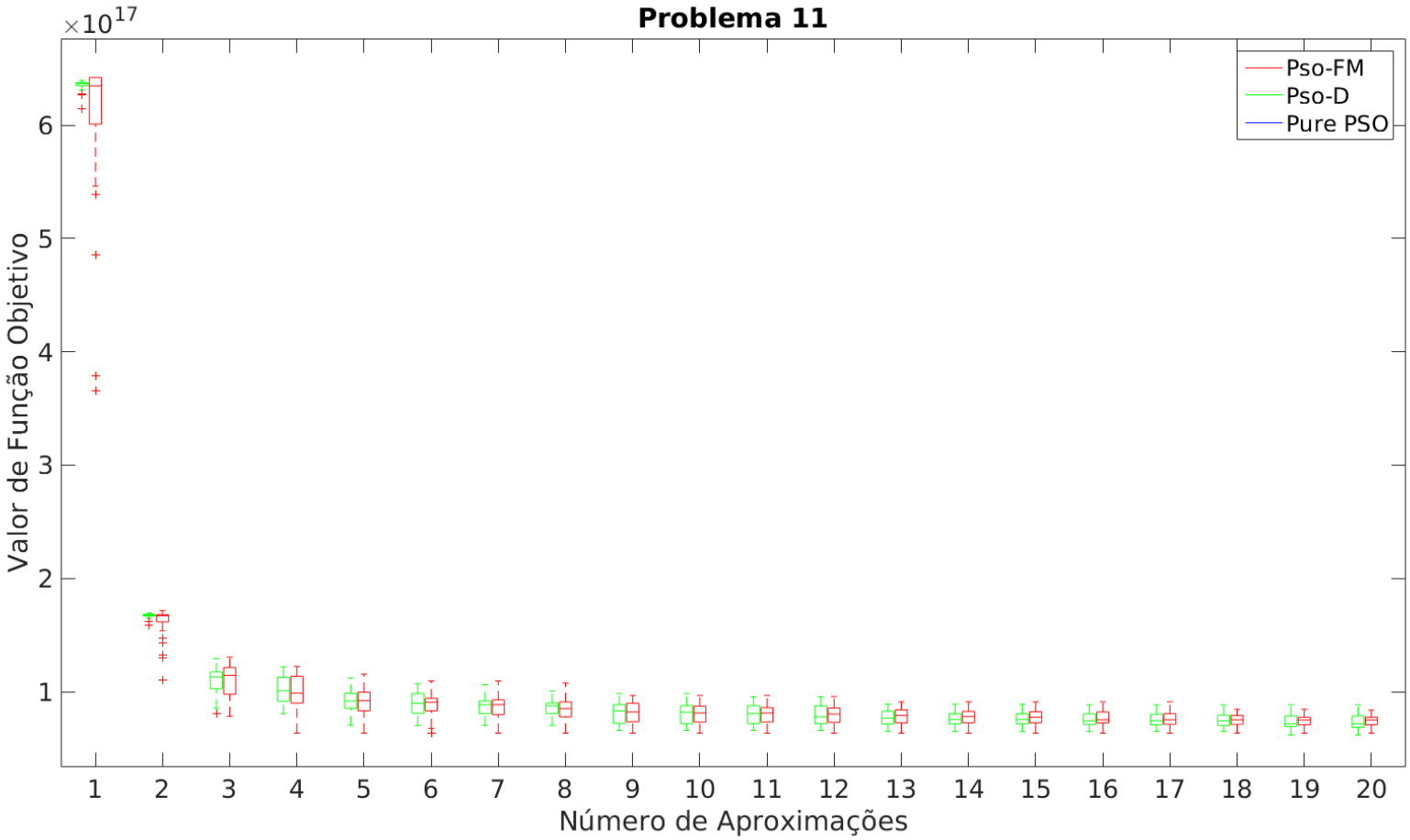

Figura 4.18: Média de aptidão do melhor indivíduo de cada execução feita no problema P11.

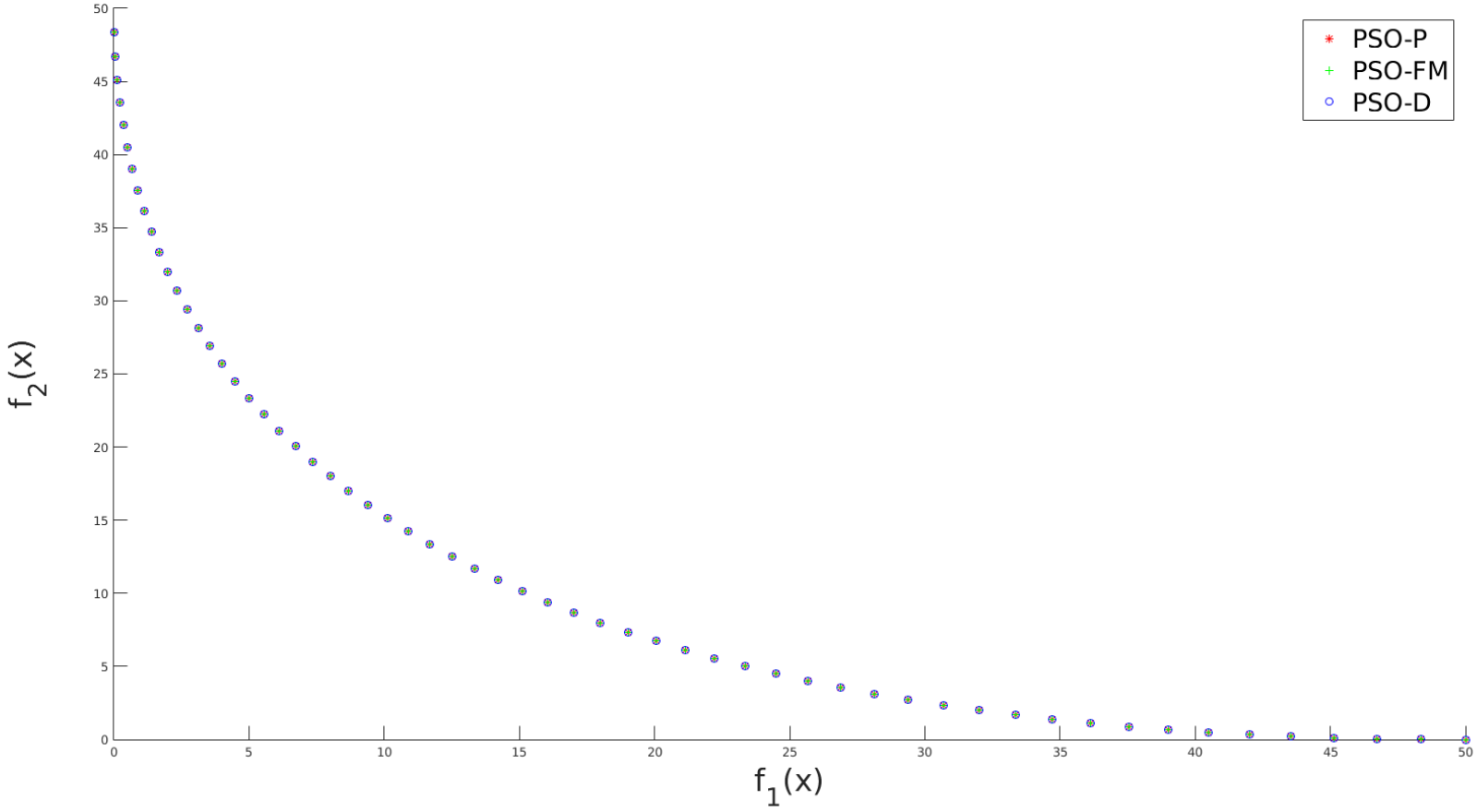

Figura 4.19: Conjunto de soluções final encontradas por cada algoritmo no MP1, incluindo soluções dominadas.

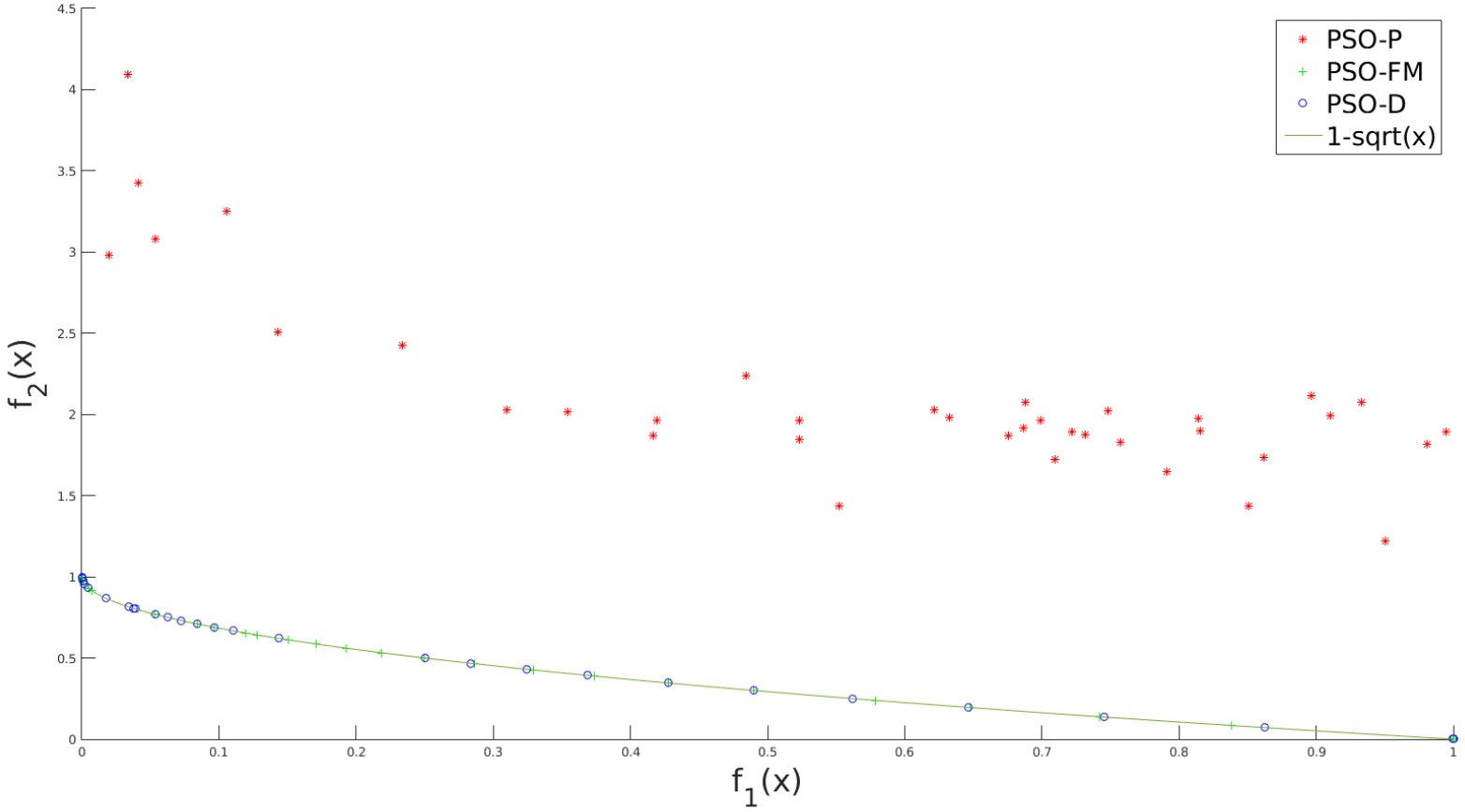

Figura 4.20: Conjunto de soluções final encontradas por cada algoritmo no MP2, incluindo soluções dominadas. A linha contínua representa a FPO, que é convexa.

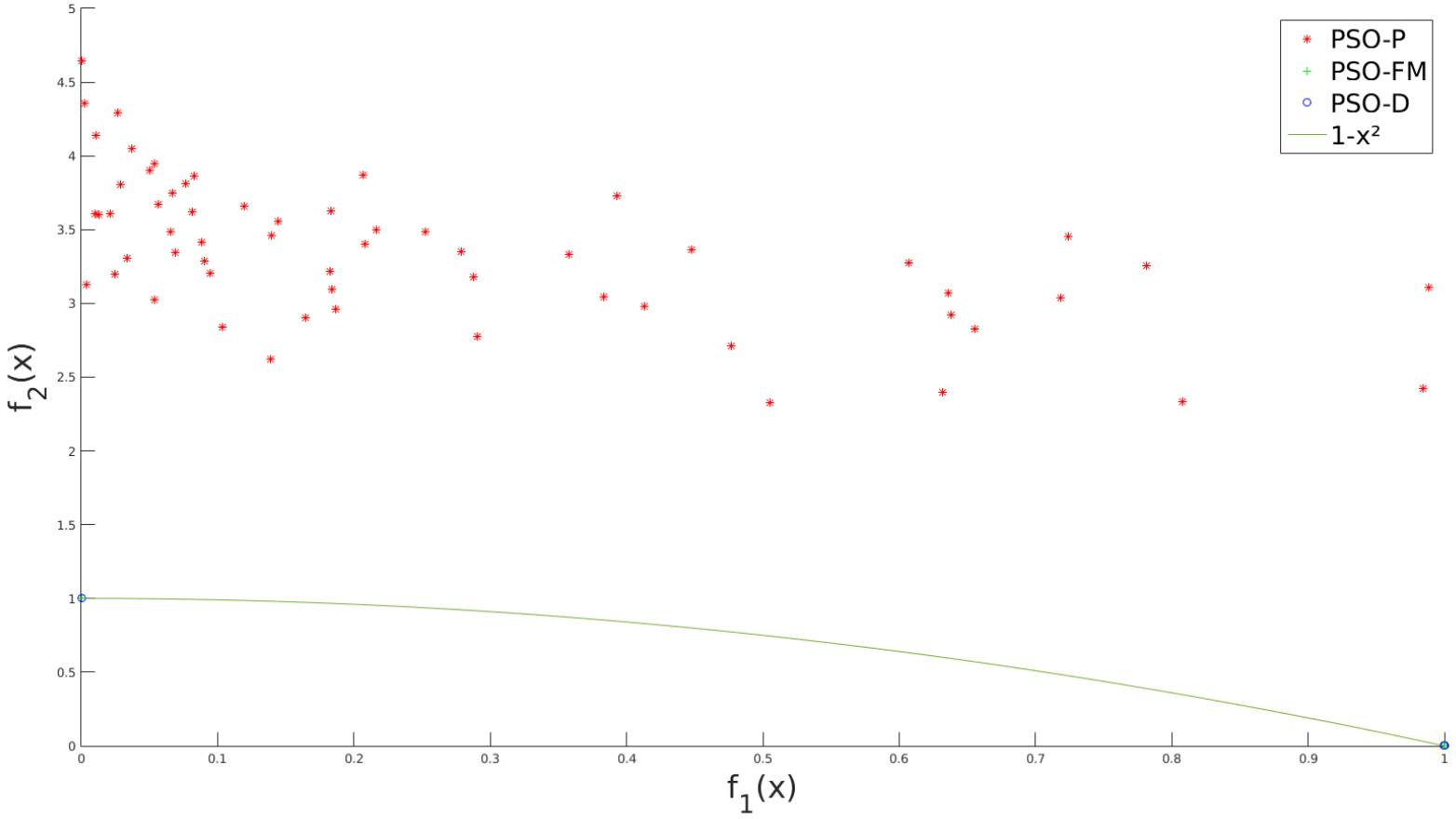

Figura 4.21: Conjunto de soluções final encontradas por cada algoritmo no MP3, incluindo soluções dominadas. A linha contínua representa a FPO, que é côncava.

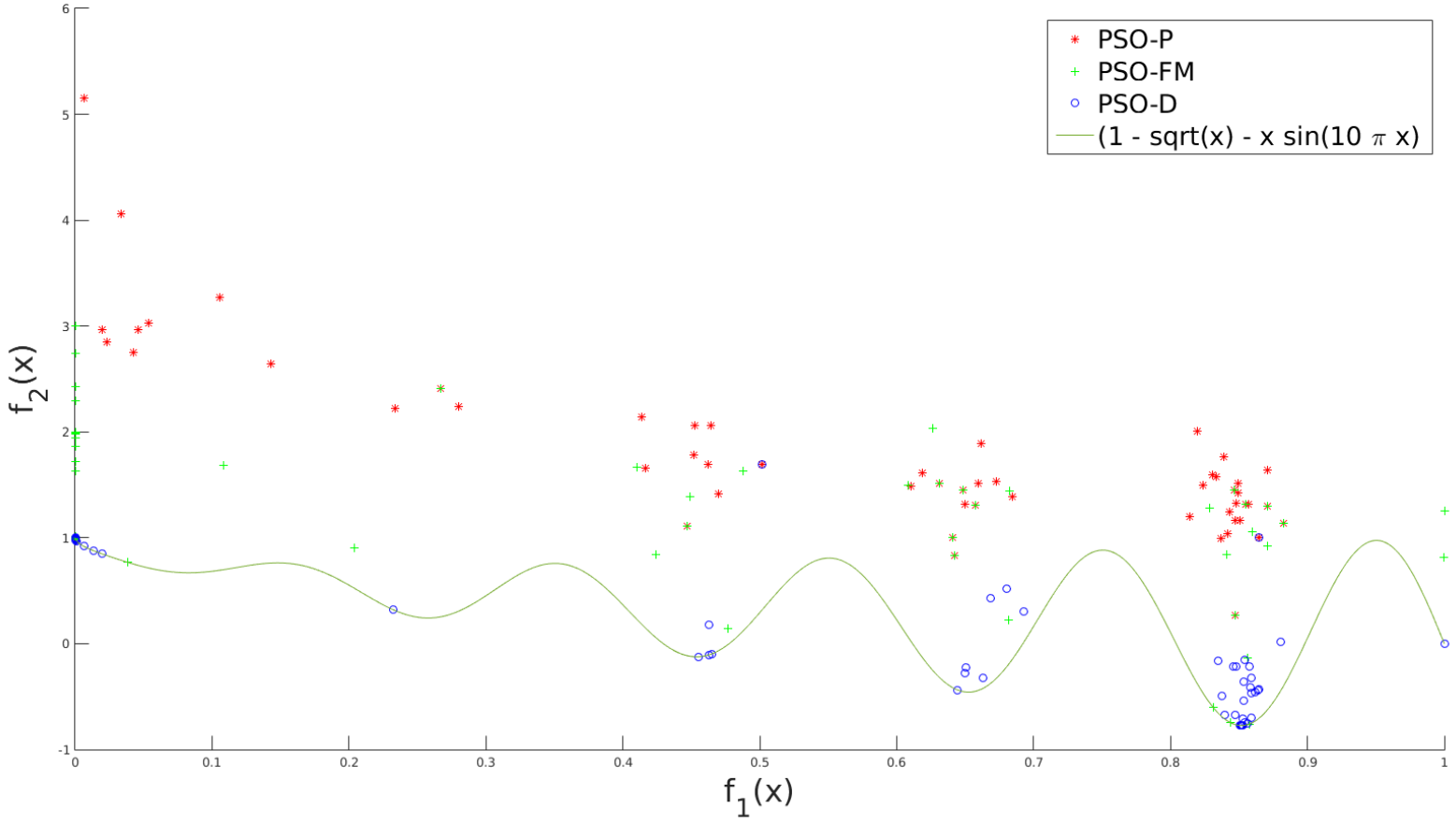

Figura 4.22: Conjunto de soluções final encontradas por cada algoritmo no MP4, incluindo soluções dominadas.A linha contínua contém a FPO, que é descontínua e distribuída entre as bacias de atração.

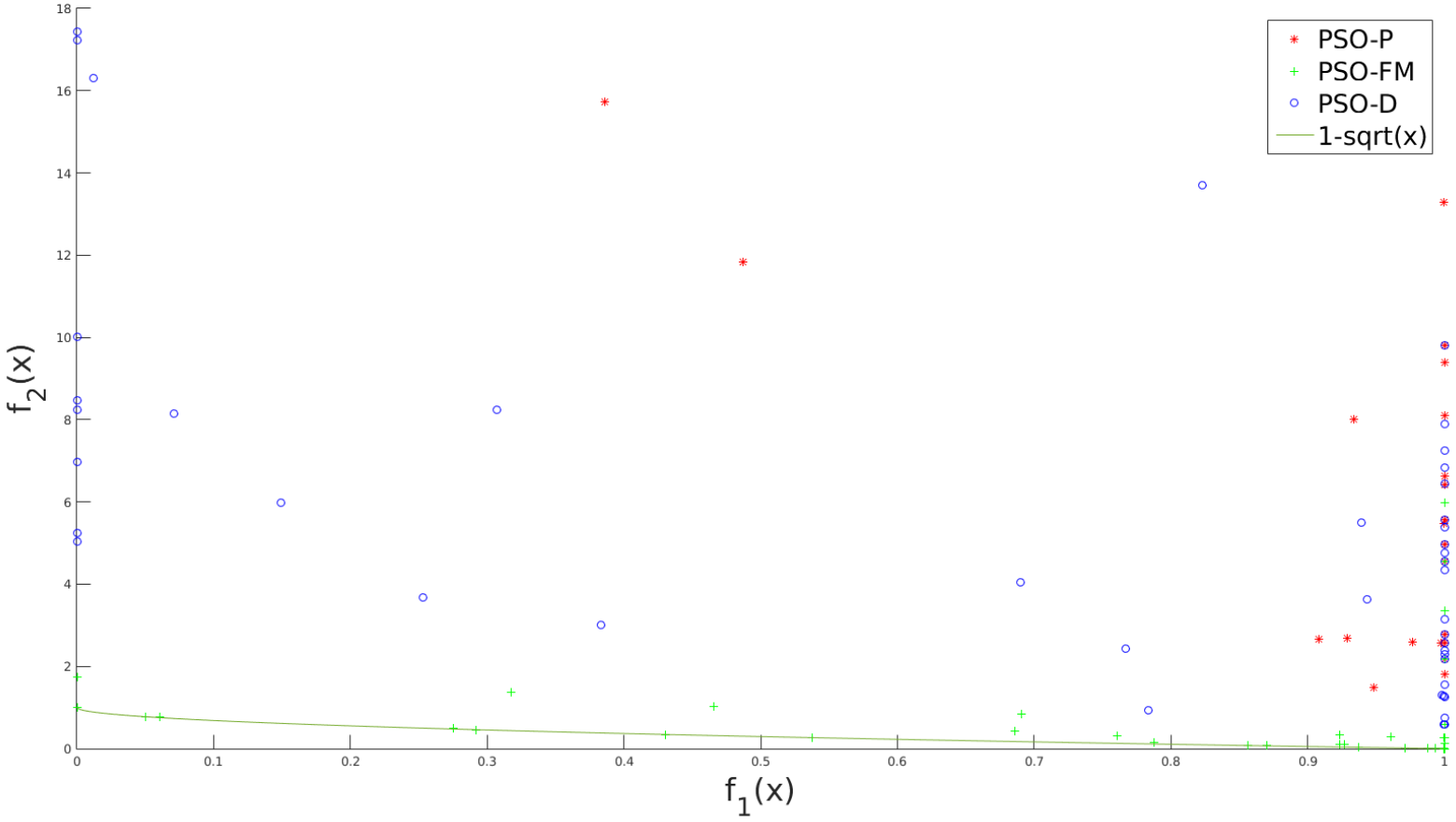

Figura 4.23: Conjunto de soluções final encontradas por cada algoritmo no MP5, incluindo soluções dominadas.A linha contínua contém a FPO, que é discreta.

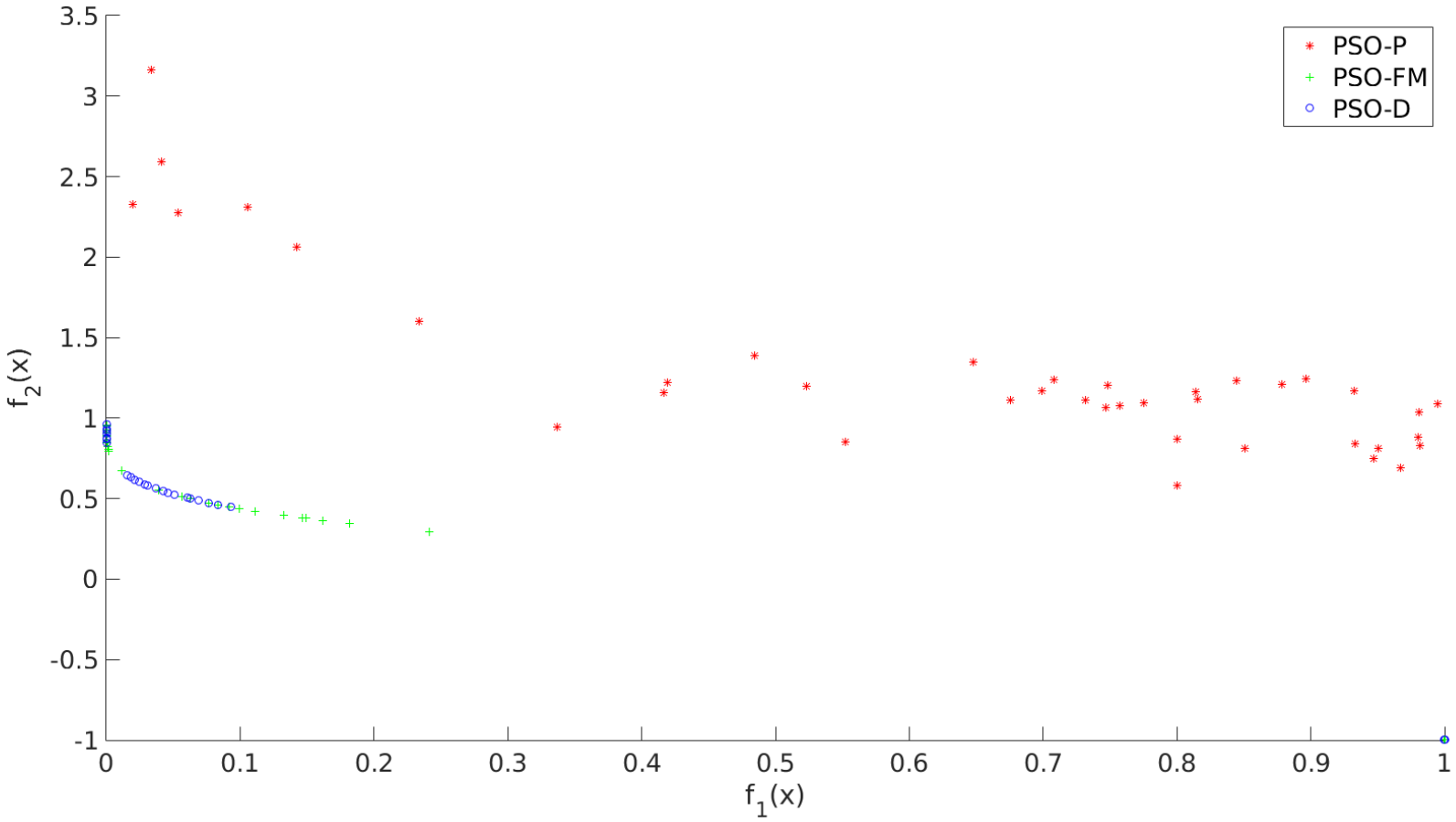

Figura 4.24: Conjunto de soluções final encontradas por cada algoritmo no MP6, incluindo soluções dominadas.

Capítulo 5

Conclusão

Neste trabalho, analisamos o desempenho de dois operadores de busca local baseados em aproximação quadrática utilizados em conjunto com o algoritmo PSO. Lidamos com 11 SOP's com restrições de igualdade não-lineares, e 6 MOP's com até 10 dimensões.

Os operadores se diferenciavam pela forma da matriz Hessiana da aproximação: uma contendo todos os elementos reais, com qualquer sinal e outra com apenas os elementos da diagonal principal e positivos. Testamos a importância destes dois operadores em 17 problemas, e pode-se concluir que o PSO, por mais que seja capaz de lidar com problemas simples, tem uma grande queda de desempenho ao tratar problemas mais difíceis. Os operadores, apesar de terem um custo computacional grande, aumentaram significativamente o desempenho do PSO, e a hibridização foi capaz de gerar resultados com boa precisão nos problemas testados. As soluções obtidas nos SOP's apresentaram em sua maioria, gaps de menos de 10% e pequenas violações nas restrições, embora em alguns casos elas ainda sejam violações maiores que a tolerância estabelecida. Nos MOP's, as métricas de espaçamento e hipervolume avaliaram as soluções eficientes e mostraram que os algoritmos foram precisos nos problemas unimodais, e precisos nos problemas convexos. Assim, pudemos concluir que o PSO necessita de mais recursos para lidar com problemas de maior complexidade, e os operadores de busca local se mostraram uma opção eficaz.

Outro objetivo alcançado foi a comparação entre os dois operadores propostos, analisando se era compensativo determinar um maior número de elementos matriz da aproximação quadrática e abrir mão da convexidade. Os testes estatísticos mostraram que não existe uma tendência geral quando todos os problemas são levados em conta, e portanto o ganho de tempo ao utilizar apenas matriz diagonal positiva definida é maior que o ganho de desempenho.

Os testes feitos mostram a dificuldade de se atacar MOP's com robustez, uma vez que pequenas mudanças no espaço de busca foram o suficiente para tornar os métodos elaborados imprecisos. Com a tendência de que se aumente o interesse e a complexidade destes problemas, é necessário que hajam melhorias nos algoritmos desenvolvidos para estes problemas. Em particular, os MOEA's ainda são a opção mais natural por evoluírem várias soluções juntas, mas dificilmente a simplicidade deles será suficiente. Para se obter avanços no campo da otimização, operadores como o proposto aqui devem ser mais testados e aprimorados.

5.1 Trabalhos Futuros

Seria interessante, posteriormente, testar mais operadores para o PSO, utilizando as aproximações quadráticas com diferentes métodos de busca local, como VNS ou ILS.

Apêndice A

Problemas teste

- **P1:**

$$\begin{aligned} &\min f(x) = x_1^2 + x_2^2 + x_3^2 \\ &\text{Sujeito à: } (x_1 - 2)^2 + (x_2 - 1)^2 + 4(x_3 - 1)^2 - 1 = 0 \\ &-10 \leq x_i \leq 10,\ i = 1, 2, 3 \end{aligned} \tag{A.1}$$

- **P2:**

$$\begin{aligned} &\min f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 \\ &\text{Sujeito à: } x_1^2 - x_2 = 0 \\ &-10 \leq x_i \leq 10,\ i = 1, 2 \end{aligned} \tag{A.2}$$

- **P3:**

$$\begin{aligned} &\min f(x) = x'.A'.Ax - 10[1\ 1]\cos(2\pi Ax) \\ &\text{Sujeito à: } (x_1 - 2)^2 + (x_2 - 2)^2 = 1 \\ &A = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \\ &-4 \leq x_i \leq 4,\ i = 1, 2 \end{aligned} \tag{A.3}$$

- **P4:**

$$\begin{aligned} \max f(x) \quad &= (\sqrt{n})^n \prod_{i=1}^{n} x_i \\ \text{Sujeito à:} \quad &\begin{cases} \sum_{i=1}^{n} x_i^2 - 1 = 0 \\ 0 \leq x_i \leq 1, \ i = 1, 2, \ldots, n \end{cases} \end{aligned} \tag{A.4}$$

  em que $n = 10$.

- **P5:**

$$\begin{aligned} &\min f(x) = x_1^2 + (x_2 - 1)^2 \\ &\text{Sujeito à: } x_2^2 - x_1 = 0 \\ &-1 \leq x_i \leq 1, \ i = 1, 2 \end{aligned} \tag{A.5}$$

- **P6:**

$$\min f(x) = (x_1 - 4)^2 + (x_2 - 6)^2 + (x_3 - 1)^2 + (x_4 - 2)^2 + (x_5 - 1)^2$$

$$\text{Sujeito à:} \begin{cases} x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ x_2 x_3 - 5 x_4 x_5 = 0 \\ x_1^3 + x_2^3 + 1 = 0 \\ -2.3 \leq x_i \leq 2.3, \ i = 1, 2, 3, 4 \\ -3.2 \leq x_i \leq 3.2, \ i = 5 \end{cases} \tag{A.6}$$

- **P7:**

$$\begin{aligned}\min f(x) =& x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ &+2(x_6 - 1)^2 + 5x_7 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45; \\ \text{Sujeito à: } &\begin{cases} 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 = 0 \\ 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 = 0 \\ x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 = 0 \\ -10 \leq x_i \leq 10,\ i = 1, ..., 10 \end{cases}\end{aligned} \tag{A.7}$$

- **P8:**

$$\begin{aligned}\min f(x) =& (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3x_7^4 - x_6x_7 - 10x_6 - 8x_7 \\ \text{Sujeito à: } &\begin{cases} 2x_1^2 + 3x_2^4 + x_3 + x_4^2 + 5x_5 - 127 = 0 \\ 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 = 0 \\ -10 \leq x_i \leq 10,\ i = 1, ..., 7 \end{cases}\end{aligned} \tag{A.8}$$

- **P9:**

$$\min f(x) = x_1$$

$$\text{Sujeito à: } \begin{cases} -x_1 + 35x_2^{0.6} + 35x_3^{0.6} = 0 \\ -300x_3 + 7500x_5 - 7500x_6 + 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0 \\ 100x_2 + 155.365x_4 + 2500x_6 - x_2x_4 + 25x_4x_7 - 15536.5 = 0 \\ -x_5 + ln(-x_4 + 900) = 0 \\ -x_6 + ln(x_4 + 300) = 0 \\ -x_7 + ln(-2x_4 + 700) = 0 \end{cases} \tag{A.9}$$

Em que $0 \leq x_1 \leq 1000, 0 \leq x_2 \leq 40, 0 \leq x_3 \leq 40, 100 \leq x_4 \leq 300, 6.3 \leq x_5 \leq 6.7, 5.9 \leq x_6 \leq 6.4$ e $4.5 \leq x_7 \leq 6.25$

- **P10:**

$$\min f(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \left(\frac{x_i}{\sum_{j=1}^{10}} \right) \right)$$

$$\text{Sujeito à: } \begin{cases} x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 \\ x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\ x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 \\ 0 \leq x_i \leq 10, \ i = 1, ..., 7 \end{cases} \tag{A.10}$$

Em que $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.100$, $c_8 = -10.708$, $c_9 = -26.662$ e $c_{10} = -22.179$.

- **P11:**

$$\min f(x) = x_1$$

$$\text{Sujeito à:} \begin{cases} -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} = 0 \\ x_5 - 100000x_8 + 1e07 = 0 \\ x_6 + 100000x_8 - 100000x_9 = 0 \\ x_7 + 100000x_9 - 5e07 = 0 \\ x_5 + 100000x_{10} - 3.3e07 = 0 \\ x_6 + 100000x_{11} - 4.4e07 = 0 \\ x_7 + 100000x_12 - 6.6e07 = 0 \\ x_5 - 120x_2x_{13} = 0 \\ x_6 - 80x_3x_14 = 0 \\ x_7 - 40x_4x_1 = 0 \\ x_8 - x_{11} + x_{16} = 0 \\ x_9 - x_12 + x_17 = 0 \\ -x_{18} + ln(x_{10} - 100) = 0 \\ -x_{19} + ln(-x_8 + 300)0 \\ -x_{20} + ln(x_{16}) = 0 \\ -x_{21} + ln(-x_9 + 400) = 0 \\ -x_{22} + ln(x_{17}) = 0 \\ -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0 \\ x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0 \\ x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0 \\ \leq x_i \leq 10, \ i = 1, ..., 7 \end{cases} \tag{A.11}$$

Em que $0 \leq x_1 \leq 20000, 0 \leq x_i \leq 1e06, i = (2, 3, 4), 0 \leq x_i \leq 4e07, i = (5, 6, 7), 100 \leq x_8 \leq 299.99, 100 \leq x_9 \leq 399.99, 100.01 \leq$

$x_{10} \leq 300,100 \leq x_{11} \leq 400,100 \leq x_{12} \leq 600,0 \leq x_{10} \leq 500, i = (13,14,15), 0.01 \leq x_{16} \leq 300, 0.01 \leq x_{17} \leq 400$ e $-4.7 \leq x_i \leq 6.25, = (18,...,22)$.

- **MP1:**

$$\begin{aligned} \min f_1(x) =& x_1^2 + x_2^2 \\ \min f_2(x) =& (x_1 - 5)^2 + (x_2 - 5)^2 \\ & -5 \leq x_i \leq 10, i = 1,2 \end{aligned} \tag{A.12}$$

- **MP2:**

$$\begin{aligned} \min f_1(x) =& x_1 \\ g(x) =& 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(x) =& 1 - \sqrt{f_1/g} \\ \min f_2(x) =& h * g \\ & \text{Sujeito à: } 0 \leq x_i \leq 1,\ i = 1,...,m \end{aligned} \tag{A.13}$$

- **MP3:**

$$\begin{aligned} \min f_1(x) =& x_1 \\ g(x) =& 1 + 9 \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(x) =& 1 - (f_1/g)^2 \\ \min f_2(x) =& h * g \\ & \text{Sujeito à: } 0 \leq x_i \leq 1,\ i = 1,...,m \end{aligned} \tag{A.14}$$

- **MP4:**

$$\begin{aligned}
\min f_1(x) =& x_1 \\
g(x) =& 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1} \\
h(x) =& 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1) \\
\min f_2(x) =& h * g \\
\text{Sujeito à: } & 0 \leq x_i \leq 1,\ i = 1, ..., m
\end{aligned} \tag{A.15}$$

- **MP5:**

$$\begin{aligned}
\min f_1(x) =& x_1 \\
g(x) =& 1 + 10(m-1) + \sum_{i=2}^{m} x_i^2 - 10cos(4\pi x_i) \\
h(x) =& 1 - \sqrt{f_1/g} \\
\min f_2(x) =& h * g \\
\text{Sujeito à: } & \begin{cases} 0 \leq x_1 \leq 1 \\ -5 \leq x_i \leq 5,\ i = 2, ..., m \end{cases}
\end{aligned} \tag{A.16}$$

- **MP6:**

$$\begin{aligned}
\min f_1(x) =& x(1) \\
g(x) =& 1 + 9 * \sum_{i=2}^{m} \frac{x_i}{m-1}; \\
h(x) =& 1 - (f_1/g)^{0.25} - (f_1/g)^4 \\
\min f_2(x) =& h * g \\
\text{Sujeito à: } & 0 \leq x_i \leq 1, \forall i
\end{aligned} \tag{A.17}$$

Referências Bibliográficas

Adra, Salem F, Dodd, Tony J, Griffin, Ian A, & Fleming, Peter J. 2009. Convergence acceleration operator for multiobjective optimization. *Ieee transactions on evolutionary computation*, **13**(4), 825–847.

Binh, To Thanh, & Korn, Ulrich. 1996. An evolution strategy for the multiobjective optimization. *Pages 23–28 of: Pro. 2nd int. conf. genetic algorithms*.

Bonyadi, Mohammad Reza, & Michalewicz, Zbigniew. 2017. *Particle swarm optimization for single objective continuous space problems: a review*.

Böhning, D, & Lindsay, B G. 1998. Monotonicity of quadratic-approximation algorithms. *Annals of the institute of statistical mathematics*, **40**(4), 641–663.

Carothers, N L. 1998. *A short course on approximation theory*. Department of Mathematics and Statistics, Bowling Green State University.

Chiba, Kazuhisa, Obayashi, Shigeru, Nakahashi, Kazuhiro, & Morino, Hiroyuki. 2005. High-fidelity multidisciplinary design optimization of wing shape for regional jet aircraft. *Pages 621–635 of: International conference on evolutionary multi-criterion optimization*. Springer.

Coello, CA Coello. 2006. Evolutionary multi-objective optimization: a historical view of the field. *Ieee computational intelligence magazine*, **1**(1), 28–36.

Coello, Carlos A Coello. 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, **191**(11-12), 1245–1287.

Coello, Carlos A Coello, Lamont, Gary B, Van Veldhuizen, David A, *et al.* . 2007. *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. Springer.

Coleman, Thomas F, & Li, Yuying. 1996. An interior trust region approach for nonlinear minimization subject to bounds. *Siam journal on optimization*, **6**(2), 418–445.

Côté, Pascal, Wong, Tony, & Sabourin, Robert. 2004. A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. *Pages 294–312 of: International conference on the practice and theory of automated timetabling*. Springer.

da Cruz, André R, Wanner, Elizabeth F, Cardoso, Rodrigo TN, & Takahashi, Ricardo HC. 2011. Using convex quadratic approximation as a local search operator in evolutionary multiobjective algorithms. *Pages 1217–1224 of: Evolutionary computation (cec), 2011 ieee congress on*. IEEE.

Deb, Kalyanmoy, & Goel, Tushar. 2001. A hybrid multi-objective evolutionary approach to engineering shape design. *Pages 385–399 of: International conference on evolutionary multi-criterion optimization*. Springer.

Deep, Kusum, & Bansal, Jagdish Chand. 2009. Hybridization of particle swarm optimization with quadratic approximation. *Opsearch*, **46**(1), 3–24.

Deep, Kusum, & Das, Kedar Nath. 2008. Quadratic approximation based hybrid genetic algorithm for function optimization. *Applied mathematics and computation*, **203**(1), 86–98.

Dwass, Meyer. 1957. Modified randomization tests for nonparametric hypotheses. *The annals of mathematical statistics*, 181–187.

Eberhart, R., & Kennedy, J. 1995. A new optimizer using particle swarm theory. *Pages 39–43 of: Micro machine and human science, 1995. mhs'95., proceedings of the sixth international symposium on*.

Ehrgott, Matthias. 2012. Vilfredo pareto and multi-objective optimization. *Doc. math*, 447–453.

Fonseca, Carlos H, & Wanner, Elizabeth Fialho. 2016. A quadratic approximation-based local search operator for handling two equality constraints in continuous optimization problems. *Pages 4911–4917 of: Evolutionary computation (cec), 2016 ieee congress on*. IEEE.

Fonseca, Carlos M, & Fleming, Peter J. 1995. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, **3**(1), 1–16.

Fonseca, Leonardo G, Bernardino, Heder S, & Barbosa, Helio JC. 2012. A genetic algorithm assisted by a locally weighted regression surrogate model. *Pages 125–135 of: International conference on computational science and its applications*. Springer.

Freiberger, Marianne. 2011. *Convex is complex*. https://plus.maths.org/content/convexity.

Friedman, J H. 2001. Greedy function approximation: A gradient boosting machine. *The annals of statistics*, **29**(5), 1189 – 1232.

Goel, Tushar, & Deb, Kalyanmoy. 2002. Hybrid methods for multi-objective evolutionary algorithms. *Pages 188–192 of: Proceedings of the fourth asia-pacific conference on simulated evolution and learning (seal'02)*.

Greiner, David, Winter, Gabriel, Emperador, José M, & Galván, Blas. 2005. Gray coding in evolutionary multicriteria optimization: application in frame structural optimum design. *Pages 576–591 of: International conference on evolutionary multi-criterion optimization*. Springer.

Hancock, Peter. 1994. An empirical comparison of selection methods in evolutionary algorithms. *Evolutionary computing*, 80–94.

Hanne, Thomas, & Nickel, Stefan. 2005. A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *European journal of operational research*, **167**(3), 663–678.

Huband, Simon, Hingston, Philip, Barone, Luigi, & While, Lyndon. 2006. A review of multiobjective test problems and a scalable test problem toolkit. *Ieee transactions on evolutionary computation*, **10**(5), 477–506.

Jankovic, V. 2005. Quadratic functions in several variables. *The teachings of mathematics*, **8**(2), 53–60.

Jin, Yaochu, Olhofer, Markus, & Sendhoff, Bernhard. 2001. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? *Pages 1042–1049 of: Proceedings of the 3rd annual conference on genetic and evolutionary computation*. Morgan Kaufmann Publishers Inc.

Knowles, Joshua, & Corne, David. 2002. On metrics for comparing nondominated sets. *Pages 711–716 of: Evolutionary computation, 2002. cec'02. proceedings of the 2002 congress on*, vol. 1. IEEE.

Konstantinidis, Andreas, & Yang, Kun. 2011. Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific moea/d. *Applied soft computing*, **11**(6), 4117–4134.

Lahanas, Michael. 2004. Application of multiobjective evolutionary optimization algorithms in medicine. *Applications of multi-objective evolutionary algorithms*, 365–391.

Liang, JJ, Runarsson, Thomas Philip, Mezura-Montes, Efren, Clerc, Maurice, Suganthan, Ponnuthurai Nagaratnam, Coello, CA Coello, & Deb, Kalyanmoy. 2006. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of applied mechanics*, **41**(8).

Lourenço, Helena R, Martin, Olivier C, & Stützle, Thomas. 2003. Iterated local search. *Pages 320–353 of: Handbook of metaheuristics*. Springer.

Luenberger, David G. 2003. *Linear and nonlinear programming*. Springer.

Mladenović, Nenad, & Hansen, Pierre. 1997. Variable neighborhood search. *Computers & operations research*, **24**(11), 1097–1100.

Moghaddam, Amjad Anvari, Seifi, Alireza, Niknam, Taher, & Pahlavani, Mohammad Reza Alizadeh. 2011. Multi-objective operation management of a renewable mg (micro-grid) with back-up micro-turbine/fuel cell/battery hybrid power source. *Energy*, **36**(11), 6490–6507.

Moré, Jorge J. 1978. The levenberg-marquardt algorithm: implementation and theory. *Pages 105–116 of: Numerical analysis*. Springer.

Mota, Felipe O, Wanner, Elizabeth F, Luz, Eduardo JS, & Moreira, Gladston JP. 2018. Vnd-based local search operator for equality constraint problems in pso algorithm. *Electronic notes in discrete mathematics*, **66**, 111–118.

Mumford, D, & Shah, J. 1989. Likelihood and higher-tions to tail areas: A review and annotated bibliography. *Communications on pure and applied mathematics*, **17**, 577–685.

Peconick, Gustavo, Wanner, Elizabeth F, & Takahashi, Ricardo HC. 2007. Projection-based local search operator for multiple equality constraints within genetic algorithms. *Pages 3043–3049 of: Evolutionary computation, 2007. cec 2007. ieee congress on*. IEEE.

Pirlot, Marc. 1996. General local search methods. *European journal of operational research*, **92**(3), 493–511.

Poli, Riccardo. 2008. Analysis of the publications on the applications of particle swarm optimisation. *Journal of artificial evolution and applications*, **2008**.

Radcliffe, Nicholas, & Surry, Patrick. 1994. Formal memetic algorithms. *Evolutionary computing*, 1–16.

Reid, N. 1996. Optimal approximations by piecewise smooth functions and associated variational problems. *The canadian journal of statistics*, **24**(2), 141–166.

Rivas-Dávalos, Francisco, & Irving, Malcolm R. 2005. An approach based on the strength pareto evolutionary algorithm 2 for power distribution system planning. *Pages 707–720 of: International conference on evolutionary multi-criterion optimization*. Springer.

Rutten, David. 2011. *Evolutionary solvers: Fitness functions*. https://ieatbugsforbreakfast.wordpress.com/2011/03/04/fitness-functions/.

Schott, Jason R. 1995. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Tech. rept. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH.

Shafer, R E. 1974. On quadratic approximation. *Siam journal on numerical analysis*, **11**(2), 447–460.

Shi, Yuhui, & Eberhart, Russell. 1998. A modified particle swarm optimizer. *Pages 69–73 of: Evolutionary computation proceedings, 1998. ieee world congress on computational intelligence., the 1998 ieee international conference on*. IEEE.

Singh, Dipti, & Agrawal, Seema. 2015. Hybridization of self organizing migrating algorithm with quadratic approximation and non uniform mutation for function optimization. *Pages 373–387 of: Proceedings of fourth international conference on soft computing for problem solving*. Springer.

Sinha, Ankur, Malo, Pekka, & Deb, Kalyanmoy. 2013. Efficient evolutionary algorithm for single-objective bilevel optimization. *arxiv preprint arxiv:1303.3901*.

Student. 1908. Probable error of a correlation coefficient. *Biometrika*, 302–310.

Van Veldhuizen, David A, & Lamont, Gary B. 1998. Evolutionary computation and convergence to a pareto front. *Pages 221–228 of: Late breaking papers at the genetic programming 1998 conference*.

Van Veldhuizen, David A, & Lamont, Gary B. 2000. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary computation*, **8**(2), 125–147.

Wanner, Elizabeth F, Guimarães, Frederico G, Saldanha, Rodney R, Takahashi, Ricardo HC, & Fleming, Peter J. 2005. Constraint quadratic approximation operator for treating equality constraints with genetic algorithms. *Pages 2255–2262 of: Evolutionary computation, 2005. the 2005 ieee congress on*, vol. 3. IEEE.

Wanner, Elizabeth F, Guimarães, Frederico G, Takahashi, Ricardo HC, & Fleming, Peter J. 2008. Local search with quadratic approximations into memetic algorithms for optimization with multiple criteria. *Evolutionary computation*, **16**(2), 185–224.

Wanner, Elizabeth Fialho. 2006. *Operadores para algoritmos genéticos baseados em aproximações quadráticas de funções de variáveis contínuas*. UFMG.

Yeniay, Özgür. 2005. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and computational applications*, **10**(1), 45–56.

Zitzler, Eckart. 1999. Evolutionary algorithms for multiobjective optimization: Methods and applications.

Zitzler, Eckart, Deb, Kalyanmoy, & Thiele, Lothar. 2000. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, **8**(2), 173–195.