Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação

Um algoritmo de estimação de distribuição para otimização multiobjetivo baseado em colônia de abelhas e clusters

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto, como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

Aluno: Fabiano Tomás Novais

Orientador: Frederico Gadelha Guimarães

Coorientador: Agnaldo José de Rocha Reis

Ouro Preto – MG
2013

Um algoritmo de estimação de distribuição para otimização multiobjetivo baseado em colônia de abelhas e clusters

Fabiano Tomás Novais

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto, como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

Aprovada por:

Frederico Gadelha Guimarães, D.Sc. / DEE-UFMG
(Orientador)

Agnaldo José de Rocha Reis, D.Sc. / DECAT-UFOP
(Coorientador)

Lucas de Souza Batista, D.Sc. / DEE-UFMG

Haroldo Gambini Santos, D.Sc. / DECOM-UFOP

Ouro Preto, 31 de outubro de 2013.

N935a Novais, Fabiano Tomás.
Um algoritmo de estimação de distribuição para otimização multiobjetivo baseado em colônia de abelhas e clusters / Fabiano Tomás Novais. – 2013.
94 f.: il. color.; grafs.; tabs.

Orientador: Prof. Dr. Frederico Gadelha Guimarães.

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: Otimização e Inteligência Computacional.

1. Otimização combinatória - Teses. 2. Abelha (Inteligência de enxames) - Teses. 3. Algoritmos de computador - Teses. I. Guimarães, Frederico Gadelha. II. Universidade Federal de Ouro Preto. III. Título.

CDU: 004.421.2

Catalogação: sisbin@sisbin.ufop.br

“Pequena é a abelha entre o seres alados, o que produz, entretanto, é o que há de mais doce.”(Eclesiástico 11:3)

Agradecimentos

Inicialmente gostaria de agradecer a Deus por estar presente em todos momentos de minha vida, principalmente naqueles de maior dificuldade.

Agradeço também a todas pessoas que diretamente ou indiretamente contribuíram para a conclusão deste trabalho.

Em especial agradeço aos meus pais Benjamim Constant por ter me ensinado os valores do trabalho e a ser persistente e a minha mãe Maria Eduarda por acreditar em mim, por todas suas orações e por ter me ensinado vários valores que carrego comigo.

Agradeço também a minha querida esposa Aline Silva por toda sua paciência, dedicação e amor, os quais deram grande motivação para a conclusão deste trabalho. Sou grato também a meus familiares e a família de minha esposa que me acolheram.

Aos meus orientadores, o Frederico Gadelha e Agnaldo José que acreditaram e deram suporte ao desenvolvimento deste trabalho.

Agradeço também a todos funcionários do NTI pela amizade e pelos momentos de descontração e aprendizado.

Finalmente, agradeço a todas as amizades feitas nesta longa e bela jornada que chamamos de vida.

Resumo

Neste trabalho, propõem-se um novo algoritmo híbrido denominado *Multiobjective Optimization Estimation of Distribution Algorithm based on Bee Colonies and Clusters* (MOEDABC) para resolução de problemas de otimização multiobjetivo de larga escala no domínio contínuo. Este algoritmo é inspirado na organização de uma colônia de abelhas e baseia-se nos algoritmos de estimação de distribuição. Como forma de gerar melhores soluções utiliza-se também técnicas de clusterização com a finalidade de aumentar a convergência local das soluções na fronteira Pareto. O algoritmo é baseado em quatro tipos de abelhas: as campistas, as observadoras, as nutrizes e as escoteiras, onde cada uma utiliza uma forma diferente de gerar as novas soluções. Combinando diferentes técnicas como clusterização, estimação de distribuição e algoritmos genéticos possibilitou-se um melhor aprendizado por meio de modelos probabilísticos baseados em distribuições Gaussianas e de Cauchy, obtendo assim soluções de maior qualidade. Em busca de obter maior flexibilidade do algoritmo na resolução de problemas foi introduzido um feromônio de controle responsável por controlar a proporção de cada tipo de abelhas na colônia. Comparado com outros algoritmos os resultados obtidos demonstram que o algoritmo proposto apresenta uma maior velocidade de convergência e uma melhor distribuição das soluções na fronteira Pareto conforme os indicadores utilizados.

PALAVRAS-CHAVE: Otimização Multiobjetivo, Inteligência de enxames, Algoritmos híbridos, Algoritmos de Estimação de Distribuição e *Clusters*.

Abstract

In this paper, are proposed a new hybrid optimization algorithm denominated Multiobjective Estimation of Distribution Algorithm based on Bee Colonies and Clusters (MOEDABC) to solve large scale multi-objective optimization problems in continuous domain. This algorithm is inspired in the organization of a bee colony and is based on estimation of distribution algorithms. As a way to generate better solutions also employ the clustering methods in order to increase the local convergence of the solutions in the Pareto front . The algorithm is based in four types of bees, the employer, the onlookers, the nursings and scouts, each a of which uses differents way of generating new solutions . Combining different techniques such as clustering, estimation of distribution algorithms and genetic algorithms was possible a better learning through probabilistic models based on Gaussian distributions and Cauchy, thus obtaining higher quality solutions . In search of greater flexibility of the algorithm in solving problems we introduce a pheromone control that is responsible for controlling the proportion of each type of bees in the colony . Compared with other algorithms the results obtained show that the proposed algorithm shows a faster convergence and a better distribution of solutions in the front Pareto according to the metrics used.

Keywords: Multiobjective Optimization, Swarm Intelligence, Hybrid Algorithm, Estimation of Distribution Algorithms and Clusters

Sumário

Lista de Figuras **x**

Lista de Algoritmos **xi**

Lista de Tabelas **xii**

Lista de Abreviaturas e Siglas **xiii**

1 Introdução **1**
1.1 Objetivos . . . 3
1.1.1 Objetivo Geral . . . 3
1.1.2 Objetivos Específicos . . . 3
1.2 Motivação . . . 3
1.3 Estrutura . . . 4

2 Fundamentos de Otimização **6**
2.1 Otimização Multiobjetivo . . . 6
2.1.1 Definição . . . 6
2.1.2 Pareto dominância . . . 7
2.1.3 Solução Pareto-ótima . . . 7
2.1.4 Conjunto Pareto-ótimo . . . 8
2.1.5 Fronteira Pareto . . . 8
2.2 ϵ-Dominância . . . 8
2.2.1 Definição . . . 9
2.2.2 Fronteira Pareto-ótima ϵ-aproximada . . . 9

3 Revisão Bibliográfica **10**
3.1 Inteligência de enxames . . . 10
3.2 Algoritmos Multiobjetivos . . . 11
3.2.1 NSGA-II . . . 11
3.2.2 MOABC . . . 13
3.2.3 MOEA/D . . . 16
3.2.4 GDE3 . . . 17
3.3 Algoritmos de estimação de distribuição . . . 17
3.3.1 Algoritmo UMDA$_c^G$ . . . 18
3.3.2 RECEDA . . . 19
3.3.3 Covariance Matrix Repairing . . . 19
3.4 Operador de Mutação de Cauchy . . . 20

3.5 Clusterização . . . 21
3.5.1 K-Means . . . 21
3.6 Framework MOEA . . . 21
3.6.1 Executor . . . 23
3.6.2 Instrumenter . . . 23
3.6.3 Analyzer . . . 25
3.7 Trabalhos Relacionados . . . 25
3.8 Considerações Finais . . . 26

4 Metodologia **27**
4.1 Sociedade das abelhas . . . 27
4.2 Inicializando a população . . . 28
4.3 Dividindo o espaço objetivo em clusters . . . 28
4.4 Abelhas Campistas . . . 29
4.5 Abelhas Observadoras . . . 30
4.6 Abelhas Nutrizes . . . 31
4.7 Abelhas Escoteiras . . . 31
4.8 Arquivo Externo . . . 31
4.9 Controle da População . . . 32
4.10 Algoritmos . . . 34

5 Experimentos **38**
5.1 Indicadores . . . 38
5.1.1 Additive Epsilon Indicator . . . 38
5.1.2 Hypervolume . . . 38
5.1.3 Generational Distance . . . 39
5.1.4 Inverted Generational Distance . . . 39
5.1.5 Maximum Pareto Front Error . . . 40
5.1.6 Spacing . . . 41
5.2 Problemas . . . 41
5.3 Testes estatísticos . . . 41
5.3.1 Teste Mann-Whitney . . . 41
5.3.2 Teste Kruskal-Wallis . . . 42
5.3.3 Indicadores dos testes . . . 42
5.4 Parâmetros adotados . . . 43

6 Resultados Computacionais **44**
6.1 Análises baseadas nos indicadores . . . 44
6.1.1 Additive Epsilon Indicator . . . 44
6.1.2 Hypervolume . . . 47
6.1.3 Inverted Distance . . . 47
6.1.4 Maximum Pareto Front Error . . . 47
6.1.5 Spacing . . . 54
6.2 Análise gráfica dos resultados . . . 57
6.3 Considerações Finais . . . 65

7 Conclusões e Trabalhos Futuros **66**
7.1 Trabalhos Futuros . . . 67

A Publicações **68**

B Resultados Completos **69**

Referências **75**

Lista de Figuras

2.1 Pareto dominância adaptado de Engelbrecht (2006) 7
2.2 Espaço Decisão e Objetivo adaptado de Zitzler et al. (2004) 8
2.3 ϵ-Dominância adaptado de Engelbrecht (2006) 9

3.1 Operador de diversidade crowding distance utilizado no NSGA-II . . 13
3.2 Clusters gerados pelo *K-means* com k=5 22
3.3 Diagrama simplificado do MOEA 24

4.1 Variáveis do espaço de decisão associados a 3 clusters do espaço objetivo 29

5.1 Additive Epsilon Indicator 39
5.2 Hipervolume adaptado de Knowles e Corne (2002) 40

6.1 Problema ZDT1 57
6.2 Problema ZDT2 58
6.3 Problema ZDT3 58
6.4 Problema ZDT4 59
6.5 Problema ZDT6 59
6.6 Problema DTLZ2 60
6.7 Problema DTLZ4 61
6.8 Problema DTLZ7 61
6.9 Problema UF1 62
6.10 Problema UF2 63
6.11 Problema UF3 64
6.12 Problema UF4 64
6.13 Problema UF6 65

Lista de Algoritmos

1 Fast non-dominated sorting 12
2 Crowding Distance 14
3 RECEDA 19
4 CMR 20

5 Control Pheromone 33

6 MOEDABC 35
7 MOEDABC^{Phe} 36
8 UMDA$_c^{GC}$ 37

Lista de Tabelas

6.1 Problemas da classe ZDT: Indicador Additive Epsilon Indicator . . . 45
6.2 Problemas da classe UF: Indicador Additive Epsilon Indicator 46
6.3 Problemas da classe ZDT: Indicador Hypervolume 48
6.4 Problemas da classe UF: Indicador Hypervolume 49
6.5 Problemas da classe ZDT: Indicador Inverted Distance 50
6.6 Problemas da classe UF: Indicador Inverted Distance 51
6.7 Problemas da classe ZDT: Indicador Maximum Pareto Front Error . . 52
6.8 Problemas da classe UF: Indicador Maximum Pareto Front Error . . 53
6.9 Problemas da classe ZDT: Indicador Spacing 55
6.10 Problemas da classe UF: Indicador Spacing 56

B.1 Problemas da classe DTLZ: Indicador Additive Epsilon Indicator . . . 70
B.2 Problemas da classe DTLZ: Indicador Hypervolume 71
B.3 Problemas da classe DTLZ: Indicador Inverted Distance 72
B.4 Problemas da classe DTLZ: Indicador Maximum Pareto Front Error . 73
B.5 Problemas da classe DTLZ: Indicador Spacing 74

Lista de Abreviaturas e Siglas

ABC: ARTIFICIAL BEE COLONY

ABCRK: ARTIFICIAL BEE COLONY WITH RANDOM KEYS

ACO: ANT COLONY OPTIMIZATION

CCEAS: COOPERATIVE COEVOLUTIONARY ALGORITHMS

CDF: CUMULATIVE DISTRIBUTION FUNCTION

CGA: COMPACT GENETIC ALGORITHM

CMR: COVARIANCE MATRIX REPAIRING

CR: CROSSOVER CONTROL PARAMETER

DE: DIFFERENCIAL EVOLUTION

EA: EVOLUTION ALGORITHM

EDAs: ESTIMATION OF DISTRIBUTION ALGORITHMS

EGS: EVOLUTIONARY GRADIENT SEARCH

F: MUTATION FACTOR

GA:GENETIC ALGORITHM

GD: GENERATIONAL DISTANCE

GDE: GENERALIZED DIFFERENTIAL EVOLUTION

ICFMO: IN-CORE FUEL MANAGEMENT OPTIMIZATION

IGD: INVERTED GENERATIONAL DISTANCE

HBOA: HIERARCHICAL BAYESIAN OPTIMIZATION ALGORITHM

MOABC: MULTIOBJECTIVE ARTIFICIAL BEE COLONY

MOEA/D: MULTIOBJECTIVE EVOLUTIONARY ALGORITHM BASED ON DECOMPOSITION

MOEDABC: MULTIOBJECTIVE OPTIMIZATION ESTIMATION OF DISTRIBUTION ALGORITHMS BASED ON BEE COLONIES AND CLUSTERS

MOEDABC^{Phe}: MULTIOBJECTIVE OPTIMIZATION ESTIMATION OF DISTRIBUTION ALGORITHMS BASED ON BEE COLONIES AND CLUSTERS WITH PHEROMONE

MOO: MULTIOBJECTIVE OPTIMIZATION

MOPC: MULTIOBJECTIVE PROBABILITY COLLECTIVES

MOPED: MULTI-OBJECTIVE PARZEN-BASED ESTIMATION OF DISTRIBUTION ALGORITHM FOR CONTINUOUS PROBLEMS

MPFE: MAXIMUM PARETO FRONT ERROR

NSGA-II:NONDOMINATED SORTING GENETIC ALGORITHM

PBIL: POPULATION BASED INCREMENTAL LEARNING

PC: PROBABILITY COLLECTIVES

PSO: PARTICLE SWARM OPTIMIZATION

RECEDA: REAL CODED ESTIMATION OF DISTRIBUTION ALGORITHM

SDS: ALGORITHM STOCHASTIC DIFFUSION SEARCH

UMDA$_c^{GC}$: UNIVARIATE MARGINAL DISTRIBUTION ALGORITHM FOR GAUSSIAN MODELS

Capítulo 1

Introdução

Muitos algoritmos já foram propostos para otimização multiobjetivo, onde o problema a ser resolvido é composto de dois ou mais objetivos, um conjunto de variáveis de decisão e um conjunto de restrições. Tais algoritmos visam encontrar as variáveis que minimizam (ou maximizam se o problema for de maximização) as funções objetivos de forma a encontrar um conjunto de soluções ou balanço de compromissos que satisfaça o conjunto de restrições do problema. O conjunto ótimo dos vetores de decisão para estes problemas é denominado conjunto Pareto-ótimo, já o conjunto dos vetores objetivos obtidos a partir do conjunto Pareto-ótimo correspondem a Fronteira Pareto.

Assim o objetivo de um algoritmo de otimização multiobjetivo consiste em encontrar um conjunto de vetores no espaço de decisão que pertença ao conjunto Pareto-ótimo ou a um conjunto Pareto-ótimo aproximado, onde neste caso, as soluções no espaço objetivo estão próximas a Fronteira Pareto.

Problemas como estes envolvendo vários objetivos são vistos em diversas áreas da ciência, os quais em geral são conflitantes entre si e apresentam uma grande complexidade de resolução. Como forma de lidar com estes problemas, técnicas baseadas na inteligência de enxames (*swarm intelligence*) vêm sendo aplicadas obtendo bons resultados. De acordo com Thampi (2009), a inteligência de enxames consiste na inteligência coletiva que emerge de grupos de agentes simples que atuam no meio utilizando regras locais para governar suas ações e interações com o grupo como forma de alcançar seus objetivos.

Dentre os algoritmos de otimização baseados na inteligência de enxames tem-se os de Enxames de Partículas (Particle Swarm Optimization - PSO) (Kennedy e Eberhart, 1995), Colônia de Formigas (Ant Colony Optimization - ACO) Dorigo e Di Caro (1999), Colônia de Abelhas (Artificial Bee Colony (ABC)) (Karaboga e Basturk, 2007), entre outros.

Devido a sua simplicidade e robustez o ABC vem sendo utilizado para resolver uma variedade de problemas complexos. Conforme Votano et al. (2011) o algoritmo ABC é um dos mais recentes membros da inteligência artificial. O ABC realiza um procedimento de busca estocástica combinada por busca a vizinhança baseado no comportamento das abelhas em busca por fontes de alimentos ou soluções. Estas soluções são modificadas com operadores genéticos como cruzamento e mutação com a finalidade de encontrar as soluções com maior qualidade. O ABC é um algoritmo mono-objetivo, assim Hedayatzadeh et al. (2010) propuseram uma versão do método

ABC multiobjetivo, o método *Multi-Objective Artificial Bee Colony* (MOABC) utilizando o método ϵ-dominância com um arquivo externo para manter as melhores soluções.

Na busca por melhores soluções muitos algoritmos utilizam algumas informações extraídas das soluções encontradas até então para o problema, tais informações podem ser obtidas analisando a formação de agrupamentos, ou clusters, no espaço objetivo e as correlações entre as variáveis e as funções objetivos.

Visto sua capacidade de explorar as correlações entre as variáveis do problema, os algoritmos baseados em estimação de distribuição (*Estimation of Distribution Algorithms* - EDAs) vêm ganhando atenção principalmente devido a sua capacidade de lidar com grandes números de variáveis. De acordo com Pelikan et al. (2012) estes algoritmos, também chamados de algoritmos evolucionários de estimação de densidade iterativos ou algoritmo genéticos baseado em modelos, visualizam a otimização como uma série de incrementais atualizações de um modelo probabilístico, começando por soluções admissíveis geradas por distribuições uniformes e finalizando com aquele modelo que gera apenas o modelo ótimo.

Conforme Zhang et al. (2007), a utilização de EDAs combinados com algoritmos evolucionários em um algoritmo híbrido pode levar a melhores desempenhos na solução de problemas de otimização de alto grau de complexidade e com estruturas complicadas guiando o processo de mutação e geração de novos sucessores.

Contudo, visto que muitas vezes certos algoritmos apresentam maior desempenho que outros na geração de novas soluções de determinados problemas e em determinados momentos no decorrer da exploração do espaço de decisão, torna-se interessante a utilização de algoritmos híbridos adaptativos os quais possam lidar com estes problemas com a finalidade de aumentar seu desempenho sem a necessidade de maiores ajustes e interferências.

Portanto a principal proposta deste trabalho consiste no desenvolvimento de um algoritmo híbrido denomidado Algoritmo de Otimização Multiobjetivo baseado em Colônias de Abelhas e *Clusters* (*Multiobjective Estimation of Distribution Algorithm based on Artificial Bee Colonies and Clusters* - MOEDABC) inspirado na organização das abelhas na divisão das tarefas da colônia. Nesta divisão atribuí-se papéis a cada tipo de abelha da colônia. Estes papéis são expressados pela utilização de determinados algoritmos de estimação de distribuição (EDAs) baseados nas informações obtidas a partir de modelos probabilísticos do conjunto de soluções e dos subconjuntos formados a partir da clusterização do espaço objetivo. Outro ponto importante deste algoritmo é a utilização de um feromônio de controle o qual permite ajustar a proporção de cada grupo de abelhas alocadas para explorar o espaço de busca ou de decisão.

Tais divisões visam melhorar a convergência das soluções e sua distribuição na fronteira Pareto evitando também convergências prematuras. Como forma de melhorar as distribuições das soluções na fronteira Pareto utiliza-se os métodos *Nondominated Sorting* e *Crowding-distance assignment* do NSGA-II e um arquivo externo onde são armazenadas as soluções não dominadas. Este arquivo externo também é utilizado na geração do feromônio de controle o qual permite ajustar a proporção de cada grupo de abelhas no decorrer da execução do algoritmo.

De forma a analisar o comportamento do algoritmo proposto na resolução de algumas classes de problemas, são realizadas análises baseadas em indicadores de

desempenho, os quais permitem comparar um conjunto solução obtido pelo MOEDABC com o de alguns algoritmos do estado da arte na resolução de problemas multiobjetivo de larga escala no domínio contínuo, considerando-se problemas de larga escala aqueles com um número de variáveis maior ou igual a 100. Além disso realiza-se também um teste comparativo do MOEDABC com e sem o feromônio de controle com a finalidade de verificar seu efeito no desempenho do algoritmo.

1.1 Objetivos

Neste capítulo são apresentados os principais objetivos deste trabalho.

1.1.1 Objetivo Geral

O objetivo deste trabalho consiste em desenvolver um algoritmo que possibilite resolver problemas multiobjetivo de larga escala e com um menor número de avaliações. Visto que o custo de avaliação da função objetivo apresenta um alto custo computacional, buscou-se então neste trabalho inspiração na organização de uma colônia de abelhas, nos algoritmos de estimação de distribuição e na clusterização do espaço de objetivo como forma de extrair maiores informações da população de soluções por meio de modelos probabilísticos, com o propósito de resolver classes de problemas que envolvam muitas variáveis e mais de um objetivo.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Realizar estudo bibliográfico acerca dos algoritmos multiobjetivo com enfoque especial nos bio-inspirados em enxames e naqueles baseados em estimação de distribuição;
- Estudar e documentar o *framework* MOEA;
- Desenvolver o algoritmo proposto baseado no *framework* MOEA, assim como os outros algoritmos utilizados nos testes comparativos;
- Utilizar problemas testes com muitas variáveis de dois ou três objetivos;
- Efetuar os testes dos algoritmos conforme os indicadores *Additive Epsilon Indicator*, *Hypervolume*, *Inverted Distance*, *Maximum Pareto Front Error* e *Spacing*;
- Executar testes estatísticos dos algoritmos a partir dos resultados obtidos.

1.2 Motivação

Muitas vezes quando observa-se a natureza tem-se a leve impressão de simplicidade na forma que ela lida com diversos problemas. Porém por trás de um pequeno organismo muitos pesquisadores passam anos tentando decifrar seu funcionamento

e sua interação com o meio. Por exemplo, alguns grupos de insetos como as abelhas e as formigas que apesar de terem um tamanho reduzido em relação a muitos animais de maior porte conseguem manter sociedades complexas com funções bem definidas, capazes de realizar proezas na engenharia e na otimização por meio de um conjunto de ações cujo objetivo é alcançar o melhor para a sobrevivência do grupo e das novas gerações.

Nestas sociedades uma das características responsáveis pelo grande sucesso destes animais na natureza consiste na divisão de tarefas. De acordo com Robinson (1992) a divisão de tarefas tem como característica chave a sua plasticidade, onde colônias respondem a mudanças internas e externas por ajustar as taxas de indivíduos trabalhando em diversas tarefas necessárias a sua manutenção. A evolução da complexidade da divisão de tarefas tem grande envolvimento com a especialização destes animais conforme sua idade (polietismo), mudança de formas (polimorfismo) e diferenças individuais de comportamento. Estudos também apontam a capacidade de certas espécies como as abelhas de reterem características associadas em formas mais primitivas as quais permitem a colônia responder rapidamente a mudanças. Estas características de divisão de tarefas, de mudança de comportamentos e de controle da população consistem de estratégias que podem ser muito úteis para desenvolver algoritmos mais flexíveis para resolução de problemas de otimização.

Assim ao buscar novas formas de desenvolver um algoritmo híbrido adaptativo observou-se nas características da divisão de tarefas uma forma interessante e simples de controlar a produção de novas soluções baseados em diferentes métodos aqui representados por 4 tipos de abelhas.

Por outro lado, ao procurar por métodos para extrair maiores informações das populações de soluções encontrou-se nos algoritmos de estimação de distribuição (EDAs) uma forma de obter estas informações utilizando-se modelos probabilísticos. Porém verificou-se que o emprego dos EDAs na resolução de problemas de otimização multiobjetivo ainda não é muito difundido, visto o alto custo computacional dos EDAs na estimação dos parâmetros das populações de soluções e a sua dificuldade em resolver determinados problemas. Contudo visto que muitos problemas apresentam um número relativamente grande de variáveis, o custo da função de avaliação torna-se computacionalmente cara justificando a utilização destes métodos. Além disso muitos destes problemas também apresentam correlações, tanto das variáveis no espaço de decisão quanto no espaço objetivo, as quais podem ser exploradas combinando aos EDAs determinadas técnicas como a clusterização.

Portanto, com base nestas ideias iniciou-se o desenvolvimento de um novo algoritmo híbrido para resolução de problemas otimização multiobjetivo inspirados na organização de uma colônia de abelhas, nos EDAs e na clusterização do espaço objetivo com foco em resolver problemas de larga escala com um número menor de avaliações da função objetivo.

1.3 Estrutura

Este trabalho está organizado da seguinte forma: No capítulo 2 faz-se uma introdução sobre otimização multiobjetivo, aos conceitos de Pareto dominância e ϵ-dominância. No capítulo 3 realiza-se um estudo sobre algoritmos baseados em enxa-

mes, um resumo sobre os algoritmos utilizados nos testes comparativos, um resumo sobre EDAs, e os algoritmos UMDA$_c^G$, RECEDA, CMR, os métodos do NSGA-II e a distribuição de Cauchy. Neste capítulo também introduz-se a ideia de clusters e o algoritmo *K-means*, realiza-se uma revisão sobre *frameworks* e apresenta-se o *framework* MOEA utilizado neste trabalho e concluí-se este capítulo mostrando alguns trabalhos relacionados. No capítulo 4 descreve-se os algoritmos MOEDABC e MOEDABC^{Phe} desenvolvidos e o UMDA$_c^{GC}$. No capítulo 5 descreve-se o planejamento experimental, os indicadores de desempenho e os problemas testes utilizados. No capítulo 6 apresentam-se os resultados experimentais e um comparativo dos algoritmos. Finalmente, no capítulo 7 concluí-se este trabalho.

Capítulo 2

Fundamentos de Otimização

2.1 Otimização Multiobjetivo

Otimização é um processo essencial em muitas aplicações como negócios, administração e engenharia. Nestas áreas, múltiplos e frequentemente conflitantes objetivos precisam ser satisfeitos. Solucionar tais problemas tradicionalmente consiste em converter todos os objetivos dentro de uma única função objetivo (Ngatchou et al., 2005). Porém conforme Srinivas e Deb (1994) a otimização multiobjetivo é muito diferente da otimização com um único objetivo, esta última tenta obter as variáveis de decisão cujo resultado na função objetivo corresponde o mínimo valor global ou o máximo valor global dependendo se o problema de otimização for de minimização ou maximização. No caso de múltiplos objetivos, pode não existir apenas uma solução a qual é melhor (global mínimo ou máximo) com respeito a todos objetivos. Em problemas de otimização multiobjetivo típicos, existe um conjunto de soluções no espaço objetivo que são superiores ao restante de soluções quando todos os objetivos são considerados simultaneamente, diferindo uma solução da outra no espaço em dois ou mais objetivos. Tais soluções são conhecidas como soluções Pareto-ótimas (*Pareto-optimal*) ou soluções não dominadas (*non domitaded solutions*) (Vira e Haimes, 1983).

2.1.1 Definição

Problemas de otimização multiobjetivo consistem em um conjunto de funções f, em geral conflitantes entre si, as quais devem ser minimizadas (ou maximizadas) a partir de um conjunto de variáveis x, denominado vetor de decisão, satisfazendo também a um conjunto de restrições de desigualdade g e de igualdade h conforme visto abaixo na equação 2.1.

$$min_x f(x) = (f_1(x), f_2(x), ..., f_n(x)) \in \mathbb{R}^n, x \in \mathcal{F} \tag{2.1}$$

$$\mathcal{F} = \begin{cases} f(x) = (f_1(x), f_2(x), ..., f_n(x)) \in \mathcal{O} \\ g(x) = (g_1(x), g_2(x), ..., g_k(x)) \leq 0 \\ h(x) = (h_1(x), h_2(x), ..., h_r(x)) = 0 \\ x = (x_1, x_2, ..., x_m) \in \mathcal{S} \end{cases}$$

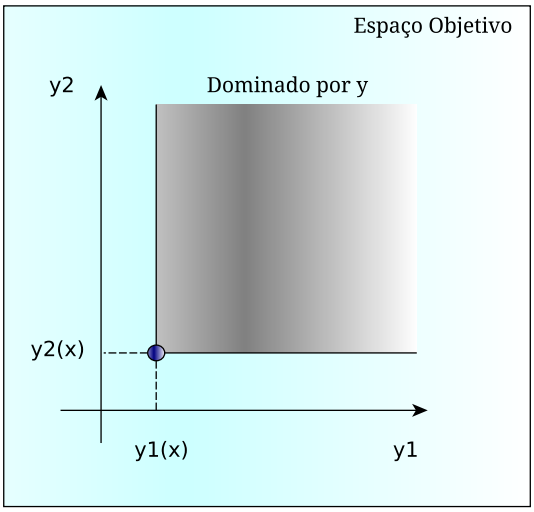


Figura 2.1: Pareto dominância adaptado de Engelbrecht (2006)

No espaço de busca pode-se definir $\mathcal{S} \subseteq \mathbb{R}^m$ em que m corresponde à dimensão do espaço de busca, e $\mathcal{F} \subseteq \mathcal{S}$ ao espaço factível, tal que $x = (x_1, x_2, ..., x_m) \in \mathcal{S}$ é o vetor de decisão e $f(x) = (f_1(x), f_2(x), ..., f_n(x)) \in \mathcal{O} \subseteq \mathbb{R}^n$ corresponde ao vetor objetivo contendo n funções objetivo de avaliação. $\mathcal{O}$ é referenciado como espaço objetivo e $\mathcal{S}$ como espaço de decisão (Rao, 2009) (Engelbrecht, 2006).

2.1.2 Pareto dominância

Seja $x_1 \in \mathcal{F}$ um vetor de decisão arbitrário, o vetor de decisão x_1 é dito ser não dominado em relação ao conjunto $\mathcal{F}' \subseteq \mathcal{F}$ se e somente se não existe um vetor x_2 em $\mathcal{F}$' o qual domine x_1; ou seja

$$\nexists x_2 \in \mathcal{F}' : x_2 \prec x_1 \tag{2.2}$$

Assim conforme Zitzler et al. (2000) o espaço de busca multiobjetivo é ordenado de tal forma que duas soluções arbitrárias estão relacionadas uma com a outra de duas formas possíveis: uma das duas domina a outra ou nenhuma domina. A figura 2.1 ilustra o conceito de dominância referente a um problema com dois objetivos.

2.1.3 Solução Pareto-ótima

Em geral não existe nenhum vetor x que minimiza todas as n funções objetivos simultaneamente. Portanto o conceito conhecido como solução Pareto-ótima é usado em problemas de otimização multiobjetivo (Rao, 2009). O vetor decisão x^* é Pareto-ótimo se e somente se x^* é não dominando em relação a todo $x \neq x^* \in \mathcal{F}$. Este vetor não pode ser melhorado em qualquer objetivo sem causar degradação em ao menos um dos outros objetivos; ele representa uma solução global ótima. Diz se então que a solução $y^* = f(x^*)$ é ponto Pareto-ótimo a qual não é dominado por nenhuma outra solução factível.

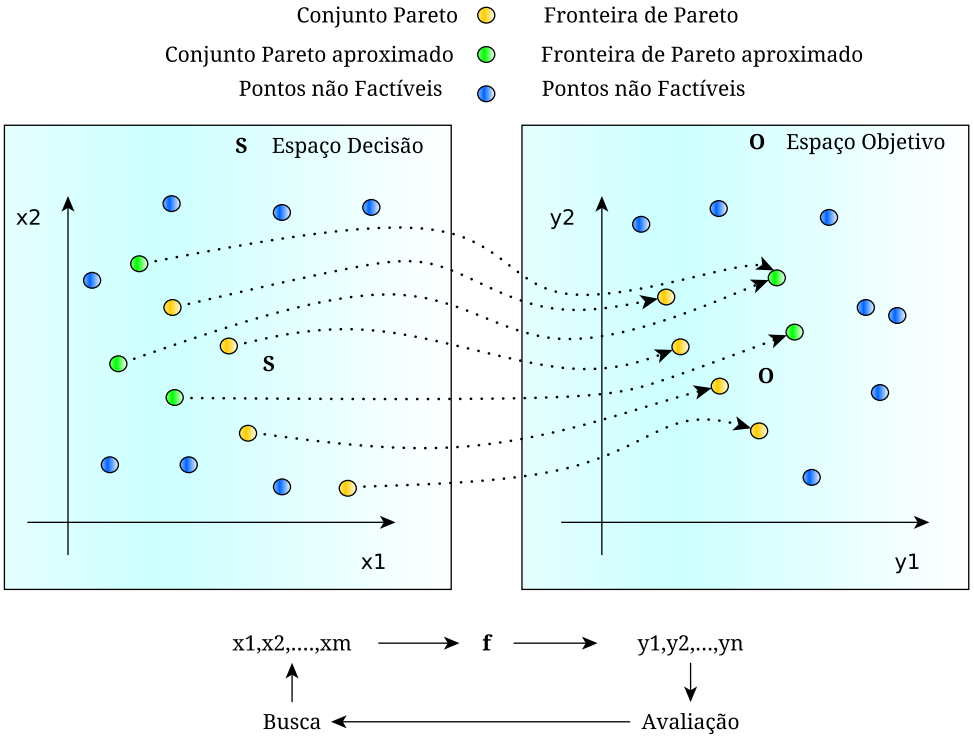


Figura 2.2: Espaço Decisão e Objetivo adaptado de Zitzler et al. (2004)

2.1.4 Conjunto Pareto-ótimo

O conjunto de todos vetores de Decisão Pareto-ótimos formam o conjunto Pareto-ótimo, $\mathcal{P}^*$. Isto é,

$$\mathcal{P}^* = \{x^* \in \mathcal{F} | \nexists x \in \mathcal{F} : x \prec x^*\} \tag{2.3}$$

O conjunto Pareto-ótimo portanto contém o conjunto de soluções, ou o balanço dos compromissos, para o problema de otimização multiobjetivo (MOP).

2.1.5 Fronteira Pareto

Dado o vetor objetivo , $f(x)$, e o conjunto Pareto-ótimo, P^*, então a fronteira Pareto-ótima, $\mathcal{PF}^* \subseteq \mathcal{O}$, pode ser definida como

$$\mathcal{PF}^* = \{f = (f_1(x^*), f_2(x^*), ..., f_n(x^*)) | x^* \in \mathcal{P}^*\} \tag{2.4}$$

A fronteira Pareto portanto contém todas os vetores objetivos correspondentes aos de decisão os quais não são dominados por qualquer outro vetor de decisão. Um exemplo da fronteira Pareto e das demais definições é ilustrado na figura 2.2.

2.2 ϵ-Dominância

Com a finalidade de melhorar a característica de convergência para fronteira Pareto-ótimo e ao mesmo tempo garantir o espalhamento das soluções na fronteira Lau-

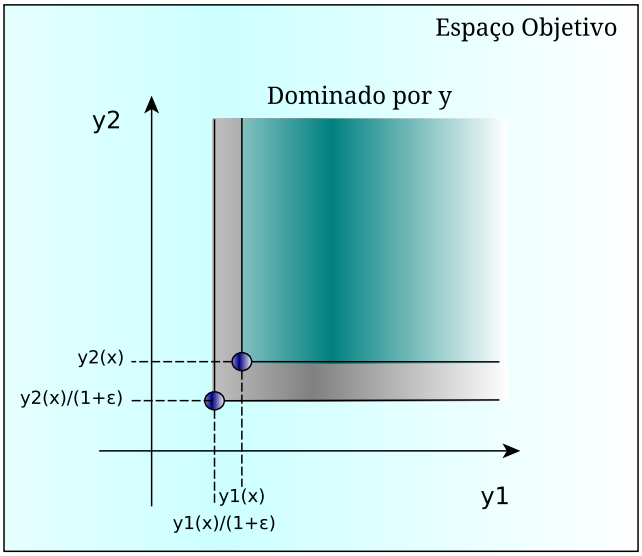


Figura 2.3: ϵ-Dominância adaptado de Engelbrecht (2006)

manns et al. (2002) propõem uma nova estratégia denominada $\epsilon-$Dominância como forma de garantir tais propriedades.

Esta estratégia de arquivamento consiste em manter um subconjunto das soluções geradas visando garantir a convergência e a diversidade de acordo com um critério definido pelo valor do parâmetro ϵ, o qual define a resolução de uma grade a ser adotada pela população que será arquivada. A ideia geral deste mecanismo é dividir o espaço objetivo dentro de caixas inversamente proporcionais a ϵ onde cada caixa é uma região geográfica que contém uma única solução(Hernández-Díaz et al., 2011).

2.2.1 Definição

Um vetor de decisão, x_1 ϵ-domina outro vetor de decisão, x_2 (denotado por $x_1 \prec_\epsilon x_2$), para algum $\epsilon > 0$, se e somente

$$f_k(x_1)/(1+\epsilon) \leq f_k(x_2), \forall k = 1, ..., n \tag{2.5}$$

$$\exists k = 1, ..., n : f_k(x_1)/(1+\epsilon) < f_k(x_2) \tag{2.6}$$

O conceito de ϵ-dominância é ilustrado na figura 2.3 e pode ser comparado com a figura 2.1 (Engelbrecht, 2006).

2.2.2 Fronteira Pareto-ótima ϵ-aproximada

Para um dado vetor objetivo, $f(x)$, um $\epsilon > 0$, a fronteira Pareto-ótima ϵ-aproximada, $\mathcal{PF}^*_\epsilon \subseteq \mathcal{O}$, contém todos valores dos vetores objetivos, onde $f_1 = f(x_1) \in \mathcal{O}$ os quais não são ϵ-dominados por qualquer outro valor do vetor objetivo, $f_2 = f(x_2) \in \mathcal{O}$. Isto é,

$$\mathcal{PF}^*_\epsilon = \{f = (f_1(x^*), f_2(x^*), ..., f_k(x^*)) | \nexists f(x) \in \mathcal{F} : f(x) \prec_\epsilon f(x^*)\} \tag{2.7}$$

Capítulo 3

Revisão Bibliográfica

3.1 Inteligência de enxames

Cada vez mais o homem vem buscando inspiração na natureza como forma de resolver problemas complexos, para isso muitos estudos vêm sendo feitos de forma a entender como certos mecanismos funcionam na natureza e como eles podem ser aproveitados para resolver problemas do nosso dia a dia. Muitas vezes quando se observa a natureza percebe-se que ela tende a otimizar estruturas, funções, sentidos de forma a permitir que os seres vivos e suas próximas gerações possam lidar de forma mais adequada com os problemas encontrados no ambiente. Para alcançar tais objetivos a natureza se baseia em alguns mecanismos como a mutação, o cruzamento e a seleção que atuam no processo de evolução cuja ideia introduzida por Charles Robert Darwin em seu livro "A Origem das Espécies" vem sendo utilizados para explicar tais fenômenos.

Contudo existem outros mecanismos utilizados na natureza para resolver problemas de forma eficiente, estes podem ser observados em diversos grupos de animais que atuam em grupos ou enxames utilizando os mecanismos de cooperação e interação associados com alguma forma de comunicação e controle descentralizado. Conforme Engelbrecht (2006) o comportamento de enxame proporcionam grandes benefícios em problemas onde não se tem um conhecimento global do ambiente. Nestas situações os indivíduos se baseiam no mecanismo de cooperação onde cada indivíduo dentro do grupo interage com os outros para solucionar um objetivo global baseando-se na troca de informações locais disponíveis cujo resultado final das interações no grupo é a solução do problema de forma mais eficiente do que se fosse feito por um único indivíduo.

Este tipo de comportamento denominado inteligência de enxame tem origem na inteligência de grupos de agentes simples e autônomos, onde um agente autônomo é um subsistema que interage com seu ambiente o qual geralmente consiste de outros agentes que atuam de forma independente um dos outros sem seguir os comandos de um líder (Thampi, 2009).

Algumas das principais características que permitem tais organismos lidarem com problemas complexos na natureza são a capacidade de auto organização, cooperação, diferenciação de funções que os possibilitando resolver estes problemas por meio do comportamento coletivo que emerge das interações dos indivíduos com ações que ao mesmo tempo correspondem a objetivos individuais e coletivas, como,

por exemplo, informar a melhor localização de alimento ou fugir de um predador.

Este tipo de comportamento emergente tem levado pesquisadores a utilizar enxames de agentes como técnicas de modelagem computacional e como ferramentas para estudo de sistemas complexos, algumas aplicações incluem negócios, economia, sistemas ecológicos e simulação de enxames de pássaros (Hinchey et al., 2007).

O objetivo destes modelos computacionais de inteligência de enxames é modelar simples comportamentos de indivíduos , suas interações locais com o ambiente e com seus vizinhos, de forma a obter comportamentos mais complexos que podem ser utilizados para resolver problemas, geralmente problemas de otimização (Engelbrecht, 2006).

Algumas aplicações da inteligência de enxames incluem um algoritmo híbrido utilizado por al Rifaie et al. (2012) para o esboço de desenhos com enfoque artístico baseado nos algoritmos *stochastic diffusion search* (SDS) o qual é inspirado no comportamento de algumas espécies de formigas (Leptothorax acervorum) forradeiras e no algoritmo *particle swarm optimiser*(PSO) que tenta imitar o comportamento de um grupo de pássaros.

Bonabeau e Meyer (2001) citam a utilização de um algoritmo baseado no comportamento de formigas pela *Southwest Airlines* em um problema de alocação de cargas permitindo uma redução de 80% da taxa de embarque de cargas e da carga de trabalho dos funcionários em cerca de 20%, além de uma redução dramática da transferência de cargas durante a noite.

De Oliveira e Schirru (2011) desenvolveram o algoritmo Artificial Bee Colony with Random Keys (ABCRK) para resolução de problemas combinatórios de difícil solução como o *In-Core Fuel Management Optimization* (ICFMO) da Engenharia Nuclear, o qual durante anos era apenas resolvido por especialistas.

Kıran et al. (2012) propõem dois novos modelos baseados nos algoritmos artificial bee colony (ABC) e no PSO para estimar a demanda de energia na Turquia, possibilitando uma projeção da demanda até 2025 de acordo com três cenários diferentes.

3.2 Algoritmos Multiobjetivos

3.2.1 NSGA-II

O algoritmo *Non-dominated Sorting* GA (NSGA) proposto por Srinivas e Deb Srinivas e Deb (1994) se baseia numa solução elitista para selecionar as melhores soluções, porém devido a ordem de complexidade computacional Deb, Agarwal e Meyarivan propuseram uma nova versão denominada NSGA-II onde um novo método de seleção chamado *fast non-dominated sorting* é sugerido como forma de diminuir a complexidade de $O(nN^3)$ para $O(nN^2)$, onde n é o número de objetivos e N é o tamanho da população. Ambos algoritmos baseiam no ranqueamento das soluções de uma população no espaço objetivo para o nível de não dominância, onde a primeira fronteira F_1 de ranque 1 consiste das soluções não dominadas por qualquer outra solução da população, a fronteira F_2 de ranque 2 consiste das soluções que não são dominadas pelas soluções restantes excluindo as da fronteira F_1, a fronteira F_3 de ranque 3 consiste das soluções que não são dominadas pelas soluções restantes ex-

cluindo as fronteiras F_1 e F_2 e assim sucessivamente até a fronteira F_n, ranqueando toda população de soluções.

Com esta solução elitista pode-se a partir de uma população $P(t)$ e uma população sucessora O(t) obter uma nova população $P(t+1)$ de tamanho N a qual será formada pelas soluções de melhor ranque obtidas ao aplicar o método de *fast non-dominated sorting* sobre a união das populações de $P(t)$ e $O(t)$, cuja população total é $2N$, e realizar o truncamento obtendo uma população final de tamanho N. O algoritmo do *fast-nondominated-sort* pode ser visto no Algoritmo 1.

Algoritmo 1: Fast non-dominated sorting

Data: P
Result: P ordenado pelo critério de não dominância Pareto

```
1  begin
2      for each p ∈ P do
3          for each q ∈ P do
4              if p ≺ q then
5                  S_p = S_p ⋃ {q};
6              end
7              else
8                  if q ≺ p then
9                      n_p = n_p + 1;
10                 end
11             end
12         end
13         if n_p = 0 then
14             F_1 = F_1 ⋃ {p};
15         end
16         i=1;
17     end
18     while F_i ≠ 0 do
19         H = 0;
20         for each p ∈ F_i do
21             for each q ∈ S_p do
22                 n_q = n_q − 1;
23                 if n_q = 0 then
24                     H = H ⋃ {q};
25                 end
26             end
27         end
28         i = i + 1;
29         F_i = H;
30     end
31 end
```

Com a finalidade de melhorar a distribuição das soluções na fronteira Pareto o NSGA-II utiliza um método de estimação da densidade das soluções baseados na distância média de um ponto i a outros dois pontos $i-1$ e $i+1$,para problemas com dois objetivos, que se encontram lado a lado a este ao longo de cada objetivo

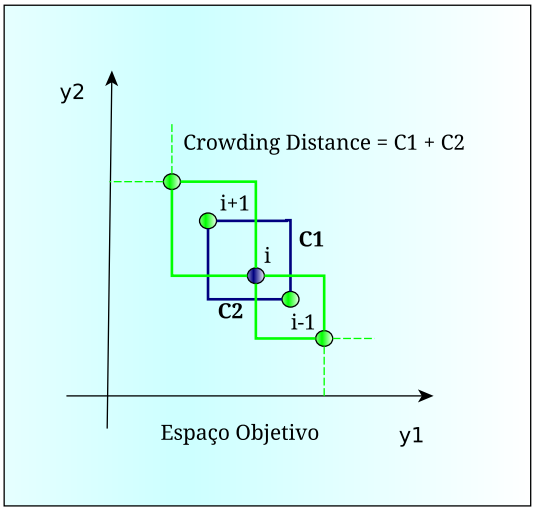


Figura 3.1: Operador de diversidade crowding distance utilizado no NSGA-II

Deb et al. (2000). A distância média a multidão é dada pelo volume do cuboide destes dois pontos mais próximos ao ponto i como ilustrado na figura 3.1 além disso para as soluções que se encontram no limite da fronteira Pareto é atribuído um valor grande de forma a preservar estas soluções.

O algoritmo de crowding distance retorna a população selecionada ordenada pela distância a multidão e pode ser visto com mais detalhes no Algoritmo 2.

3.2.2 MOABC

Como mencionado na introdução deste trabalho, o MOABC é uma versão do ABC para problemas de otimzação multiobjetivo utilizando a dominância Pareto e um arquivo externo que manter as melhores soluções encontradas pelo algoritmo. A seguir alguns detalhes sobre os métodos utilizados pelo MOABC podem ser vistos.

A) Tipos de abelhas

Tanto o algoritmo ABC quanto o MOABC apresentam os mesmos três tipos de abelhas: operárias, observadoras e escoteiras. A fonte de alimento corresponde a uma possível solução para o problema de otimização. A seguir é detalhado a função de cada uma das abelhas.

- As abelhas operárias escolhem as fontes de alimento memorizando as regiões onde existem as melhores fontes, em seguida retornam à colmeia onde compartilham as informações com as suas companheiras com certa probabilidade;
- As abelhas observadoras escolhem as fontes de alimento dependendo da experiência de suas companheiras de colmeia, as abelhas operárias, ajustando assim suas posições no espaço de busca. Quanto maior a quantidade de néctar de uma fonte de alimento apresentada às abelhas observadoras por uma abelha operária maior será a probabilidade de uma abelha observadora escolher esta fonte para explorar;

Algoritmo 2: Crowding Distance

Data: F_n
Result: F_n ordenado pela distância a multidão

```
1  begin
2      //Número de soluções do conjunto I;
3      l = length(I);
4      //Inicializa distâncias;
5      for each i ∈ I do
6          D[i]_distance = 0;
7      end
8      for each objetive m do
9          //Ordena usando cada valor objetivo;
10         I = sort(I, m);
11         //Os pontos extremos são sempre selecionados;
12         D[1]_distance = D[l]_distance = ∞;
13         //Para os outros pontos;
14         for i = 2 to l − 1 do
15             D[i]_distance = D[i]_distance + I[i + 1].m − I[i − 1].m;
16         end
17     end
18 end
```

- As abelhas escoteiras são responsáveis pela descoberta de novas fontes, diversificando as regiões de busca e evitando desta forma a convergência prematura do algoritmo. Elas realizam este procedimento de busca aleatório à medida que as fontes de alimento esgotam sem utilizar da experiência de suas companheiras.

No algoritmo do MOABC cada fonte de comida é associada a um vetor posição $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ onde i corresponde a uma solução do espaço de variáveis e m é a dimensão deste espaço. O algoritmo usa a colônia de abelhas para explorar o espaço de soluções com a finalidade de encontrar a melhor fonte de néctar,ou seja, a melhor solução.

B) Arquivo externo

Um arquivo externo formado por uma grade é utilizada para armazenar as melhores soluções utilizando ϵ-dominância. Esta grade permite armazenar as soluções x_i de forma que cada caixa de tamanho ϵ armazena uma solução na grade a qual deve ter uma dimensão igual a do espaço de objetivos. Caso duas soluções ocupem a mesma caixa a solução que tiver a menor distância Euclidiana em relação ao canto inferior e esquerdo da caixa (no caso de problemas de minimização) permanece e a outra é excluída.

C) Parâmetro de abandono

O parâmetro max_trial é utilizado para identificar quando uma fonte de comida deve ser abandonada de forma que uma abelha escoteira possa ser enviada para encontrar uma nova fonte. Assim cada fonte apresenta uma variável

$trial_i$ que indica a situação daquela fonte e quando ela deve ser abandonada, ou seja, quando $trial_i > max\_trial$.

D) Enviando abelhas operárias

Inicialmente a população da colmeia é divida em duas partes sendo a primeira a de abelhas operárias e a segunda a de abelhas observadoras. Para cada abelha operária i é vinculada uma solução candidata a qual a abelha tentará melhorar a partir da seleção de um dos parâmetros da solução da abelha vizinha selecionada aleatoriamente, depois a fonte é atualizada usando a equação 3.1

$$v_{id} = v_{id} + \phi_{id}(x_{id} - x_{kd}) \tag{3.1}$$

onde i representa as soluções candidatas as quais serão otimizadas, $k \in \{1, 2, ..., n_{fontes}\}$ e $d \in \{1, 2, ..., m\}$ são índices aleatoriamente escolhidos com $k \neq i$. O número ϕ_{id} é um número aleatório com distruição uniforme na faixa de $[-1, 1]$ que permite controlar produção de novas soluções vizinhas da fonte em torno da solução x_i.

E) Enviando abelhas observadoras

Ao retornarem para a colmeia, as abelhas operárias compartilham as soluções encontradas com as abelhas observadoras de acordo com a probabilidade dada pela equação 3.2

$$p_k = \frac{fit(\vec{x}_k)}{\sum_{s=1}^{total\_fontes} fit(\vec{x}_s)} \tag{3.2}$$

onde $p_k(\vec{x}_k)$ é a probabilidade da solução proposta pela abelha operária k de ser escolhida pela abelha observadora. A qualidade da solução depende do número de soluções dominadas pela solução k em relação ao total de soluções conforme a equação 3.3

$$fit(\vec{x}_k) = \frac{dom(s)}{total\_fontes} \tag{3.3}$$

onde $dom(s)$ é o número de soluções dominadas pela solução s. Após calcular a probabilidade, cada abelha observadora utiliza uma roleta para escolher a solução anunciada pela abelha k, baseado nesta probabilidade ela utiliza a equação 3.1 para encontrar uma nova solução que domine a atual.

F) Enviando abelhas escoteiras

A cada ciclo do algoritmo as soluções candidatas são avaliadas e se a solução não for melhorada em número de interações dado pelo parâmetro max_trial ela é abandonada e a abelha escoteira parte para encontrar uma solução substituindo a atual a partir da equação 3.4

$$x_i^j = x_{min}^j + rand[0, 1](x_{max}^j - x_{min}^j) \tag{3.4}$$

onde $j \in \{1, 2, ..., D\}$ e $rand[0, 1]$ é um valor aleatório com distribuição uniforme.

3.2.3 MOEA/D

O *Multiobjective evolutionary algorithm based on decomposition* (MOEA/D) é um novo algoritmo proposto por Zhang e Li (2007) baseado na estratégia de decomposição, a qual consiste em decompor um problema de otimização multiobjetivo dentro de um conjunto de subproblemas de otimização escalares e otimizá-los simultaneamente. As abordagens de decomposição, ou funções de agregação, mais utilizadas por este algoritmo incluem a Abordagem da Somas Ponderadas e a Abordagem Tchebycheff (Steuer e Choo, 1983), porém outras abordagens podem ser utilizadas.

A) Somas Ponderadas

Esta abordagem considera uma combinação de diferentes objetivos ponderados conforme o vetor peso $\lambda = (\lambda_1, ..., \lambda_n)^T$, isto é, para $\lambda_i \geq 0$ e para todo $i = 1, ..., n$ tem-se $\sum_{i=1}^{n} \lambda_i = 1$.

Assim cada solução ótima i para o problema de otimização escalar consiste em:

$$minimizar \quad g^{sp}(x|\lambda) = \sum_{i=1}^{n} \lambda_i f_i(x) \tag{3.5}$$

onde $x \in \mathcal{S}$ são as variáveis a serem otimizadas e $g^{sp}(x|\lambda)$ corresponde a função objetivo a ser otimizada a partir de diferentes vetores pesos λ.

B) Abordagem de Tchebycheff

Nesta abordagem o problema escalar consiste em

$$minimizar \quad g^{te}(x|\lambda, z^*) = \underset{1 \leq i \leq n}{max}\{\lambda_i |f_i(x) - z_i^*|\} \tag{3.6}$$

tal que $x \in \mathcal{S}$ e $z^* = (z_1^*, ..., z_n^*)^T$ é o ponto de referência, isto é, $z_i^* = min\{f_i(x)|x \in \mathcal{S}\}$ para cada $i = 1, ..., n$. Onde para cada solução x^* existe um vetor peso λ tal que x^* é a solução ótima de 3.6 e cada solução de 3.6 é pareto ótima de 2.1. Ou seja para obter a solução ótima $f(x^*)$ minimiza-se $g^{te}(x|\lambda, z^*)$ considerando o máximo das normas infinitas dos objetivos do problema referentes a um ponto de referência, ponderando-as com um determinado peso de forma a obter a solução mais próxima a z^*.

Os relacionamentos entres os problemas vizinhos são baseados nas distâncias entre seus vetores pesos, onde um subproblema i é um vizinho de um subproblema j se o vetor peso do subproblema i está próximo de j. Cada subproblema é otimizado no MOEA/D utilizando informações principalmente de seus vizinhos por meio de operadores genéticos, mantendo as melhores soluções encontradas até o momento em uma memória vinculada ao subproblema (Zhou et al., 2011).

3.2.4 GDE3

Evolução diferencial (*Differencial Evolution* - DE) é uma estratégia de busca baseada em população muito similar aos algoritmos evolucionários. A principal diferença está na maneira como ocorre a reprodução, onde um sucessor passa a ser criado a partir de outras três soluções usando um operador aritmético de cruzamento (Engelbrecht, 2006).

Conforme Kukkonen e Lampinen (2005) como um típico EA, o DE gera uma população inicial aleatória de tamanho NP, a qual é melhorada usando seleção, operadores de mutação (*crossover control parameter* - CR) e de cruzamento (mutation factor - F). Vários métodos podem ser utilizados aqui para determinar o critério de parada mas normalmente adota-se o número de gerações ou o número de funções de avaliações. Em cada geração, o DE vai em cada vetor de decisão $\vec{x}_{i,G}$ da população e cria um correspondente vetor *trial* $\vec{u}_{i,G}$. Aqui, i corresponde ao índice do vetor na população e G o índice da geração. O GDE3 corresponde a uma terceira versão do *Generalized Differential Evolution* (GDE) em que além de extender o DE para problemas de otimização multiobjetivo restritos e utilizar métodos de seleção baseados na distância a multidão (crowding distance - CD) como seus antecessores este adiciona uma nova estratégia. Nesta estratégia caso ocorra a seleção de duas soluções factíveis e não dominadas, ambas são salvas e utilizadas na próxima geração, porém antes de continuar a população é ordenada com *non-dominated sorting* e truncada para preservação da diversidade usando uma abordagem similar ao NSGA-II.

3.3 Algoritmos de estimação de distribuição

Algoritmos de Estimação de Distribuição (*Estimation of Distribution Algorithm* - EDA) vêm sendo amplamente estudados devido a sua capacidade de lidar com problemas com muitas variáveis por meio da construção de modelos de probabilidade amostrados das populações mais promissoras.

Conforme Pelikan et al. (2012) enquanto muitas meta-heurísticas essencialmente amostram de uma distribuição de probabilidade implícita por usar uma combinação de operadores de busca estocásticos, a percepção de que dentro do problema existe uma representação de explícitos modelos probabilísticos das populações candidatas dão aos EDAs uma clara vantagem sobre as outras meta-heurísticas.

Para Armañanzas et al. (2008) outra vantagem dos EDAs em relação as técnicas evolucionárias que se baseiam em algoritmos genéticos são a ausência de múltiplos parâmetros para serem ajustados, o que permite uma maior expressividade e transparência dos modelos que guiam o processo de busca.

Pode-se distinguir os algoritmos de estimação de distribuição basicamente em duas abordagens (Luo e Qian, 2009); a primeira abordagem é denominada estimação de densidade paramétrica, onde se assume que a forma da distribuição é conhecida, e o problema consiste em descobrir os parâmetros que modelam a distribuição. Na segunda abordagem, denominada estimação de densidade não paramétrica, a suposição acerca da estrutura da função distribuição não existe, o que eleva em muito o custo computacional.

Os EDAs também podem ser divididos conforme a complexidade dos modelos

probabilísticos usados para capturar a interação entre as variáveis do problema: abordagens univariadas, bivariadas ou multivariadas (Armañanzas et al., 2008). Algoritmos univariados e contínuos como UMDA$_c^G$ (Larrañaga et al., 2000), PBILc (Sebag e Ducoulombier, 1998) e EGNA$_{ee}$ (Larrañaga et al., 2000) baseiam-se em redes Gaussianas, porém por não considerarem as interelações das variáveis dos problema, estes apresentam certas limitações para estimar o espaço de busca das novas soluções. Sendo assim algoritmos baseados em relações de duas variáveis foram propostos, como MIMICC (Larrañaga et al., 2000), COMMIT (Baluja e Davies, 1997) e BMDA (Pelikan e Mühlenbein, 1999). Porém como muitos problemas apresentam mais de duas variáveis correlacionadas, algoritmos como ECGA (Harik e Harik, 1999), rBOA (Ahn e Ramakrishna, 2007) e FDA (Mühlenbein e Mahnig, 1999) foram propostos com finalidade de capturar maiores informações destas relações.

Neste trabalho considera-se apenas as distribuições contínuas Gaussianas que são amplamente utilizadas em outros algoritmos, inclusive no UMDA$_c^G$ utilizado neste trabalho e cuja função aqui será manter a diversidade das soluções a fim de evitar também convergências prematuras devida a utilização do algoritmo *Real-Coded Estimation of Distribution Algorithm* (RECEDA) que avalia as correlações de todas as variáveis.

3.3.1 Algoritmo UMDA$_c^G$

O algoritmo UMDA$_c$ proposto por Larrañaga et al. (2000) pertence a uma categoria dos EDAs que não leva em conta as dependências entre as variáveis do problema. Este algoritmo é uma evolução do UMDA discreto para o espaço contínuo onde métodos de aprendizagem de estruturas de dados e simulação de redes Gaussianas são aplicados, assim como métodos baseados na teoria da informação. Conforme Dong e Yao (2007) nesta abordagem, para todas gerações e todas variáveis algumas estatísticas de teste são executadas com a finalidade de obter a função de densidade que melhor se ajusta a variável. Em UMDA$_c$ a fatorização da função de densidade conjunta é dada por 3.7.

$$f_l(x, \theta^l) = \prod_{i=1}^{n} fl(x_i, \theta_i^l) \tag{3.7}$$

Para esse trabalho utiliza-se um caso especial do UMDA$_c$ denominado *Univariate Marginal Distribution Algorithm for Gaussian models* ou UMDA$_c^G$, onde cada uma das variáveis do problema são consideradas normais e são estimadas conforme as estimativas de máximo verossimilhança dadas pelas equações 3.8 e 3.9.

$$\hat{\mu_i}^l = \overline{x}_i^l = \frac{1}{N} \sum_{r=1}^{N} x_{i,r}^l \tag{3.8}$$

$$\hat{\sigma_i}^l = \sqrt{\frac{1}{N} \sum_{r=1}^{N} (x_{i,r}^l - \overline{x}_i^l)^2} \tag{3.9}$$

onde $\hat{\mu_i}^l$ e $\hat{\sigma_i}^l$ são a média e o desvio padrão, respectivamente, da l-ésima geração e $(x_{1,r}^l, x_{2,r}^l, ..., x_{N,r}^l)$ são os valores das i-ésima variáveis das N soluções selecionadas.

3.3.2 RECEDA

Baseando somente na média e na matriz de covariância de uma população promissora Paul e Iba (2003) propõem o algoritmo *Real-Coded Estimation of Distribution Algorithm* ou RECEDA para estimação de novos indivíduos da população a partir de uma distribuição normal multivariada. Neste algoritmo todas as variáveis são consideradas normais e a sua matriz de covariância é decomposta usando a decomposição de Cholesky. Considerando a matriz de covariância Σ da população amostrada sendo simétrica e positiva definida, pode-se decompor-lá em uma única matriz triangular inferior L com $LL^T = \Sigma$. Tomando $Z1, Z2, ..., Zn$ um vetor com n variáveis independentes com desvio normal, pode-se gerar $X \sim N(\mu, \Sigma)$ usando a equação 3.10.

$$X = \mu + LZ \tag{3.10}$$

Porém nem sempre a matriz de covariância obtida da população é positiva definida, neste caso utiliza-se um método de correção da matriz de covariância chamado Covariance Matrix Repairing (CMR) e que será descrito na próxima seção. No algoritmo 3 o RECEDA modificado com o método CMR pode ser visto.

Algoritmo 3: RECEDA

```
   Data: Population
   Result: L
1  begin
2      //Média de cada variável dos indivíduos selecionados;
3      μ ← means(P_n);
4      //Covariância de cada variável dos indivíduos selecionados;
5      Σ ← means(P_n);
6      //Repara matriz de covariância;
7      CMR(Σ);
8      //Decompõem a matriz de covariância usando a decomposição de
       Cholesky;
9      LL^T ← Σ;
10     //Retorna a matriz inferior após a decomposição;
11     return L;
12 end
```

3.3.3 Covariance Matrix Repairing

Os EDAs empregam em sua grande maioria modelos gaussianos para estimar as novas soluções, porém devido ao emprego da matriz covariância completa tais algoritmos falham em específicas condições causadas por inevitáveis erros de computação, sendo assim Dong e Yao (2007) propõem um método para a correção de matrizes de covariância mal formadas como forma de aumentar a robustez dos EDAs sobre qualquer escala de tamanho da população. Estes erros se intensificam à medida que utiliza-se amostras relativamente pequenas da população selecionada e aumenta-se o número de variáveis do problema.

O algoritmo Covariance Matrix Repairing (CMR) consiste em calcular os autovalores da matriz de covariância e atribuir o menor valor obtido a λ. Com isso pode-se verificar se a matriz é positiva definida ao obter valores positivos de λ, porém caso o valor de λ seja menor que 0 a matriz não é positiva definida e o algoritmo CMR tenta corrigir a matriz incrementando gradativamente o valor da diagonal principal utilizando o módulo de λ multiplicado por um parâmetro K até obter uma matriz de covariância positiva definida com o mínimo de modificações. O método CMR pode ser visto no algoritmo 4.

Algoritmo 4: CMR

```
Data: Σ
Result: Σ
begin
    K ← 1.5;
    repeat
        λ ← Calcula o menor auto valor de Σ;
        if λ ≥ 0 then
            Σ é positivo semi-definido;
            Sai do loop;
        end
        else
            Σ ← Σ + λ.K.I, onde I é uma matriz identidade
        end
    until;
    K ← K.Δ
end
```

3.4 Operador de Mutação de Cauchy

A distribuição de Cauchy diferentemente da distribuição Gausiana apresenta variância infinita o que a torna muito útil para sair de mínimos locais e para melhorar a diversidade. Devido a propensão dos EDAs e de outros algoritmos a convergir para mínimos locais, alguns trabalhos vêm utilizando a distribuição de Cauchy como operador de mutação obtendo bons resultados (Luo e Qian, 2010) (Wang et al., 2007). Para uma dimensão a função densidade centrada na origem é dada por:

$$f_c(x) = \frac{1}{\pi}\frac{t}{t^2 + x^2}, -\infty < x < +\infty \tag{3.11}$$

onde $t > 0$ é um parâmetro escalar e a função de distribuição correspondente é definida por:

$$F_c(x) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x}{t}) \tag{3.12}$$

3.5 Clusterização

A clusterização, ou agrupamento, é uma das mais primitivas atividades mentais do ser humano, usadas para manipular uma grande quantidade de informação que se recebe todos dias. Processar todas essas informações de forma única poderia ser impossível. Assim o ser humando tende a categorizar entidades (i.e., objetos, pessoas, eventos) dentro de clusters, sendo cada *cluster* caracterizado pelos atributos comuns das entidades que ele contém (Theodoridis e Koutroumbas, 2008).

O principal objetivo dos algoritmos de clusterização é o de formar grupos de pontos similares. A clusterização faz uso das indicadores de distância, tal como distância Euclidiana, para definir a similaridade entre dois pontos dados. Baseados nesta medida de similaridade, a clusterização pode ser formulada como um problema de otimização onde o objetivo é simultaneamente maximizar as distâncias entre *clusters* e ao mesmo tempo minimizar as distâncias dentro do *cluster* (Engelbrecht, 2006).

3.5.1 K-Means

O *K-means* é um dos mais populares algoritmos de clusterização proposto por MacQueen (1967). Este algoritmo de clusterização de ordem $O(n)$ permite particionar populações com D dimensões dentro de k conjuntos.

A resolução deste problema de otimização pelo *K-means* consiste em minimizar a função objetivo 3.13 , onde dado um conjunto de n pontos no espaço real de dimensão D, $\mathbb{R}^d$, e um número de clusters k, deseja-se determinar um conjunto de k pontos em $\mathbb{R}^d$, chamados de centros, tal que minimize a distância média quadrática de cada ponto x_j de seu mais próximo centro C_i, onde μ_i aqui é o centroide do cluster C_i (Kanungo et al., 2002).

$$\underset{\mu_1,\ldots,\mu_k}{min} \sum_{i=1}^{k} \sum_{x_j \in C_i} d(x_j - \mu_i) \tag{3.13}$$

No *K-means* os conjuntos de clusters são formados da seguinte forma: inicialmente são gerados k centros aleatoriamente ou por meio de algum critério, em seguida para cada ponto da população calcula-se a distância até um dos k centros e associa-se este ponto ao centro mais próximo, depois de associar todos pontos é recalculado os centros de cada cluster a partir da média dos pontos dos clusters formados e tenta-se associar novamente os pontos a cada centro de forma que ao final de n iterações não se tenha mais nenhuma mudança. A figura 3.2 mostra um exemplo do *K-means* sendo utilizado na formação de 5 clusters no espaço objetivo, cada $*$ corresponde ao centroide do cluster e cada uma das cores representa o cluster a qual pertence cada um dos pontos.

3.6 Framework MOEA

A utilização de *frameworks* ou arcabouços vem sendo amplamente difundida no desenvolvimento de softwares, permitindo aumentar sua produtividade e qualidade. Conforme Fayad e Schmidt (1997) os principais benefícios de um *framework* são

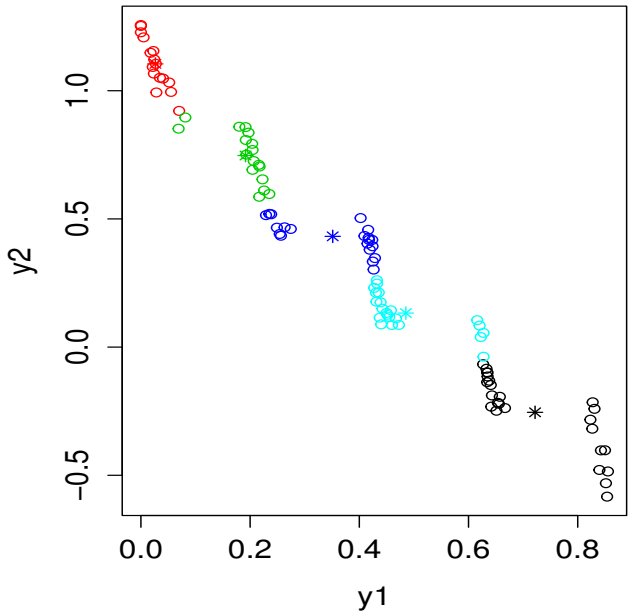


Figura 3.2: Clusters gerados pelo *K-means* com k=5

a modularidade, reusabilidade, extensibilidade, e a inversão de controle, onde a sequência de controle não é mais definida pelo programador e sim pelo *framework*. Um *framework* pode ser classificado pelas técnicas que permitem sua utilização pelo programador, variando do modelo caixa branca onde o programador tem acesso a estrutura interna do *framework*, necessitando porém de maior conhecimento sobre esta, ao modelo caixa preta onde o programador tem acesso a apenas uma composição baseada em objetos por meio de interfaces bem definidas não sendo possível visualizar a estrutura interna.

Muitos *frameworks* foram propostos para otimização multiobjetivo com meta-heurísticas, incluindo o jMetal (Durillo et al., 2006), PISA (Bleuler et al., 2003), Shark (Igel et al., 2008), ParadisEO (Cahon et al., 2004), MOMHLib++ (Jaszkiewicz, 2005), MOEA (Hadka, 2011) entre outros. Um estudo comparativo sobre frameworks para otimização com meta-heurísticas pode ser visto com mais detalhes em (Parejo et al., 2012). Neste trabalho optou-se pelo framework MOEA devido sua ampla utilização em estudos comparativos de algoritmos multiobjetivos evolucionários (*Multiobjective Evolutionary Algorithms*-MOEAs).

O MOEA é um *framework open source* desenvolvido na linguagem Java modelo caixa branca para computação evolucionária com enfoque no desenvolvimento e experimentação com MOEAs para otimização multiobjetivo e outros algoritmos de otimização de propósito geral. O MOEA provê suporte a vários algoritmos, muitos destes já implementados no *framework* como o NSGA-II, ϵ-MOEA, GDE3 e MOEA/D, além de integrar outros algoritmos de frameworks como o jMetal e PISA. Algumas das principais características deste *framework* são listadas abaixo:

- Velocidade, segurança na implementação de muitos MOEAs do estado da arte;

- Possibilidade de estender algoritmos, problemas e operadores;
- Desenvolvimento modular para a construção de novos algoritmos a partir dos componentes existentes;
- Documentação completa do código fonte;
- Classes para análise e comparação dos resultados;
- Mais 1100 casos testes para assegurar a validade.

Entre outras características do MOEA tem-se a possibilidade de estender problemas, realizar análises baseadas em indicadores, e estender algoritmos baseado nas classes *AbstractEvolutionaryAlgorithm* ou *AbstractAlgorithm*. Na figura 3.3 é mostrado um diagrama simplificado em UML o qual foi construído a partir da especificação da API deste framework. Neste diagrama pode-se visualizar o relacionamento de cada classe como herança e composição. Além das classes mostradas neste diagrama o MOEA provê outras envolvidas na execução, instrumentação e análise as quais são detalhadas a seguir baseadas no manual do MOEA (Hadka, 2011).

3.6.1 Executor

O executor consiste de uma classe responsável pela construção e execução de um algoritmo. Para sua execução são necessárias 3 informações.

- o problema teste;
- o algoritmo usado para resolver o problema;
- o número de avaliações da função objetivo para resolver o problema.

3.6.2 Instrumenter

Além de suportar vários algoritmos e problemas testes, o MOEA também contém um conjunto de ferramentas para analisar o desempenho dos algoritmos. Ao utilizar esta classe tem-se a possibilidade de executar dois tipos de análises mostrados a seguir.

- *Run-time dynamics*: captura o comportamento do algoritmo em tempo de execução, registrando como a qualidade das suas soluções e outros elementos mudam;
- *End-of-run*: foca no resultado após a execução completa do algoritmo e compara o desempenho relativo a outros algoritmos.

Neste trabalho utiliza-se a análise *End-of-run* visto que se deseja comparar vários algoritmos na resolução de problemas testes.

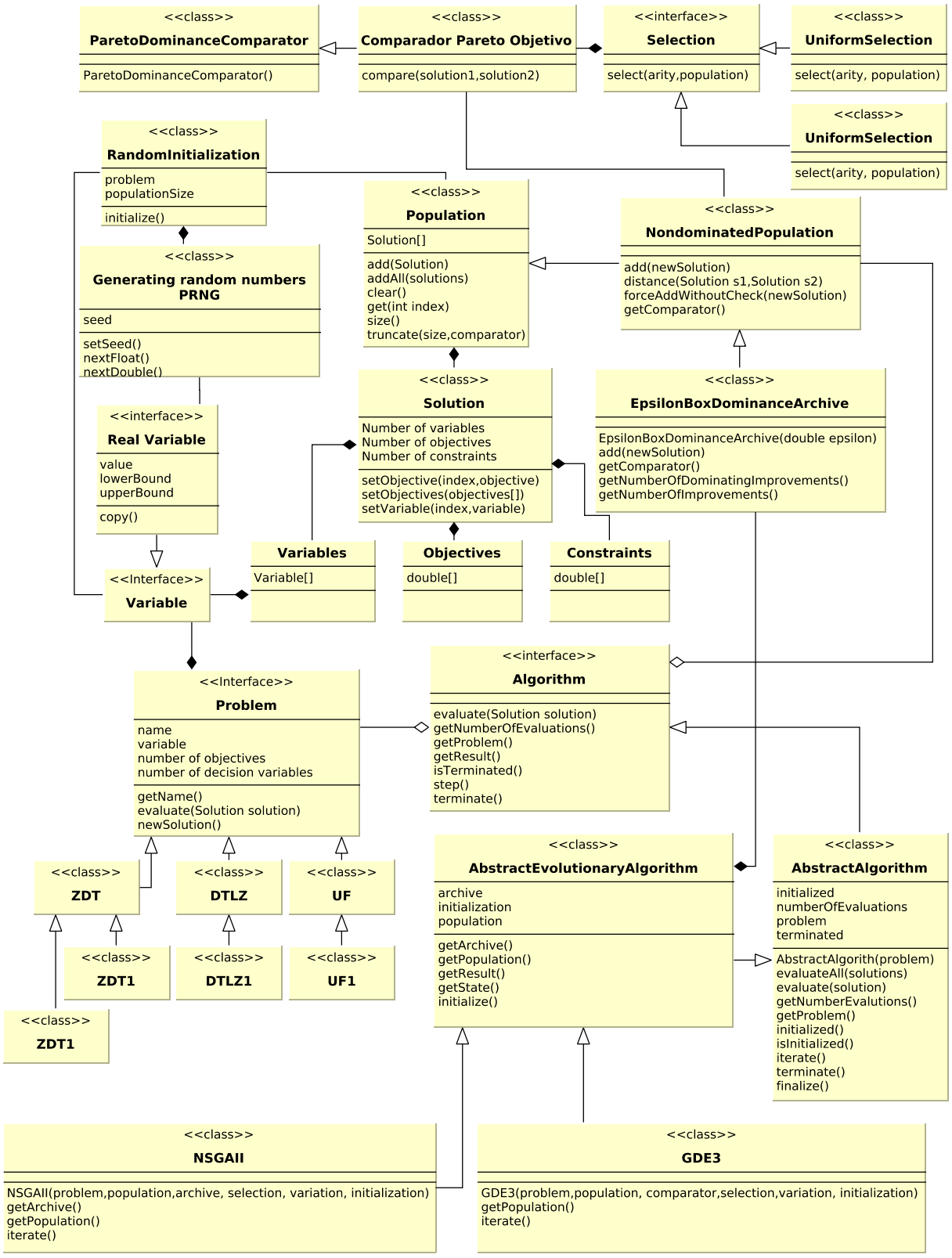


Figura 3.3: Diagrama simplificado do MOEA

3.6.3 Analyzer

A classe *Analyzer* provê uma análise após a execução do algoritmo (*End-of-run*). Esta análise foca no resultado do conjunto Pareto ótimo aproximado comparando este a um conjunto de referência. O *Analyzer* é particularmente útil na comparação estatística dos resultados produzidos por dois ou mais algoritmos, ou pelo menos um com diferentes parâmetros de configuração. Nestas análises o *Analyzer* produz várias estatísticas referentes a cada indicador utilizado tal como a média, o máximo e mínimo após n execuções dos algoritmos. Além disso estatísticas de significância como os testes de Mann-Whitney e Kruskal-Wallis também são aplicados.

3.7 Trabalhos Relacionados

Nesta seção destaca-se alguns algoritmos baseados em EDAs e em *clusters* que buscam extrair maiores informações das populações ou subpopulações utilizando modelos probabilísticos. Também desta-se alguns algoritmos baseados em inteligência de enxames e no operador de mutação de Cauchy.

Pelikan et al. (2005) propõem um algoritmo para resolução de problemas multiobjetivos decompostos combinando *hierarchical Bayesian optimization algorithm* (hBOA) com o NSGA-II e *clusters* de igual tamanho no espaço objetivo como forma de melhorar a escalabilidade.

Waldock e Corne (2010) descrevem um algoritmo de Otimização Multiobjetivo(*Multiobjective Optimization* - MOO) utilizando o *framework* de otimização *Probability Collectives* (PC). O PC é uma abordagem de otimização onde o foco consiste em encontrar um modelo de distribuição ideal de uma solução no espaço ao invés de uma solução ideal. O *Multiobjective Probability Collectives* (MOPC) é um algoritmo que utiliza uma estratégia de ranqueamento baseado em Pareto dominância mantendo as melhores soluções em um arquivo baseado na distância a multidão (*Crowding Archive*).

Wang et al. (2007) propõem o algoritmo híbrido *hybrid PSO* (HPSO) para solução de problemas de otimização adicionando o operador de mutação de Cauchy nas melhores partículas encontradas até o momento, possibilitando as outras partículas alcançar melhores posições na resolução de problemas com funções multimodais.

Vo et al. (2009) discutem o forte relacionamento entre (*Co-operative Coevolutionary Algorithms* - CCEAs) e EDAs, especialmente modelos baseados em EDAs univariados como o UMDA, CGA (Harik e Harik, 1999) e PBIL (Sebag e Ducoulombier, 1998) e a possibilidade de transferir teorias e algoritmos de CCEA para EDAs e vice versa.

Costa e Minisci (2003) propõem o algoritmo *Multi-objective Parzen-Based Estimation of Distribution Algorithm for Continuous Problems* (MOPED) baseado em EDAs usando o estimador de Parzen para aproximar a densidade de probabilidade das soluções próximas a fronteira Pareto, com multivariadas dependências entre as variáveis. O algoritmo também emprega os métodos *Nondominated Sorting Genetic* e *Crowding Distance* para classificar as soluções promissoras.

Levando em conta que não existe evidências de que um Algoritmo Evolutivo (EA) seja superior a outro em todos os problemas, Shim et al. (2012) propõem dois algoritmos híbridos. O primeiro consiste de um híbrido baseado em GA, DE e EDA

sintetizado de tal maneira que a taxa de soluções produzidas por cada algoritmo é controlada conforme o número de soluções promissoras produzidas por cada um em estágios anteriores. O algoritmo adaptativo é também hibridizado com uma busca local baseada na busca evolutiva por gradiente(*evolutionary gradient search* - EGS). O segundo é baseado nos conceitos de dominância e decomposição.

Zhou et al. (2008) combinam EDA com DE em um algoritmo híbrido denominado EDA+DE, o qual explora uma propriedade encontrada em alguns problemas, onde a fronteira Pareto com m objetivos consiste de $(m-1)$ conjuntos de variáveis contínuas duplicadas no espaço objetivo (espaço de decisão) sujeito a pequenas condições, explorando desta forma a localização da informação guiando o processo de seleção e o processo de construção do modelo no espaço de decisão.

Como muitos algoritmos evolucionários os EDAs podem convergir para ótimos locais, visto isto Luo e Qian (2010) desenvolvem um novo algoritmo combinando as funções de densidade Gaussiana e a função de densidade Cauchy cujo o objetivo é manter a diversidade das novas soluções.

3.8 Considerações Finais

Neste capítulo vários conceitos necessários para o entendimento deste trabalho foram apresentados, como os conceitos de EDAs, de clusterização, dos algoritmos CMR, GDE3, MOEA/D, MOABC, NSGA-II, UMDA$_c^G$ e RECEDA, assim como do *framework* MOEA utilizado no desenvolvimento e nos testes dos algoritmos. Destacou-se também neste capítulo alguns trabalhos relacionados do estado da arte como alguns algoritmos basedos em EDAs, em clusters e no operador de mutação de Cauchy.

Capítulo 4

Metodologia

Neste capítulo são apresentados os métodos utilizados para o desenvolvimento do algoritmo proposto.

4.1 Sociedade das abelhas

Uma colônia de abelhas pode ser vista como um grande organismo com diferentes agentes e com funções específicas buscando de forma cooperada o melhor para o grupo. Este grande organismo poderia não funcionar da maneira desejada caso não houvesse uma diferenciação das funções de cada indivíduo na colônia, pois dentro desta existem diferentes necessidades a serem realizadas como exploração de novas fontes, armazenamento de alimentos, limpeza, comunicação entre os indivíduos de novas fontes de comida, preparação da geleia real, entre outras inerentes a manutenção da colônia. Inspirados nesta organização e diferenciação de funções propõem-se neste trabalho um novo algoritmo híbrido denominado *Multiobjective Optimization Estimation of Distribution Algorithm based on Bee Colonies and Clusters* (MOEDABC) onde baseados em EDAs e Clusters tenta-se simular tal organização como forma de resolver problemas complexos com um número relativamente grande de variáveis.

A divisão de cada grupo de abelhas é mostrado a seguir:

- O primeiro grupo consiste de abelhas campistas que se baseiam no algoritmo UMDA$_c^G$(*Univariate Marginal Distribution Algorithm for Gaussian models*) para estimar as novas soluções por meio de um modelo probabilístico de toda população. Estas abelhas tem por objetivo uma maior exploração do espaço de decisão e a manutenção da diversidade das soluções;

- O segundo grupo consiste de abelhas observadoras que também se baseiam no algoritmo UMDA$_c^G$ mas levam em conta modelos baseados nos clusters formados no espaço objetivo. Neste caso as novas soluções são estimadas utilizando as informações extraídas de cada cluster;

- O terceiro grupo é representado pelas abelhas nutrizes, responsáveis pelo refinamento das soluções. Estas abelhas utilizam um conhecimento mais refinado

para explorar as novas soluções a partir da análise das correlações das variáveis da população de um determinado cluster utilizando para isto o algoritmo RECEDA;

- Finalmente o quarto grupo de abelhas denominadas escoteiras as quais utilizam a distribuição de Cauchy como operador de mutação com a finalidade de descobrir as novas fontes de comida e aumentar a diversidade.

4.2 Inicializando a população

Como muitos algoritmos evolutivos e outros EDAs, inicialmente o MOEDABC gera uma população P_0 com soluções aleatórias utilizando distribuições uniformes, e em seguida as soluções da população são avaliadas. O número de soluções da população é fixo e representa as melhores fontes de comida encontradas até o momento pelas abelhas da colônia. Cada solução será representada aqui como um vetor de decisão $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,m})$, onde $i = (1, 2, ..., T)$ é a i-ésima solução, no espaço de decisão, de uma população de tamanho T, sendo os valores de $x_{i,d}$ inicializados da seguinte forma:

$$x_{i,d} = x_d^{min} + rand[0,1](x_d^{max} - x_d^{min}) \tag{4.1}$$

onde $d \in \{1, 2, ..., m\}$, $rand[0, 1]$ corresponde a um número real de uma distribuição aleatória uniforme, x_d^{min} e x_d^{max} são os limites inferior e superior respectivamente da variável $x_{i,d}$. Após gerar as novas soluções avalia-se cada uma delas conforme o problema utilizado e o número de objetivos. O resultado da avaliação da função objetivo é atribuído ao vetor y_i, tal que:

$$y_i = f_i(x_i) = (f_1(x_i), f_2(x_i), ..., f_n(x_i)) \tag{4.2}$$

onde i é a i-ésima solução, no espaço de busca, da população e n o número de objetivos.

Cabe notar que cada abelha da colônia será responsável por gerar uma nova solução e que a cada geração ou iteração do algoritmo uma nova população de soluções B_t será produzida. A seguir esta população será combinada com a população anterior P_t, ordenada e truncada com o objetivo de manter o mesmo número inicial de soluções conforme pode ser visto a seguir.

4.3 Dividindo o espaço objetivo em clusters

Anteriormente definiu-se x_i e y_i, porém adota-se agora uma nomenclatura adicional a qual permitirá identificar a qual *cluster* cada solução pertence, tanto no espaço objetivo quanto no espaço de decisão. Baseando-se em um conjunto de *clusters* $c = 1, 2, ..., k$ onde k é o número de *clusters*, diz-se então que o vetor decisão x_i^c e o vetor objetivo y_i^c pertence ao cluster c o qual é obtido após a clusterização do espaço objetivo por meio do algoritmo $K\text{-}means(P_t, k)$ o qual recebe como entrada a

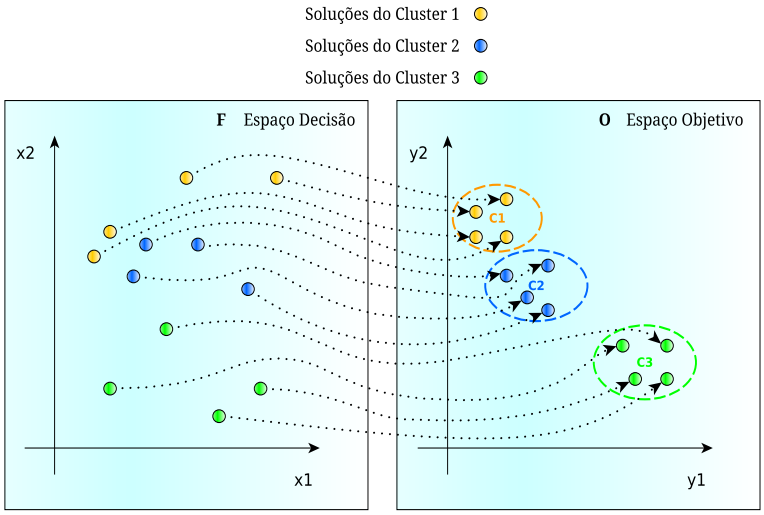


Figura 4.1: Variáveis do espaço de decisão associados a 3 clusters do espaço objetivo

população P_t do espaço objetivo e o número de clusters k. Na figura 4.1 é ilustrado a formação de 3 *clusters* e a associação destes no espaço de decisão.

O propósito aqui de dividir o espaço objetivo em *clusters* é o de extrair maiores informações das subpopulações, permitindo diminuir o espaço de busca e aumentar a convergência do algoritmo a partir da maior exploração dos *clusters* formados. Refere-se como informações a média, o desvio padrão e a matriz de covariância das subpopulações dos clusters. Estas subpopulações definidas como P_c são utilizadas pelas abelhas observadoras e nutrizes para produzir as novas soluções baseando nos algoritmos de estimação de distribuição UMDA$_c^G$ e no RECEDA respectivamente.

4.4 Abelhas Campistas

As abelhas campistas representam a maior parte das abelhas da colônia e têm como função manter a diversidade das soluções e a exploração das áreas promissoras. Para isto elas se baseiam em algumas informações da população corrente P_t como a média e o desvio padrão de cada variável. Com estas informações as abelhas campistas podem estimar as novas soluções utilizando o algoritmo UMDA$_c^G$, que por considerar a população inteira permite obter uma maior diversidade das soluções evitando convergências prematuras. As soluções estimadas com o algoritmo UMDA$_c^G$ se baseiam na equação 4.3 onde x_i, N, $\mu = (\mu_1, \mu_2, ..., \mu_m)$ e $\sigma = (\sigma_1, \sigma_2, ..., \sigma_m)$ são vetores de dimensão m igual ao número de variáveis do problema e correspondem respectivamente a nova solução estimada, a distribuição Gaussiana multivariada, a média e ao desvio padrão.

$$x_{i,d} = N(\mu_d, \sigma_d) \tag{4.3}$$

Os valores de μ_d e σ_d, referentes a dimensão d do espaço de decisão, são calculados a partir das equações 4.4 e 4.5.

$$\mu_d \cong \hat{\mu}_d = \frac{1}{T}\sum_{i=1}^{T} x_{i,d} \tag{4.4}$$

$$\sigma_d \cong \hat{\sigma}_d = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(x_{i,d} - \overline{x}_{i,d})^2} \tag{4.5}$$

onde os parâmetros μ_d e σ_d da população corrente P_t de tamanho T são considerados iguais aos parâmetros de máximo verossimilhança $\hat{\mu}_{i,d}$ e $\hat{\sigma}_{i,d}$. Considerando as restrições de cada problema a variável $x_{i,d}$ será dada então por:

$$x_{i,d} = Min(Max(N(\mu_d, \sigma_d), x_d^{min}), x_d^{max}) \tag{4.6}$$

Dizer que a distribuição das variáveis $x_{i,d}$ da população P_t são originadas de uma função de densidade (pdf) Gaussiana com os parâmetros $\mu_{i,d}$ e $\sigma_{i,d}$ consiste de uma afirmação muito forte já que não sabe se a distribuição de tais variáveis são realmente normais. Além disso a população representada aqui é relativamente pequena o que pode levar a resultados incorretos ao estimar os valores dos parâmetros. No entanto esta abordagem permite restringir nosso espaço de decisão e guiar o processo de busca baseando no conhecimento obtido das soluções produzidas pelos outros agentes, abelhas, concentrando a busca nas regiões de maior probabilidade.

4.5 Abelhas Observadoras

Ao empregar o algoritmo UMDA$_c^G$ baseado nos parâmetros μ e σ da população P_t tem-se grandes chances de convergir para mínimos locais ao deixar regiões do espaço de decisão com menor concentração de soluções sem a devida atenção. Para contornar este viés dividiu-se o espaço objetivo em *clusters*, como mencionado anteriormente, estimando logo a seguir os parâmetros μ_d^c e σ_d^c os quais serão utilizados para gerar as novas soluções. Cada um destes parâmetros representam a média e o desvio padrão de cada *cluster* c os quais irão guiar cada abelha observadora no processo de busca. Os valores de μ_d^c e σ_d^c são calculados a partir das equações 4.7 e 4.8.

$$\mu_d^c \cong \hat{\mu}_d^c = \frac{1}{T^c}\sum_{i=1}^{T^c} x_{i,d}^c \tag{4.7}$$

$$\sigma_d^c \cong \hat{\sigma}_d^c = \sqrt{\frac{1}{T^c}\sum_{i=1}^{T^c}(x_{i,d}^c - \overline{x}_{i,d}^c)^2} \tag{4.8}$$

Deste modo pode-se calcular os valores de cada variável do vetor x_i considerando as informações obtidas dos *clusters* conforme a equação 4.9

$$x_{i,d} = Min(Max(N(\mu_d^c, \sigma_d^c), x_d^{min}), x_d^{max}) \tag{4.9}$$

4.6 Abelhas Nutrizes

Para aumentar a capacidade do algoritmo de gerar novas soluções e a sua convergência para fronteira Pareto introduz-se também um novo tipo de abelha denominada de nutriz cuja função aqui será refinar as soluções a partir do algoritmo RECEDA que se baseia na média μ^c e na matriz de covariância Σ^c de cada cluster para gerar as novas soluções. Estas abelhas agem gerando soluções muito próximas principalmente quando os valores da matriz de covariância tendem para valores próximos a zero e quando as soluções estão muito próximas da fronteira Pareto. Os passos executados para gerar as novas soluções pelas abelhas nutrizes são mostrados a seguir:

- Seleciona-se as soluções do cluster c e calcula-se a média e a matriz de covariância Σ^c desta subpopulação;
- Repara-se a matriz de covariância: $CMR(\Sigma^c)$;
- Em seguida decompõem-se a matriz de covariância do cluster c utilizando a decomposição de Cholesky: $L^c L^{cT} = \Sigma^c$;
- É gerado um vetor com distribuição normal com $\mu = 0$ e $\sigma = 1$: $Z = N(0, 1)$;
- Finalmente calcula-se as novas soluções usando a equação: $x_i = \mu^c + L^c Z$.

Considerando também as restrições do problema ajusta-se cada nova solução x_i conforme as restrições de cada variável

$$x_{i,d} = Min(Max(x_{i,d}, x_d^{min}), x_d^{max}) \tag{4.10}$$

4.7 Abelhas Escoteiras

Devido ao fato do algoritmo convergir com facilidade para mínimos locais um novo tipos abelha escoteira foi adicionado cuja função é aumentar a diversidade a partir da mutação de umas das variáveis de uma solução escolhida aleatoriamente da população armazenada no arquivo externo. Para produzir esta mutação utiliza-se a distribuição de Cauchy centrada na média da variável x_d com parâmetro $t = 1$ conforme a equação 4.11.

$$x_{i,d} = Min(Max(C(\mu_d, 1), x_d^{min}), x_d^{max}) \tag{4.11}$$

onde $C(\mu_d, 1)$ corresponde a um número aleatório produzido a partir da distribuição de Cauchy.

4.8 Arquivo Externo

Com o propósito de preservar as melhores soluções não dominadas utiliza-se um arquivo externo. Esse arquivo é o mesmo utilizado pelo MOABC e consiste de uma grade com caixas de tamanho ϵ com um número de dimensões igual ao espaço objetivo. Cada uma destas caixas armazena apenas uma solução; caso existam duas

soluções aquela que dominar a outra é preservada e a outra é removida. Caso ainda exista mais de uma solução na caixa, aquela que estiver mais próxima do canto esquerdo da caixa é mantida e as outras removidas.

Este arquivo externo também é utilizado no cálculo da pontuação de cada tipo abelha e consiste de uma simples regra; para cada nova solução armazenada no arquivo é incrementado um ponto caso contrário nenhum ponto é adicionado. Os pontos de cada abelha são representados aqui por pt_c, pt_o, pt_n e pt_e referentes as abelhas campistas, observadoras, nutrizes e escoteiras respectivamente. Estes pontos serão zerados a cada iteração do algoritmo de forma a evitar o acúmulo de pontos.

Essa pontuação será utilizada para o cálculo do feromônio de controle o qual irá regular proporção de cada tipo de abelha. Esse feromônio é utilizado apenas no algoritmo MOEDABC^{Phe} com a finalidade de verificar o efeito da variação da proporção de cada tipo de abelha no desempenho do algoritmo em relação ao algoritmo MOEDABC o qual não utiliza o feromônio e sim uma proporção fixa de cada população.

4.9 Controle da População

O controle de cada nova solução produzida por cada abelha no algoritmo MOEDABC em cada iteração é realizado usando uma roleta, sendo que a probabilidade das soluções serem produzidas por cada tipo de abelha variando entre 0 e 1 é de $pop_c = 0.4$ para abelhas campistas, $pop_o = 0.3$ para abelhas observadoras, $pop_n = 0.2$ para abelhas nutrizes e $pop_s = 0.1$ para abelhas escoteiras. Porém como é utilizado uma roleta com distribuição uniforme entre 0 e 1 é necessário calcular a função de distribuição acumulada cdf para cada tipo de abelha da seguinte forma:

$$\begin{aligned} cdf_c &= pop_c \\ cdf_o &= cdf_c + pop_o \\ cdf_n &= cdf_o + pop_n \\ cdf_e &= cdf_n + pop_e \end{aligned} \tag{4.12}$$

onde cdf_c,cdf_o,cdf_n e cdf_e correspondem a função de distribuição acumulada para cada uma das abelhas.

Visto que estes valores foram adotados empiricamente utiliza-se aqui outra estratégia a qual é empregada no algoritmo denominado MOEDABC^{Phe}. Nesta estratégia o controle da proporção das soluções produzidas por cada tipo de abelha é controlada por um feromônio de controle ε que indica a qualidade das soluções e uma taxa de abandono ρ que regula a probabilidade de uma abelha mudar de tarefa. Para o cálculo do feromônio considera-se as seguintes informações:

- N_a: número de avaliações da função objetivo;
- N_t: número total de avaliações que o algoritmo irá executar;
- pt_c: pontuações da abelha campista;
- pt_o: pontuações da abelha olheira;
- pt_n: pontuações da abelha nutriz;

Algoritmo 5: Control Pheromone

Data: ρ,ε

begin

//**Calcula o valor total de pontos obtidos por cada tipo de abelha**

$pt_{total} = pt_c + pt_o + pt_n + pt_e$

//**Calcula o feromônio de cada tipo de abelha**

$ph_c = \frac{pt_c}{pt_{total}} + (1 - \varepsilon) \times ph_c$

$ph_o = \frac{pt_o}{pt_{total}} + (1 - \varepsilon) \times ph_o$

$ph_n = \frac{pt_n}{pt_{total}} + (1 - \varepsilon) \times ph_n$

$ph_e = \frac{pt_e}{pt_{total}} + (1 - \varepsilon) \times ph_e$

//**Zera as pontuações**

$pt_c = pt_o = pt_n = pt_e = 0$

//**Atualiza taxa de remoção**

$\rho_e = \frac{\rho \times N_a}{N_t}$

//**Remove parte das subpopulações de cada tipo de abelha**

$pop_c = (1 - \rho_e) \times pop_c$

$pop_o = (1 - \rho_e) \times pop_o$

$pop_n = (1 - \rho_e) \times pop_n$

$pop_e = (1 - \rho_e) \times pop_e$

//**Atualiza as populações de acordo com a taxa de feromônio**

$ph_{total} = ph_c + ph_o + ph_n + ph_e$

$pop_c += \rho_e \times \frac{ph_c}{ph_{total}}$

$pop_o += \rho_e \times \frac{ph_o}{ph_{total}}$

$pop_n += \rho_e \times \frac{ph_n}{ph_{total}}$

$pop_e += \rho_e \times \frac{ph_e}{ph_{total}}$

//**Normaliza as populações**

$pop_{total} = pop_c + pop_o + pop_n + pop_e$

$pop_c = \frac{pop_c}{pop_{total}}$

$pop_o = \frac{pop_o}{pop_{total}}$

$pop_n = \frac{pop_n}{pop_{total}}$

$pop_e = \frac{pop_e}{pop_{total}}$

//**Atualiza a função de distribuição acumulada**

$cdf_c = pop_c$

$cdf_o = cdf_c + pop_o$

$cdf_n = cdf_o + pop_n$

$cdf_e = cdf_n + pop_e$

end

- pt_e: pontuações da abelha escoteira.

O procedimento para o cálculo de cada feromônio pode se visto no Algoritmo 5. Para diminuir a probabilidade de uma abelha mudar de tarefa durante os instantes iniciais utiliza-se o parâmetro ρ_e o qual será proporcional ao número de avaliações variando linearmente entre 0 e ρ.

No processo de seleção tanto do MOEDABC quanto do MOEDABC^{Phe} utiliza-se o mesmo método de seleção elitista do NSGA-II, o *Fast Nondominated Sorting* juntamente com o *Crowding Distance* e um arquivo externo para manter as melhores soluções encontradas. A seguinte nomenclatura será adotada

- P_t: População de soluções atual;

- B_t: Nova população produzida;
- F_n: Última fronteira da população P_t obtida após seu ranqueamento usando *Fast Nondominated Sorting*, onde a população P_t é decomposta nas fronteiras $F_1, F_2, ..., F_n$;
- P_{t+1} : População final.

4.10 Algoritmos

Nesta seção são mostrados os Algoritmos 6, 7 e 8 desenvolvidos neste trabalho. O primeiro denominado MOEDABC é uma versão que não apresenta o feromônio de controle e sim uma proporção fixa de cada tipo de abelha. O segundo denominado MOEDABC^{Phe} apresenta um feromônio para o controle da população. Finalmente o algoritmo UMDA$_c^{GC}$ corresponde a uma versão modificada do UMDA$_c^G$ onde é acrescentado apenas o operador de mutação de Cauchy. A finalidade de cada um destes algoritmos é verificar se realmente os métodos utilizados pelas abelhas observadoras e nutrizes nos algoritmos MOEDABC e MOEDABC^{Phe} melhoram o desempenho desses em relação ao UMDA$_c^{GC}$ e se a utilização de um feromônio para o controle da produção das soluções por cada tipo abelha pode levar a melhores resultados o MOEDABC^{Phe} com relação ao MOEDABC e demais algoritmos avaliados.

Algoritmo 6: MOEDABC

Data: Problema, número total de avaliações N_t, número de clusters c
Result: População $P_t + Q_t$

begin
  //**Gera t soluções aleatoriamente**;
  $P_t = generatesPopulation(t)$;
  //**Avalia a população**
  $evaluate(P_t)$
  repeat
    //**Estatísticas de toda a população**;
    $\mu = mean(P_t)$
    $std = standardDeviation(P_t)$
    //**Divide a população em n clusters com o K-means**
    $P_C = K\text{-}Means(P_t)$
    //**Estatísticas para cada cluster c**
    for *each* $c \in C$ **do**
      $\mu^c = mean(P_c)$
      $std^c = standardDeviation(P_c)$
      $L^c = RECEDA(P_c)$
    end
    //**Gera as novas soluções B_t**
    for $i = 1$ *to* $length(P_t)$ **do**
      //**Roda roleta para selecionar abelha**
      $rouletteWheel = rand[0, 1]$
      //**Envia abelhas campistas**
      if $rouletteWheel < 0.4$ **then**
        $B_{t_i} = UMDA_c^G(\mu, std)$
      end
      //**Envia abelhas observadoras**
      else if $rouletteWheel < 0.7$ **then**
        //**Seleciona um cluster c randomicamente e produz uma nova solução**
        $B_{t_i} = UMDA_c^G(\mu^c, std^c)$
      end
      //**Envia abelhas Nutrizes**
      else if $rouletteWheel < 0.9$ **then**
        //**Seleciona um cluster c randomicamente e produz uma nova solução**
        $B_{t_i} = \mu_c + L^c Z$
      end
      //**Envia abelhas escoteiras**
      else
        $B_{t_i} = mutationCauchy(Q_t)$;
      end
    end
    //**Avalia a população**
    $evaluate(B_t)$;
    //**Tenta arquivar as soluções não dominadas no arquivo externo**
    $Q_t.arquive(B_t)$
    //**Combina populações**
    $P_{t+1} = P_t + B_t$
    //**Ordena população com Fast non-dominated sorting**
    $P_{t+1} = (F_1, F_2, ..., F_n) = sorting(P_{t+1})$
    //**Ordena F_n com Crowding Distance**
    $crowdingDistance(F_n)$
    //**Trunca população pela metade**
    $P_{t+1} = truncate(P_{t+1})$
  until $N_a < N_t$;
end

Algoritmo 7: MOEDABC^{Phe}

Data: Problema, número total de avaliações N_t, número de clusters c
Result: População $P_t + Q_t$

begin
  //**Gera t soluções aleatoriamente**;
  $P_t = generatesPopulation(t)$;
  //**Avalia a população**
  $evaluate(P_t)$
  repeat
    //**Estatísticas de toda a população**;
    $\mu = mean(P_t)$
    $std = standardDeviation(P_t)$
    //**Divide a população em n clusters com o K-means**
    $P_C = K\text{-}Means(P_t)$
    //**Estatísticas para cada cluster c**
    for *each* $c \in C$ **do**
      $\mu^c = mean(P_c)$
      $std^c = standardDeviation(P_c)$
      $L^c = RECEDA(P_c)$
    end
    //**Gera as novas soluções B_t**
    for $i = 1$ *to* $length(P_t)$ **do**
      //**Roda roleta para selecionar abelha**
      $rouletteWheel = rand[0, 1]$
      //**Envia abelhas campistas**
      if $rouletteWheel < cdf_c$ **then**
        $B_{t_i} = UMDA_c^G(\mu, std)$
      end
      //**Envia abelhas observadoras**
      else if $rouletteWheel < cdf_o$ **then**
        //**Seleciona um cluster c randomicamente e produz uma nova solução**
        $B_{t_i} = UMDA_c^G(\mu^c, std^c)$
      end
      //**Envia abelhas Nutrizes**
      else if $rouletteWheel < cdf_n$ **then**
        //**Seleciona um cluster c randomicamente e produz uma nova solução**
        $B_{t_i} = \mu_c + L^c Z$
      end
      //**Envia abelhas escoteiras**
      else
        $B_{t_i} = mutationCauchy(Q_t)$;
      end
    end
    //**Avalia a população**
    $evaluate(B_t)$;
    //**Tenta arquivar as soluções não dominadas no arquivo externo**
    $Q_t.arquive(B_t)$
    //**Combina populações**
    $P_{t+1} = P_t + B_t$
    //**Ordena população com Fast non-dominated sorting**
    $P_{t+1} = (F_1, F_2, ..., F_n) = sorting(P_{t+1})$
    //**Ordena F_n com Crowding Distance**
    $crowdingDistance(F_n)$
    //**Trunca população pela metade**
    $P_{t+1} = truncate(P_{t+1})$
  until $N_a < N_t$;
end

Algoritmo 8: UMDA$_c^{GC}$

Data: Problema, número total de avaliações N_t
Result: População $P_t + Q_t$

begin
  //**Gera** t **soluções aleatoriamente**;
  $P_t = generatesPopulation(t)$;
  //**Avalia a população**
  $evaluate(P_t)$
  repeat
    //**Estatísticas de toda a população**;
    $\mu = mean(P_t)$
    $std = standardDeviation(P_t)$
    //**Gera as novas soluções** B_t
    for $i = 1$ *to* $length(P_t)$ **do**
      //**Roda roleta para selecionar abelha**
      $rouletteWheel = rand[0, 1]$
      if $rouletteWheel < 0.9$ **then**
        $B_{t_i} = UMDA_c^G(\mu, std)$
      end
      else
        $B_{t_i} = mutationCauchy(Q_t)$;
      end
    end
    //**Avalia a população**
    $evaluate(B_t)$;
    //**Tenta arquivar as soluções não dominadas no arquivo externo**
    $Q_t.arquive(B_t)$
    //**Combina populações**
    $P_{t+1} = P_t + B_t$
    //**Ordena população com Fast non-dominated sorting**
    $P_{t+1} = (F_1, F_2, ..., F_n) = sorting(P_{t+1})$
    //**Ordena** F_n **com Crowding Distance**
    $crowdingDistance(F_n)$
    //**Trunca população pela metade**
    $P_{t+1} = truncate(P_{t+1})$
  until $N_a < N_t$;
end

Capítulo 5

Experimentos

Neste capítulo introduz-se alguns indicadores para análise de desempenho de algoritmos utilizados na resolução de problemas multiobjetivo. Conforme Zitzler et al. (2002) o objetivo desses indicadores é medir a capacidade destes algoritmos de encontrar um conjunto aproximado tão próximo da fronteira Pareto-ótima cobrindo esta com um número maior e diverso de soluções. Com este propósito selecionou-se os indicadores *Additive Epsilon Indicator*, *Hypervolume*, *Inverted Generational Distance*, *Maximum Pareto Front Error* e *Spacing* para a análise de nossos resultados.

Também neste capítulo introduz-se os testes estatísticos *Mann-Whitney* e *Kruskal-Wallis* com a finalidade de verificar se os resultados são realmente diferentes com certo nível de significância, em seguida são mostrados os problemas testes utilizados nos experimentos e o planejamento experimental.

5.1 Indicadores

5.1.1 Additive Epsilon Indicator

O indicador *Additive Epsilon Indicator* proposto por Zitzler et al. (2003) compara dois conjuntos de soluções A e B, onde B é o conjunto de referência, calculando o menor valor de translação ϵ do conjunto A de tal forma que o conjunto B seja totalmente dominado por A em ao menos uma solução conforme ilustrado na figura 5.1. Logo um vetor $y^1 = (y_1^1, y_2^1, ..., y_n^1) \in \mathcal{O}$ é dito ϵ-dominar outro vetor objetivo $y^1 = (y_1^2, y_2^2, ..., y_n^2) \in \mathcal{O}$, escrito como $y^1 \preceq_\epsilon y^2$, se e somente se

$$\forall i | 1 \leq i \leq n : y_i^1 \leq \epsilon * y_i^2 \tag{5.1}$$

para um dado $\epsilon > 0$. Define-se o ϵ-indicador binário I_ϵ como

$$I_\epsilon(A, B) = \min_{\epsilon \in \mathbb{R}} \{\forall y^2 \in B \; \exists y^1 \in A : y^1 \preceq_\epsilon y^2\} \tag{5.2}$$

5.1.2 Hypervolume

O indicador *Hypervolume* ou "$\mathcal{S}$-*metric*" foi proposto por Zitzler e Thiele (1998) e consiste em calcular o *hypervolume*, área no caso de problemas com dois objetivos,

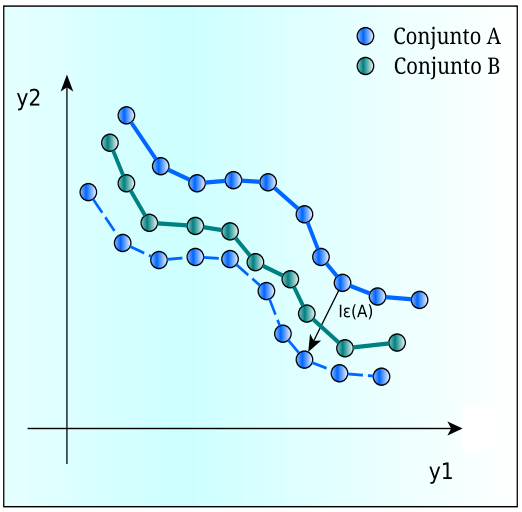


Figura 5.1: Additive Epsilon Indicator

entre uma solução do espaço de objetivos e um ponto de referência o qual deve ser dominado por todas as soluções encontradas pelos algoritmos testados. Assim quanto maior o valor do *hypervolume* mais próximo da fronteira Pareto as soluções estarão e melhor será o algoritmo.

O cálculo do *hypervolume* de regiões multidimensionais de um conjunto de soluções A e B a um ponto de referência é ilustrado na figura 5.2. Neste caso pode-se notar que o valor do *hypervolume* depende da escolha de um ponto arbitrário z^{ref} o qual tem influência direta sobre o *hypervolume* referente aos conjuntos de soluções não dominadas que estão sendo analisados (Knowles e Corne, 2002).

5.1.3 Generational Distance

O indicador *Generational Distance* permite estimar o quão longe estão um conjunto de vetores não dominados de um conjunto Pareto-ótimo (Van Veldhuizen e Lamont, 1998). Este indicador é definido como:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{5.3}$$

onde n é o número de vetores do conjunto de soluções não dominadas e d_i é a distância Euclidiana de um vetor ao membro mais próximo da fronteira Pareto verdadeira. Porém o problema deste indicador é que somente a fronteira Pareto verdadeira é considerada e não um uniforme espalhamento das soluções ao longo da fronteira. Devido a isto adota-se neste trabalho o indicador *Inverted Distance* que corrige esta deficiência.

5.1.4 Inverted Generational Distance

Este indicador permite estimar o quão longe estão um conjunto Pareto Ótimo do conjunto de vetores não dominados (Van Veldhuizen e Lamont, 1998). Formalmente,

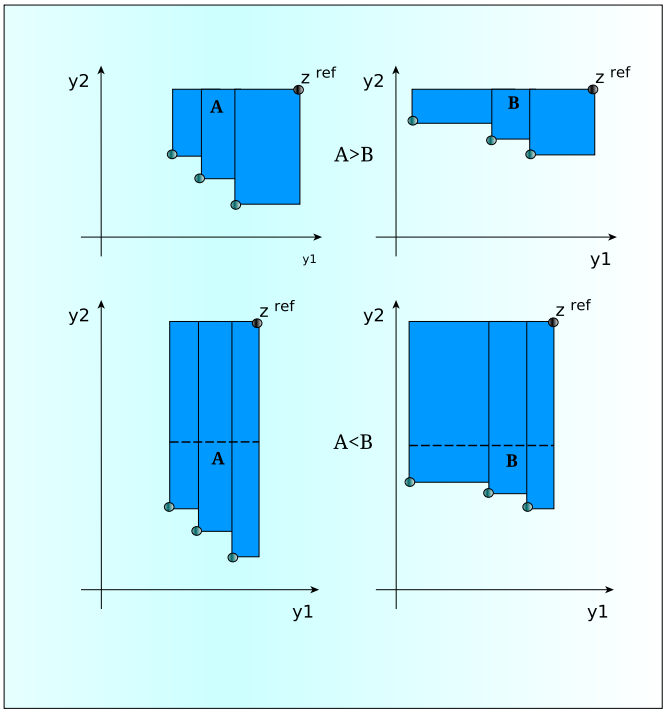


Figura 5.2: Hipervolume adaptado de Knowles e Corne (2002)

$$IGD = \frac{\sqrt{\sum_{i=1}^{m_e} d_i^2}}{m_e} \tag{5.4}$$

no qual me é o número de soluções da fronteira Pareto verdadeira P_{true} e d_i é a menor distância Euclidiana de P_{true} às soluções candidatas. Valores de IGD próximos a zero indicam que as soluções não dominadas estão bem próximas de P_{true}, por outo lado valores grandes indicam que elas estão afastadas.

5.1.5 Maximum Pareto Front Error

O indicador *Maximum Pareto Front Error* (MPFE) mede a máxima distância de um vetor do conjunto solução PF_{known} a um vetor do conjunto solução de referência P_{true} permitindo avaliar o quão separados eles estão e o quão seus formatos se assemelham (Veldhuizen e Veldhuizen, 1999). A MPFE é calculada a partir da maior das menores distâncias entre cada vetor conhecido de PF_{known} ao vetor correspondente mais próximo de P_{true}. A definição deste indicador é dada por:

$$\max_j(\min_i|f_1^i(\vec{x}) - f_1^j(\vec{x})|^p + |f_2^i(\vec{x}) - f_2^j(\vec{x})|^p)^{1/p} \tag{5.5}$$

onde $i = 1, ..., n_1$ e $j = 1, ..., n_2$ são os índices dos vetores de PF_{known} e PF_{True} respectivamente, e $p = 2$.

5.1.6 Spacing

Schott (1995) propôs o indicador *Spacing* o qual mede o nível de variância das distâncias de soluções vizinhas na fronteira PF_{known}. Este indicador é dado por:

$$s = \sqrt{\frac{1}{n-1}\sum_{i-1}(\bar{d} - d_i)^2} \tag{5.6}$$

$$d_i = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|) \tag{5.7}$$

onde $i, j = 1, 2, ..., n$, $\bar{d}$ é a média de todas as distâncias d_i e n é o número de soluções.

Um valor igual a zero para este indicador geralmente significa que todas as soluções encontradas estão igualmente espaçadas uma da outra. Valores pequenos indicam que a fronteira Pareto apresenta soluções espalhadas de forma mais homogênea.

5.2 Problemas

Nos experimentos são utilizados os problemas testes ZDT1, ZDT2, ZDT3, DTLZ2, DTLZ4 e DTLZ7 (Coello et al., 2007) além dos problemas testes UF1, UF2, UF3 e UF6 do CEC 2009 (Zhang et al., 2008). Considerando cada um destes testes com 2 funções objetivos e 100 variáveis de decisão, exceto os problemas DTLZ2, DTLZ4 e DTLZ7 que são utilizados com 3 funções objetivos.

5.3 Testes estatísticos

Neste capítulo descreve-se os testes estatísticos não-paramétricos utilizados na comparação dos resultados obtidos por cada algoritmo em relação a cada um dos problemas testes abordados.

5.3.1 Teste Mann-Whitney

O teste Mann-Whitney é um teste estatístico não-paramétrico para comparar as médias de duas amostras independentes com distribuições desconhecidas e de tamanho iguais ou diferentes. Conforme DeLong et al. (1988) este teste estima a probabilidade, θ, de uma seleção de observações aleatórias de uma população representada por P_2 ser menor ou igual a uma seleção de observações aleatórias de uma população representada por P_1. Nos testes considera-se as seguintes hipóteses referentes aos resultados obtidos em cada métrica M utilizada na análise das populações de soluções geradas pelos algoritmos A_1 e A_2:

- H_0^M: Os resultados obtidos em cada métrica referentes a A_1 e A_2 têm as mesmas médias;
- H_1^M: Os resultados têm médias diferentes e correspondem a diferentes distribuições.

5.3.2 Teste Kruskal-Wallis

O teste Kruskal-Wallis é um teste estatístico não-paramétrico para comparar as médias de k amostras independentes com distribuições desconhecidas e de tamanhos iguais ou diferentes. Para $k = 2$, o teste de Kruskal-Wallis é idêntico ao teste Mann-Whitney, porém são consideradas as seguintes hipóteses:

- H_0^K: As amostras obtidas de cada métrica referentes a K algoritmos têm as mesmas médias;
- H_1^K: Ao menos duas amostras apresentam médias diferentes .

5.3.3 Indicadores dos testes

Conforme mencionado no capítulo referente ao *framework* MOEA tanto o teste Mann-Whitney quanto o teste Kruskal-Wallis são utilizados nos testes de significância, porém como analisado nos códigos fontes destes testes implementados no MOEA, verificou-se que o MOEA adota a seguinte estratégia nos testes:

- H_0^K é verdadeira: Inicialmente é realizado o teste Kruskal-Wallis com a finalidade de verificar se a hipótese nula H_0^K é verdadeira, caso verdadeiro o teste Mann-Whitney é executado com o objetivo de verificar se existe ainda, com certo nível de significância, diferenças das amostras da métrica M obtidas das populações de cada algoritmo tomando estas par a par. Caso a hipótese H_0^M seja rejeitada considera-se que existe diferença entre os resultados dos dois algoritmos com certo nível de significância;
- H_0^K é rejeitada: Caso a hipótese H_0^K seja rejeitada o teste Mann-Whitney não é executado e o teste Kruskal-Wallis retorna os grupos de algoritmos (mais de um algoritmo) cujo resultado referente a métrica M são indiferentes ao algoritmo avaliado. Este processo é feito para todos algoritmos retornando os grupos que são indiferentes com certo grau de significância.

A partir dos resultados obtidos nos testes acima são considerados os seguintes indicadores na comparação dos resultados:

- $=$: Indica que a hipótese nula H_0^K é verdadeira e que não existe diferenças entre os resultados da métrica M para os algoritmos A_1 e A_2;
- $<$: A hipótese alternativa H_1^K é verdadeira e as amostras da métrica M para os algoritmos indicam que A_2 é pior que A_1;
- $>$: A hipótese alternativa H_1^K é verdadeira e as amostras da métrica M para os algoritmos indicam que A_2 é melhor que A_1;
- $-$: não se aplica tal comparação no caso em que se compara o algoritmo A_1 com ele mesmo.

Nas análises dos resultados computacionais considerar-se apenas A_1 o algoritmo MOEDABC^{Phe} e A_2 os demais algoritmos tomados um de cada vez.

5.4 Parâmetros adotados

Com o propósito de analisar o comportamento do MOEDABC^{Phe} na resolução de problemas com muitas variáveis foram utilizados os problemas testes ZDT1, ZDT2, ZDT3, DTLZ2, DTLZ4 e DTLZ7, além dos problemas testes UF1, UF2, UF3, UF4 e UF6 considerando 100 variáveis de decisão e 2 objetivos, exceto para os problemas DTLZ2, DTLZ4 e DTLZ7 que são com 3 objetivos. Os resultados obtidos na resolução desses problemas pelos algoritmos MOEDABC, MOEDABC^{Phe} e UMDA$_c^{GC}$ são comparados com os resultados obtidos pelos algoritmos GDE3, NSGA-II e MOEA/D. Cada algoritmo foi executado 50 vezes para cada problema. Todos algoritmos utilizam uma população de 100 indivíduos além de um arquivo externo com parâmetro $\epsilon = 0.001$ como forma de preservar as melhores soluções não dominadas encontradas. Os parâmetros utilizados dos algoritmos se baseiam nos valores padrões encontrados no *framework* MOEA utilizado para o desenvolvimento e teste de cada algoritmo. Para o algoritmo do GDE3 foram utilizados os parâmetros $CR = 0.2$ e $F = 2$, para o NSGAII a taxa de cruzamento foi de 0.8 e um operador de mutação com taxa $1/m$, onde m é o número de variáveis do problema, $F = 0.5$, $\eta = 0.01$ e um tamanho da vizinhança $T = 10$ para o MOEA/D e um número de *clusters* $k = 5$ para o MOEDABC e o MOEDABC^{Phe}, considerando para este último uma taxa de abandono $\rho = 0.01$ e a taxa de evaporação do feromônio $\varepsilon = 0.05$ com valores adotados empiricamente. O número de avaliações da função objetivo será de 20000, onde todos problemas são de minimização das funções objetivos.

Capítulo 6

Resultados Computacionais

Neste capítulo são mostrados alguns resultados computacionais obtidos referentes aos problemas testes das classes ZDT e UF. Demais resultados referentes aos problemas DTLZ podem ser vistos no Apêndice B. Cada um desses problemas é avaliado conforme os indicadores *Additive Epsilon Indicator*, *Hypervolume*, *Inverted Generational Distance*, *Maximum Pareto Front Error* e *Spacing*. Os resultados obtidos destes problemas foram produzidos pelos algoritmos GDE3, NSGA-II, e MOEA/D além dos algoritmos MOEDABC, MOEDABC^{Phe} e UMDA$_c^{GC}$ desenvolvidos neste trabalho. Nas análises dos resultados são comparados os algoritmos citados anteriormente com o MOEDABC^{Phe} baseando-se nas médias e desvios padrões obtidos e em indicadores, os quais são baseados nos testes não paramétricos de Mann-Whitney e Kruskal-Wallis definidos no capítulo 5. O intervalo de confiança utilizado nestes testes é de 95%, ou seja, *valor-p* $= 0.05$.

6.1 Análises baseadas nos indicadores

6.1.1 Additive Epsilon Indicator

Conforme o indicador *Additive Epsilon Indicator* pode-se verificar na tabela 6.1 que os algoritmos MOEDABC^{Phe} e MOEDABC apresentaram desempenho significativamente maior na resolução dos problemas da classe ZDT em relação aos demais algoritmos. É possível verificar também que o algoritmo UMDA$_c^{GC}$ apresentou um bom desempenho e que apenas no problema teste ZDT6 o algoritmo MOEDABC^{Phe} teve melhor desempenho em relação ao MOEDABC, que pode ser creditado a utilização do feromônio de controle utilizado pelo MOEDABC^{Phe} já que este é único diferencial entre os dois.

Com relação aos problemas da classe UF pode-se verificar na tabela 6.2 que nos problemas testes UF1, UF3, e UF6 o algoritmo MOEDABC^{Phe} apresentou maior desempenho. No problema teste UF2 o algoritmo NSGA-II obteve os melhores resultados. Já no problema UF4 os algoritmos GDE3 e MOEA/D apresentaram maior desempenho.

Tabela 6.1: Problemas da classe ZDT: Indicador Additive Epsilon Indicator

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
ZDT1	Média	0,12854595	0,15623485	**0,00219083**	**0,00215671**	0,91887403	0,05706258
	Desvio Padrão	0,01061681	0,02093208	0,00020989	0,00021812	0,09836547	0,00338918
	Máximo	0,15618953	0,22220336	0,00286151	0,00263033	1,12710650	0,06468216
	Minímo	0,11024005	0,12455353	0,00180507	0,00182795	0,60688224	0,05075874
	Indicador	<	<	=	-	<	<
ZDT2	Média	0,41619202	0,57785472	**0,00232270**	**0,00220829**	2,38745198	0,09567026
	Desvio Padrão	0,02822656	0,07087621	0,00161873	0,00137455	0,19196350	0,02598370
	Máximo	0,47909889	0,70921250	0,00742326	0,00632697	2,82115690	0,12895558
	Minímo	0,36293735	0,45139864	0,00096382	0,00099251	1,90226086	0,00529356
	Indicador	<	<	=	-	<	<
ZDT3	Média	0,17905534	0,15323863	**0,00861955**	**0,00900595**	0,60954866	0,19372120
	Desvio Padrão	0,01012620	0,01469964	0,00188815	0,00207677	0,07056836	0,00487606
	Máximo	0,19723511	0,19079426	0,01203840	0,01281259	0,77520111	0,20138681
	Minímo	0,15805450	0,13453141	0,00420667	0,00129365	0,43240331	0,17679523
	Indicador	<	<	=	-	<	<
ZDT4	Média	253,65661276	48,00797485	**0,00096627**	**0,00096316**	48,83073841	0,45596802
	Desvio Padrão	11,30118662	3,44649591	0,00008026	0,00008855	4,71670890	0,57910886
	Máximo	278,49101879	54,94690593	0,00134049	0,00137490	59,82863863	2,57021612
	Minímo	225,07253762	41,14159882	0,00080751	0,00080657	37,31164682	0,00164440
	Indicador	<	<	=	-	<	<
ZDT6	Média	4,97822863	5,00588313	0,13647035	**0,10571769**	7,26894064	2,87745719
	Desvio Padrão	0,07698007	0,17340486	0,06777322	0,04992293	0,14876589	0,99313164
	Máximo	5,15199470	5,48289164	0,30296524	0,20199333	7,62826124	4,35674917
	Minímo	4,81574167	4,54999174	0,01748564	0,02378009	6,90370593	0,42927172
	Indicador	<	<	<	-	<	<

Tabela 6.2: Problemas da classe UF: Indicador Additive Epsilon Indicator

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
UF1	Média	**0,24074496**	0,26449161	0,24919271	**0,23335630**	0,45243183	0,27273161
	Desvio Padrão	0,03886917	0,05091243	0,03975149	0,02126042	0,08247425	0,03615320
	Máximo	0,33751071	0,38869822	0,43286680	0,32509497	0,72163039	0,40227592
	Minímo	0,19179228	0,19731266	0,19291569	0,19335311	0,32637556	0,22176708
	Indicador	=	<	<	-	<	<
UF2	Média	0,21602706	**0,19386113**	0,30702588	0,31188201	0,29082726	0,31728781
	Desvio Padrão	0,02099216	0,02402817	0,03245619	0,02863695	0,06873320	0,01462849
	Máximo	0,27059472	0,26265006	0,34308787	0,34256498	0,45750769	0,34423756
	Minímo	0,18488837	0,15815007	0,22561347	0,22320099	0,18971113	0,26893034
	Indicador	>	>	=	-	>	=
UF3	Média	0,52171102	0,50657035	**0,37937703**	**0,39252020**	0,62612576	0,55639168
	Desvio Padrão	0,02071573	0,03427300	0,03241113	0,06261121	0,03746511	0,01566609
	Máximo	0,55937689	0,58688522	0,44775431	0,65040228	0,73317833	0,58961298
	Minímo	0,46614173	0,42325726	0,29789464	0,31890496	0,49519717	0,51732639
	Indicador	<	<	=	-	<	<
UF4	Média	**0,11290361**	0,11892456	0,15227424	0,13530813	**0,11219707**	0,15775748
	Desvio Padrão	0,00244035	0,00399293	0,00232590	0,00292385	0,00393796	0,00165773
	Máximo	0,11794700	0,12832408	0,15767796	0,14142363	0,12224705	0,16258991
	Minímo	0,10605111	0,11172530	0,14656164	0,12751479	0,10548662	0,15449562
	Indicador	<	<	<	-	<	<
UF6	Média	0,56133472	0,69474409	0,41117276	**0,35750565**	1,48053707	0,58518013
	Desvio Padrão	0,14493638	0,22182615	0,12266163	0,05420346	0,18865017	0,17756294
	Máximo	1,05913503	1,40393964	0,83415812	0,65216038	1,97661624	0,94389780
	Minímo	0,38392848	0,41627452	0,33498590	0,32212526	1,01798531	0,31916216
	Indicador	<	<	<	-	<	<

6.1.2 Hypervolume

Da mesma forma que na análise do indicador anterior, verificou-se a partir dos resultados da tabela 6.3 que ambos algoritmos MOEDABC^{Phe} e MOEDABC apresentaram maior desempenho com relação ao indicador Hypervolume referentes aos problemas da classe ZDT, inclusive em relação ao problema ZDT6 no qual novamente o MOEDABC^{Phe} apresentou uma diferença significativa em relação ao MOEDABC.

Referente a essa mesmo indicador, mas agora considerando os problemas da classe UF, pode-se verificar na tabela 6.4 que o MOEDABC^{Phe} apresentou maior *hypervolume* nos problemas UF1, UF2, UF3 e UF6, no entanto em todos esses testes houve ao menos um algoritmo com resultados, com certo nível de significância, iguais ao do MOEDABC^{Phe}, exceto no problema teste UF6.

6.1.3 Inverted Distance

Nesta seção são avaliados os desempenhos dos algoritmos utilizando o indicador *Inverted Distance* com o objetivo de verificar a proximidade das soluções da fronteira Pareto-ótima. Analisando a tabela 6.5 pôde-se verificar também que os algoritmos MOEDABC^{Phe} e MOEDABC apresentaram em média, e conforme os níveis de significância, soluções mais próximas a fronteira P_{true}, ressaltando apenas que no problema ZDT6 o MOEDABC^{Phe} obteve maior desempenho. Avaliando agora a tabela 6.6 verificou-se que nos problemas testes UF1, UF3 e UF6 o MOEDABC^{Phe} apresentou bons resultados, sendo que apenas no problema teste UF6 os resultados foram significativamente melhores aos demais. Ressalta-se aqui que a adoção do feromônio de controle proporcionou melhores resultados ao MOEDABC^{Phe} com relação ao MOEDABC.

6.1.4 Maximum Pareto Front Error

Utilizando-se o indicador *Maximum Pareto Front Error* os resultados obtidos mostram-se relativamente muito parecidos com relação aos demais indicadores citados anteriormente na avaliação do comportamento do MOEDABC^{Phe} e do MOEDABC referentes aos problemas da classe ZDT. Pôde-se verificar também na tabela 6.7 que ambos apresentam resultados significativamente melhores neste indicador, com apenas uma exceção referente ao problema ZDT6 onde NSGA-II obteve maior desempenho. Estes resultados indicam que as soluções obtidas pelos algoritmos MOEDABC^{Phe} e MOEDABC encontram-se bem próximas a P_{true} havendo poucas distorções das proximidades das soluções em relação a fronteira.

No que se refere aos problemas da classe UF os resultados vistos na tabela 6.8 indicam que o MOEDABC^{Phe} apresentou maior desempenho em relação ao MOEDABC e demais algoritmos nos problemas UF1, UF2 e UF6, havendo apenas uma exceção no problema UF2 onde o MOEDABC é indiferente. Ressalta-se aqui também a bom desempenho do UMDA$_c^{GC}$ na resolução do problema UF3 superando os demais algoritmos.

Tabela 6.3: Problemas da classe ZDT: Indicador Hypervolume

Problema	**Medida**	**GDE3**	**NSGA-II**	**MOEDABC**	**MOEDABC^{Phe}**	**MOEA/D**	**UMDA$_c^{GC}$**
ZDT1	Média	0,49252945	0,46708239	**0,66468004**	**0,66464666**	0,00949853	0,58556370
	Desvio Padrão	0,00875765	0,02413244	0,00020212	0,00019787	0,01698635	0,00393621
	Máximo	0,51271769	0,51301786	0,66519418	0,66505815	0,09693957	0,59422031
	Minímo	0,46892081	0,39257249	0,66433034	0,66409511	0,00000000	0,57642806
	Indicador	<	<	=	-	<	<
ZDT2	Média	0,11398893	0,06516718	**0,33124382**	**0,33131034**	0,00000000	0,27075416
	Desvio Padrão	0,00868089	0,02056731	0,00147394	0,00123198	0,00000000	0,01594783
	Máximo	0,13070583	0,10755073	0,33269238	0,33264899	0,00000000	0,32887632
	Minímo	0,09798476	0,02946982	0,32691474	0,32796202	0,00000000	0,25124060
	Indicador	<	<	=	-	<	<
ZDT3	Média	0,35102430	0,37671927	**0,51013738**	**0,50990305**	0,07749597	0,33101371
	Desvio Padrão	0,00754866	0,01427280	0,00144880	0,00172975	0,02435633	0,00701946
	Máximo	0,36870130	0,40063288	0,51347295	0,51618075	0,14562777	0,35311598
	Minímo	0,33185140	0,33656306	0,50769589	0,50696469	0,02082015	0,31954191
	Indicador	<	<	=	-	<	<
ZDT4	Média	0,00000000	0,00000000	**0,66603726**	**0,66603449**	0,00000000	0,38614719
	Desvio Padrão	0,00000000	0,00000000	0,00002652	0,00002607	0,00000000	0,28031675
	Máximo	0,00000000	0,00000000	0,66606802	0,66606464	0,00000000	0,66602328
	Minímo	0,00000000	0,00000000	0,66596366	0,66595672	0,00000000	0,00000000
	Indicador	<	<	=	-	<	<
ZDT6	Média	0,00000000	0,00000000	0,23695226	**0,26505020**	0,00000000	0,00436317
	Desvio Padrão	0,00000000	0,00000000	0,07389712	0,05795115	0,00000000	0,02423514
	Máximo	0,00000000	0,00000000	0,39938774	0,37033508	0,00000000	0,16521287
	Mínimo	0,00000000	0,00000000	0,13669599	0,16800620	0,00000000	0,00000000
	Indicador	<	<	<	-	<	<

Tabela 6.4: Problemas da classe UF: Indicador Hypervolume

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
UF1	Média	**0,45876225**	0,41812586	0,43236712	**0,46325548**	0,22490972	0,41704630
	Desvio Padrão	0,02779993	0,05589110	0,05495099	0,03112360	0,03985862	0,03296793
	Máximo	0,49179618	0,49008300	0,50223024	0,50326447	0,29719473	0,45370042
	Minímo	0,39190024	0,29743393	0,21008116	0,35240473	0,11158787	0,26699420
	Indicador	=	<	<	-	<	<
UF2	Média	0,52181615	0,52278325	**0,54170577**	**0,54265459**	0,49055305	0,52916815
	Desvio Padrão	0,00589588	0,00849934	0,00388602	0,00398159	0,01766706	0,00225072
	Máximo	0,53843913	0,53744318	0,55351217	0,55508231	0,52141443	0,53416391
	Minímo	0,51102841	0,50241319	0,53579798	0,53679344	0,43663204	0,52498137
	Indicador	<	<	=	-	<	<
UF3	Média	0,30134346	0,30278778	**0,36908813**	**0,36771938**	0,23217891	0,30454084
	Desvio Padrão	0,00749958	0,01342187	0,02063507	0,03043544	0,01907704	0,00944634
	Máximo	0,32069687	0,33035086	0,40535266	0,43436151	0,28874210	0,33105771
	Minímo	0,28722947	0,26879142	0,32363979	0,29314168	0,17131848	0,28613095
	Indicador	<	<	=	-	<	<
UF4	Média	0,17763723	0,17487022	0,12663039	0,15131428	**0,18225544**	0,12025756
	Desvio Padrão	0,00180017	0,00293680	0,00416917	0,00443473	0,00346914	0,00141326
	Máximo	0,18191937	0,18066284	0,14065618	0,16245396	0,19070155	0,12283983
	Minímo	0,17378419	0,16872752	0,12044821	0,14206767	0,17610122	0,11759881
	Indicador	>	>	<	-	>	<
UF6	Média	0,05966013	0,03419755	0,08474191	**0,11524225**	0,00000000	0,08861199
	Desvio Padrão	0,04072155	0,03834615	0,03688029	0,02843926	0,00000000	0,05515140
	Máximo	0,14328861	0,12310803	0,12747856	0,14717637	0,00000000	0,19480248
	Minímo	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000	0,00000000
	Indicador	<	<	<	-	<	<

Tabela 6.5: Problemas da classe ZDT: Indicador Inverted Distance

Problema	Medidas	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
ZDT1	Média	0,00381162	0,00454474	**0,00003991**	**0,00004050**	0,02485553	0,00178355
	Desvio Padrão	0,00020793	0,00059564	0,00000432	0,00000432	0,00257875	0,00009014
	Máximo	0,00443527	0,00645061	0,00004727	0,00005302	0,03058691	0,00199597
	Minímo	0,00335928	0,00345753	0,00002899	0,00003188	0,01672321	0,00158984
	Indicador	<	<	=	-	<	<
ZDT2	Média	0,00703974	0,01020785	**0,00005559**	**0,00005336**	0,05784920	0,00151396
	Desvio Padrão	0,00043323	0,00146694	0,00002281	0,00001888	0,00586312	0,00040698
	Máximo	0,00794323	0,01313708	0,00012588	0,00010750	0,07111986	0,00203094
	Minímo	0,00637202	0,00759546	0,00003444	0,00003564	0,04315443	0,00009230
	Indicador	<	<	=	-	<	<
ZDT3	Média	0,00761210	0,00628324	**0,00021614**	**0,00022304**	0,02602929	0,00895716
	Desvio Padrão	0,00041266	0,00079648	0,00003926	0,00004433	0,00258446	0,00041508
	Máximo	0,00859764	0,00861061	0,00027368	0,00029947	0,03150683	0,00963404
	Minímo	0,00678183	0,00502434	0,00012738	0,00008423	0,01953441	0,00778908
	Indicador	<	<	=	-	<	<
ZDT4	Média	8,00679472	1,50692871	**0,00001625**	**0,00001630**	1,53294006	0,01226067
	Desvio Padrão	0,35719546	0,10893530	0,00000046	0,00000046	0,14906722	0,01599113
	Máximo	8,79172633	1,72627711	0,00001754	0,00001779	1,88052526	0,07301815
	Minímo	7,10333770	1,28990829	0,00001550	0,00001548	1,16891652	0,00001897
	Indicador	<	<	=	-	<	<
ZDT6	Média	0,16617579	0,16710060	0,00275459	**0,00234221**	0,24483069	0,09296751
	Desvio Padrão	0,00267419	0,00595130	0,00118991	0,00096664	0,00512013	0,03715504
	Máximo	0,17207153	0,18348741	0,00438606	0,00386893	0,25718306	0,14480426
	Mínimo	0,16058737	0,15143242	0,00010402	0,00062193	0,23228680	0,00749159
	Indicador	<	<	<	-	<	<

Tabela 6.6: Problemas da classe UF: Indicador Inverted Distance

Problemas	Medidas	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
UF1	Média	**0,00438615**	0,00517932	0,00489024	**0,00448489**	0,01125272	0,00504554
	Desvio Padrão	0,00044992	0,00114033	0,00104019	0,00057351	0,00145319	0,00084419
	Máximo	0,00557121	0,00791581	0,00997004	0,00683287	0,01525948	0,00958664
	Minímo	0,00366227	0,00370679	0,00372867	0,00382955	0,00841595	0,00435166
	Indicador	=	<	<	-	<	<
UF2	Média	0,00415869	**0,00374911**	0,00510429	0,00511652	0,00545301	0,00516154
	Desvio Padrão	0,00023958	0,00027216	0,00054119	0,00052107	0,00147226	0,00023688
	Máximo	0,00471934	0,00432463	0,00641048	0,00604983	0,00963333	0,00616117
	Minímo	0,00340399	0,00317530	0,00377283	0,00381763	0,00391372	0,00475441
	Indicador	>	>	=	-	<	=
UF3	Média	0,00918138	0,00894990	**0,00786350**	**0,00819954**	0,01132375	0,01078409
	Desvio Padrão	0,00033261	0,00039746	0,00062040	0,00166906	0,00061756	0,00035286
	Máximo	0,01014316	0,01004357	0,00936563	0,01538943	0,01337069	0,01140505
	Minímo	0,00859196	0,00809180	0,00669931	0,00637408	0,00956173	0,00969719
	Indicador	<	<	=	-	<	<
UF4	Média	0,00379828	0,00381668	0,00507852	0,00431061	**0,00362620**	0,00586825
	Desvio Padrão	0,00006525	0,00010019	0,00020015	0,00019607	0,00010554	0,00007619
	Máximo	0,00393548	0,00402272	0,00562398	0,00473993	0,00379433	0,00598427
	Minímo	0,00368102	0,00360474	0,00468270	0,00390090	0,00329244	0,00565968
	Indicador	>	>	<	-	>	<
UF6	Média	0,01476121	0,02097940	0,01187943	**0,00909901**	0,05675679	0,01526332
	Desvio Padrão	0,00448557	0,00855617	0,00481413	0,00180759	0,00744691	0,00538249
	Máximo	0,03346193	0,04462356	0,02624746	0,01697996	0,07952656	0,03414210
	Minímo	0,01044355	0,01186127	0,00869210	0,00808619	0,04507350	0,00813936
	Indicador	<	<	<	-	<	<

Tabela 6.7: Problemas da classe ZDT: Indicador Maximum Pareto Front Error

Problemas	**Medidas**	**GDE3**	**NSGA-II**	**MOEDABC**	**MOEDABC^{Phe}**	**MOEA/D**	**UMDA$_c^{GC}$**
ZDT1	Média	0,23679565	0,31012340	**0,01115770**	**0,01177113**	1,47321738	0,10752495
	Desvio Padrão	0,02157584	0,04275180	0,00405772	0,00318998	0,15209161	0,00655765
	Máximo	0,28893902	0,42399541	0,01576424	0,01568048	1,88095266	0,12428053
	Minímo	0,16484988	0,21553480	0,00223816	0,00341001	0,98556655	0,08945717
	Indicador	<	<	=	-	<	<
ZDT2	Média	0,25552724	0,34756467	**0,00207363**	**0,00200635**	1,49690849	0,05572221
	Desvio Padrão	0,01846385	0,04916099	0,00137011	0,00108324	0,18261643	0,01475937
	Máximo	0,29540102	0,45427646	0,00622170	0,00523251	2,02216172	0,07435723
	Minímo	0,22357045	0,25679502	0,00104795	0,00108546	1,18935914	0,00321832
	Indicador	<	<	=	-	<	<
ZDT3	Média	0,21838930	0,20882013	**0,01360561**	**0,01395377**	0,81253848	0,25914634
	Desvio Padrão	0,01656618	0,02547667	0,00172786	0,00207955	0,08908076	0,01371079
	Máximo	0,25763323	0,26205465	0,01632590	0,01721811	0,98592747	0,27842862
	Minímo	0,18209440	0,16660948	0,00820047	0,00651304	0,60716257	0,21820353
	Indicador	<	<	=	-	<	<
ZDT4	Média	277,39937187	57,32466847	**0,01184583**	**0,01157017**	279,10730030	0,81896037
	Desvio Padrão	15,76770801	6,94211481	0,00332886	0,00360475	166,35655613	0,92552664
	Máximo	302,53043298	75,21048366	0,01581543	0,01558886	583,69692405	3,32741357
	Minímo	234,99649900	46,13371560	0,00250247	0,00319406	50,33618525	0,00404173
	Indicador	<	<	=	-	<	<
ZDT6	Média	4,56547944	**4,26804849**	4,56370184	4,74514975	6,56780474	6,22698588
	Desvio Padrão	0,12528628	0,16761346	1,18925960	1,14351334	0,19492402	0,55047055
	Máximo	4,91060614	4,67815192	6,69881332	7,38845229	6,92292590	7,28972252
	Minímo	4,31895585	3,81257185	1,82061470	2,28766117	6,06864131	5,08008307
	Indicador	=	>	=	-	<	<

Tabela 6.8: Problemas da classe UF: Indicador Maximum Pareto Front Error

Problemas	Medidas	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
UF1	Média	0,15594447	0,23693357	0,14233802	**0,09227402**	0,48648551	0,14899251
	Desvio Padrão	0,16366852	0,14544763	0,08339514	0,06523228	0,10472430	0,09339191
	Máximo	1,07702543	0,59872923	0,45420157	0,27363732	0,79904659	0,51128955
	Minímo	0,05752912	0,05183031	0,04304450	0,02593044	0,30707500	0,05653843
	Indicador	>	>	>	-	>	>
UF2	Média	0,26844523	**0,19645045**	**0,21145610**	**0,20276263**	0,23892128	0,26173090
	Desvio Padrão	0,07943617	0,04654704	0,06180995	0,05275297	0,08956919	0,05343144
	Máximo	0,53214434	0,33745863	0,33001818	0,31467022	0,65689755	0,38735873
	Minímo	0,14233975	0,13060306	0,10761430	0,12344241	0,09694366	0,14277060
	Indicador	<	=	=	-	>	>
UF3	Média	0,28777609	0,28832523	0,25980562	0,26060988	0,28833505	**0,11486305**
	Desvio Padrão	0,06781224	0,05438031	0,03616518	0,04089616	0,05863065	0,04326121
	Máximo	0,49642054	0,44696904	0,35745602	0,35068404	0,44330045	0,24364635
	Minímo	0,17737806	0,20701024	0,19771577	0,18197763	0,19737023	0,06851019
	Indicador	<	<	=	-	<	>
UF4	Média	0,15486558	**0,16235633**	0,20897026	0,17565442	**0,16081363**	0,21475158
	Desvio Padrão	0,00399943	0,00547058	0,00361169	0,00624163	0,01018725	0,00317111
	Máximo	0,16702863	0,17368233	0,21927297	0,19512957	0,18353584	0,22275921
	Minímo	0,14633096	0,14986625	0,19824019	0,15402806	0,14469852	0,20764486
	Indicador	>	>	<	-	>	<
UF6	Média	0,49614883	0,87797566	0,50838474	**0,32708944**	1,83648794	**0,33252090**
	Desvio Padrão	0,22159093	0,69048617	0,25762305	0,16656926	0,38074279	0,20151795
	Máximo	1,36238004	2,90920408	1,37424234	0,85797151	2,94106362	1,02506702
	Minímo	0,29299155	0,27938035	0,22311851	0,16667567	1,23190015	0,10540189
	Indicador	<	<	<	-	<	=

6.1.5 Spacing

Finalmente, avaliando a tabela 6.10 verificou-se o mesmo comportamento dos algoritmos MOEDABC^{Phe} e MOEDABC para a classe de problemas ZDT, onde exceto para o problema ZDT6 os valores obtidos pelo indicador *Spacing* para os demais problemas indicam que ambos algoritmos apresentam uma boa distribuição das soluções em relação a fronteira Pareto, além de um grande número de soluções já que quanto maior o número de soluções menor será o valor obtido por este indicador. Com relação aos problemas da classe UF pode-se verificar na tabela 6.10 que o algoritmo MOEA/D apresentou melhor desempenho nos problemas UF2 e UF3, já no problema UF1 todos algoritmos apresentaram resultados com certo nível de significância iguais. Neste indicador o MOEDABC^{Phe} apresentou melhor desempenho apenas no problema teste UF6, sendo o algoritmo UMDA$_c^{GC}$ indiferente nesse problema.

Tabela 6.9: Problemas da classe ZDT: Indicador Spacing

Problemas	**Medidas**	**GDE3**	**NSGA-II**	**MOEDABC**	**MOEDABC^{Phe}**	**MOEA/D**	**UMDA$_c^{GC}$**
ZDT1	Média	0,01237875	0,00709016	**0,00119303**	**0,00118890**	0,02728601	0,00387233
	Desvio Padrão	0,00199860	0,00091082	0,00012251	0,00011592	0,00758304	0,00045824
	Máximo	0,01734600	0,01035678	0,00153335	0,00145696	0,04618987	0,00576017
	Minímo	0,00831666	0,00531774	0,00093978	0,00096430	0,01533852	0,00317964
	Indicador	<	<	=	-	<	<
ZDT2	Média	0,02431852	0,01238213	**0,00108306**	**0,00111974**	0,02513102	0,00481056
	Desvio Padrão	0,00693674	0,00383966	0,00018426	0,00025571	0,06194005	0,00112301
	Máximo	0,04319583	0,02833616	0,00181169	0,00179851	0,41753721	0,00816806
	Minímo	0,01314290	0,00767083	0,00088565	0,00085842	0,00000000	0,00217846
	Indicador	<	<	=	-	<	<
ZDT3	Média	0,01700995	0,00645059	**0,00341987**	**0,00353658**	0,04375126	0,01120006
	Desvio Padrão	0,00320961	0,00074178	0,00023070	0,00025052	0,01219745	0,00172888
	Máximo	0,02360529	0,00787316	0,00387209	0,00405158	0,07936467	0,01621874
	Minímo	0,01130197	0,00491706	0,00292340	0,00291195	0,02169671	0,00768997
	Indicador	<	<	=	-	<	<
ZDT4	Média	3,44538237	1,02275005	**0,00112147**	**0,00112200**	25,01246661	0,07438548
	Desvio Padrão	4,86384961	0,97417450	0,00010553	0,00012892	29,97110426	0,15465832
	Máximo	18,85699563	5,52151154	0,00142830	0,00141110	136,70254845	0,64506149
	Minímo	0,00000000	0,22889665	0,00090946	0,00090404	0,37992207	0,00000000
	Indicador	<	<	=	-	<	<
ZDT6	Média	0,08361836	**0,03710241**	0,06452838	0,06407178	0,07451366	0,50068400
	Desvio Padrão	0,05503848	0,01579927	0,04552955	0,03878212	0,05825434	0,30992083
	Máximo	0,30026521	0,07991901	0,27962044	0,23308727	0,39425017	1,78403921
	Minímo	0,02009857	0,01496179	0,00684137	0,00886815	0,01108299	0,10796616
	Indicador	<	>	=	-	=	<

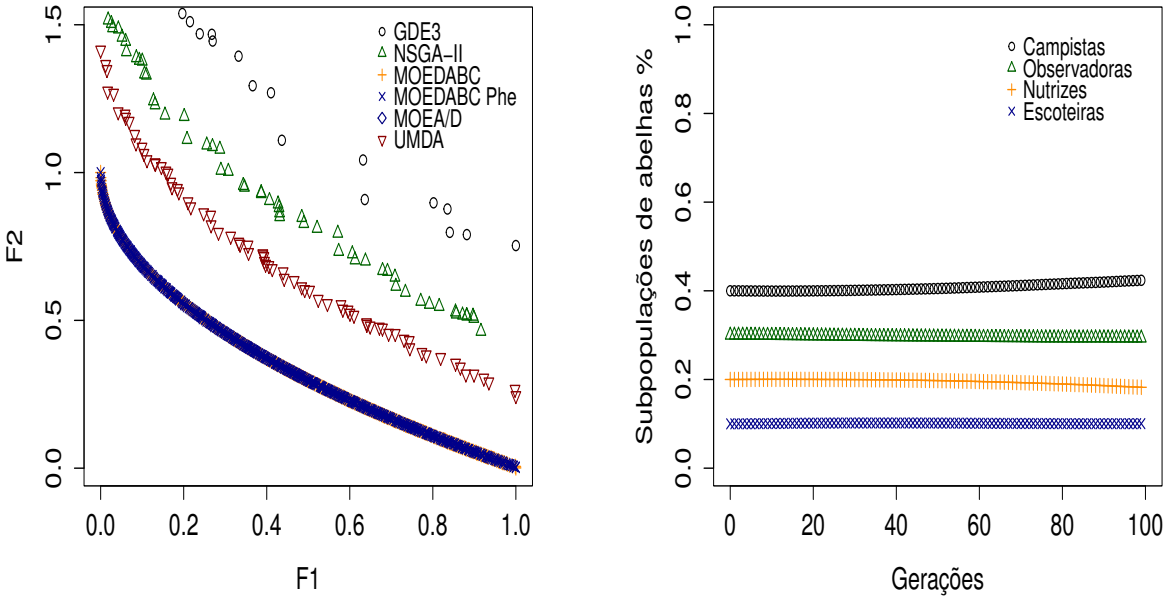
Tabela 6.10: Problemas da classe UF: Indicador Spacing

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
UF1	Média	**0,01218228**	**0,00983229**	**0,03180608**	**0,04865665**	**0,01339735**	**0,03199700**
	Desvio Padrão	0,00803527	0,00521316	0,04009889	0,06589363	0,00671858	0,05305920
	Máximo	0,03941161	0,02488432	0,14868938	0,21654970	0,03831189	0,23457727
	Minímo	0,00337559	0,00312695	0,00289355	0,00152628	0,00550690	0,00504236
	Indicador	=	=	=	-	=	=
UF2	Média	0,00984463	0,00663836	0,00906465	0,00902352	**0,00596672**	0,00814971
	Desvio Padrão	0,00309813	0,00203520	0,00721695	0,00773986	0,00209854	0,00397457
	Máximo	0,01903272	0,01418491	0,03720675	0,04155569	0,01207316	0,02538437
	Minímo	0,00508201	0,00438706	0,00340301	0,00302146	0,00323167	0,00324347
	Indicador	<	=	=	-	>	=
UF3	Média	0,00836513	0,00569666	0,01442632	0,01032787	**0,00510201**	0,00868493
	Desvio Padrão	0,00479163	0,00195293	0,02067782	0,01510049	0,00282774	0,00708012
	Máximo	0,02263829	0,01213938	0,06948559	0,06746717	0,01527748	0,02879219
	Minímo	0,00253520	0,00240514	0,00245822	0,00275739	0,00190126	0,00220079
	Indicador	<	=	=	-	>	=
UF4	Média	0,00630502	**0,00532975**	0,01433820	0,02638685	0,00324165	0,01126994
	Desvio Padrão	0,00107860	0,00070111	0,00462911	0,00722586	0,00041959	0,00226981
	Máximo	0,00933415	0,00717553	0,02828926	0,04558642	0,00457736	0,02221061
	Minímo	0,00430649	0,00396323	0,00819112	0,01377074	0,00245371	0,00794978
	Indicador	>	>	>	-	>	>
UF6	Média	0,02799544	0,02638978	0,03514110	**0,02408396**	0,04448945	**0,03238129**
	Desvio Padrão	0,01691726	0,01337921	0,03712985	0,02140978	0,04588019	0,08097775
	Máximo	0,09886332	0,07691786	0,18409325	0,09421005	0,31073342	0,49708378
	Minímo	0,00834069	0,00816730	0,00478858	0,00369945	0,01088548	0,00000000
	Indicador	<	<	<	-	<	=

6.2 Análise gráfica dos resultados

Nesta seção realiza-se uma análise gráfica dos resultados obtidos por cada algoritmo utilizado na seção anterior. Cada um desses métodos é executado uma vez para problemas com 2 objetivos com 100 variáveis de decisão. O número de avaliações das funções objetivos é de 10000 para os problemas ZDT1, ZDT2, ZDT3 e ZDT4, o restante dos problemas das classes UF utilizaram 20000 avaliações da função objetivo. Esta diferença do número de avaliações teve como objetivo melhorar a diferenciação dos resultados obtidos, destacando a velocidade de convergência dos algoritmos e sua distribuição no espaço objetivo. Cada uma das figuras é associada com outra figura que mostra a evolução da proporção das subpopulações de abelhas do algoritmo MOEDABC^{Phe}. Ressalta-se aqui que essas proporções podem variar de acordo com a população inicial de soluções, o tipo de problema e o número de gerações. Os resultados obtidos dos problemas DTLZ2, DTLZ4 e DTLZ7 também são mostrados nesta seção e são considerados aqui com apenas 2 funções objetivos.

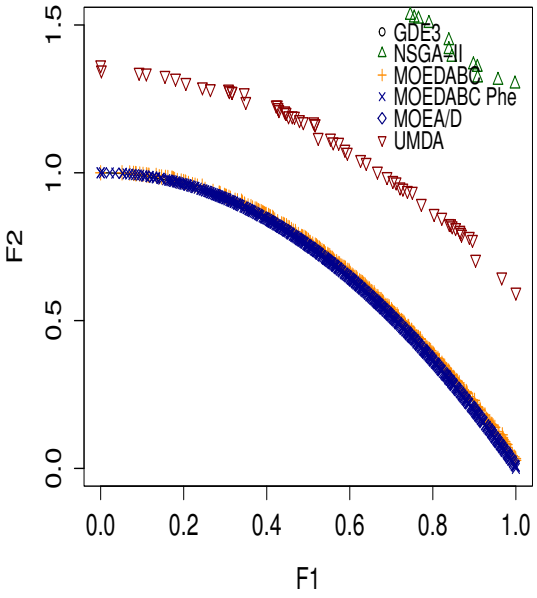
Analisando as figuras 6.1, 6.2, 6.3 e 6.4 relativos aos problemas da classe ZDT, pode-se observar que tanto o MOEDABC quanto MOEDABC^{Phe} apresentaram melhores desempenhos com relação aos demais métodos, sendo que apenas no problema ZDT6 da figura 6.5 o MOEDABC^{Phe} apresentou maior desempenho obtendo soluções mais próximas a P_{true}. Neste último gráfico pode-se ver que a proporção da subpopulação de abelhas escoteiras teve grande crescimento.



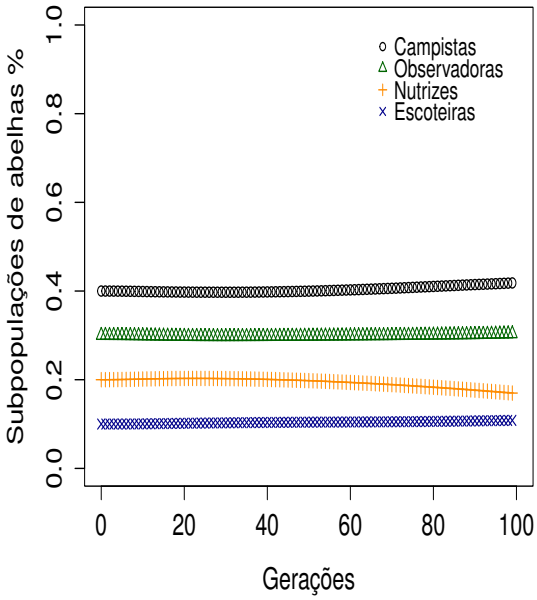
(a) Soluções obtidas por cada algoritmo

(b) Variação da proporção de cada abelha

Figura 6.1: Problema ZDT1

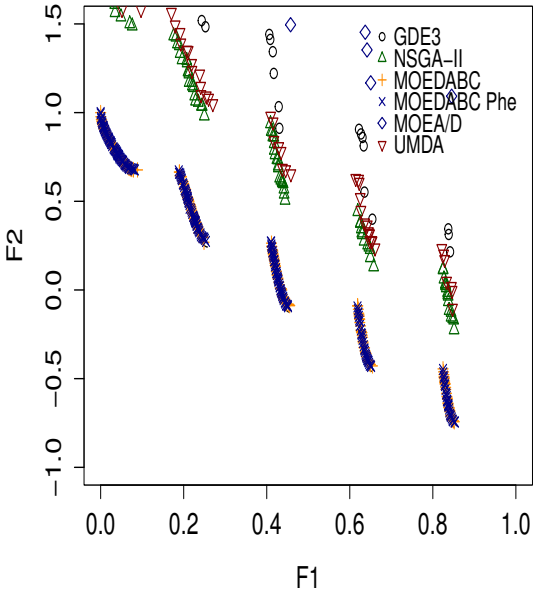


(a) Soluções obtidas por cada algoritmo

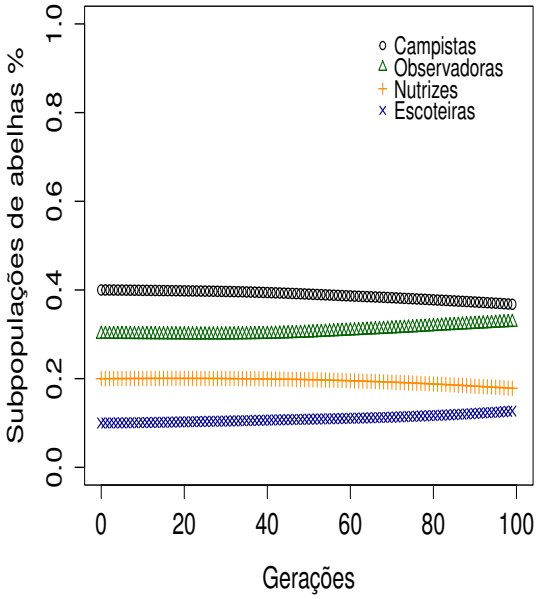


(b) Variação da proporção de cada abelha

Figura 6.2: Problema ZDT2

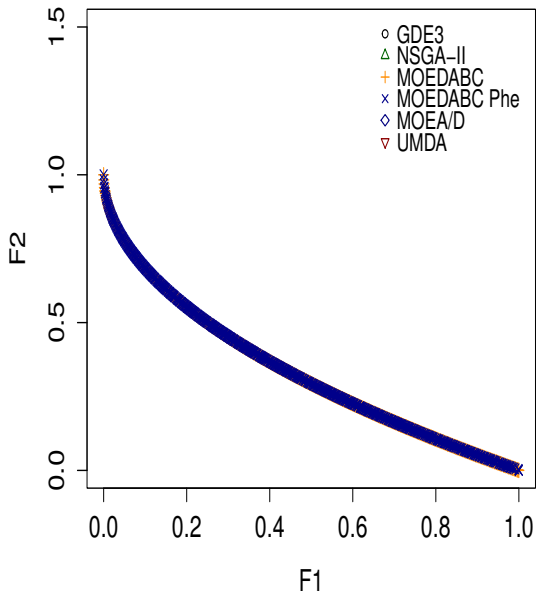


(a) Soluções obtidas por cada algoritmo

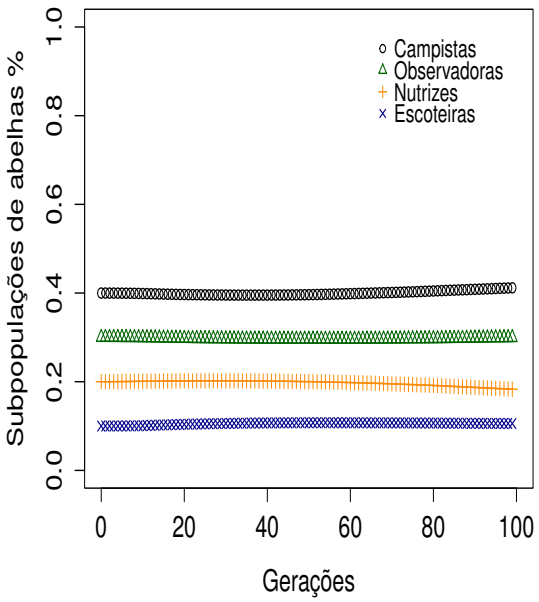


(b) Variação da proporção de cada abelha

Figura 6.3: Problema ZDT3

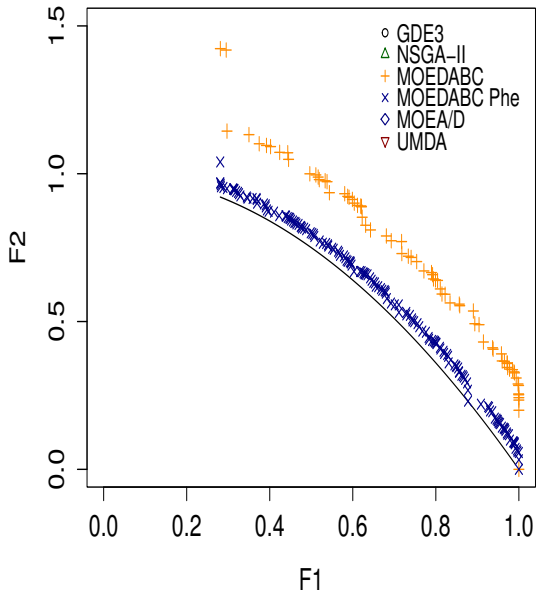


(a) Soluções obtidas por cada algoritmo

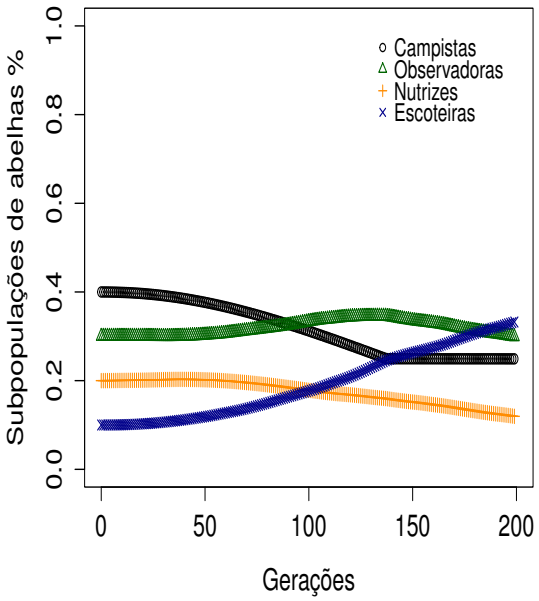


(b) Variação da proporção de cada abelha

Figura 6.4: Problema ZDT4



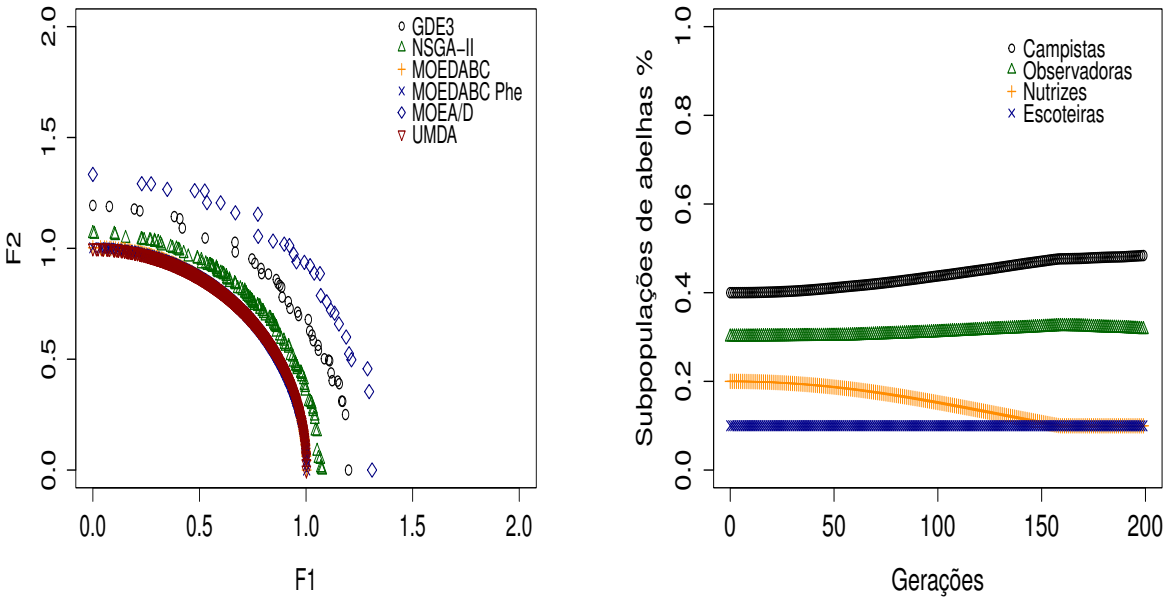
(a) Soluções obtidas por cada algoritmo



(b) Variação da proporção de cada abelha

Figura 6.5: Problema ZDT6

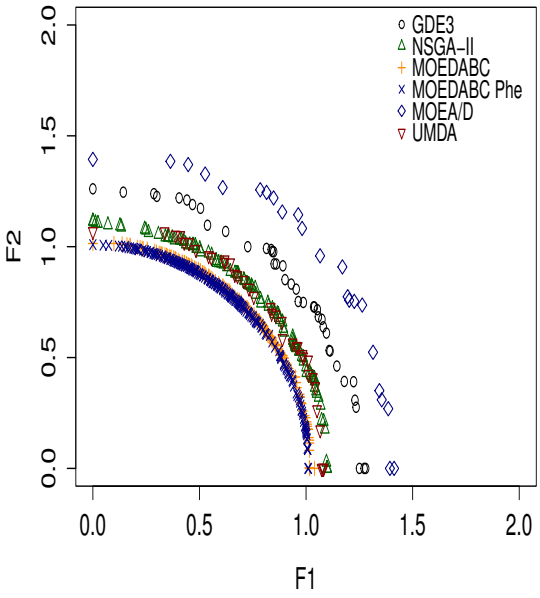
Relativo aos problemas da classe DTLZ verificou-se que tanto o MOEDABC quanto MOEDABC^{Phe} apresentaram resultados bem parecidos obtendo soluções mais próximas a P_{true} e mais espalhas que os demais métodos. Os resultados obtidos também pelo UMDA$_c^{GC}$ sugerem que os métodos utilizados pelas abelhas observadoras e campistas realmente levam a uma melhora do MOEDABC e do MOEDABC^{Phe} na resolução desses problemas.



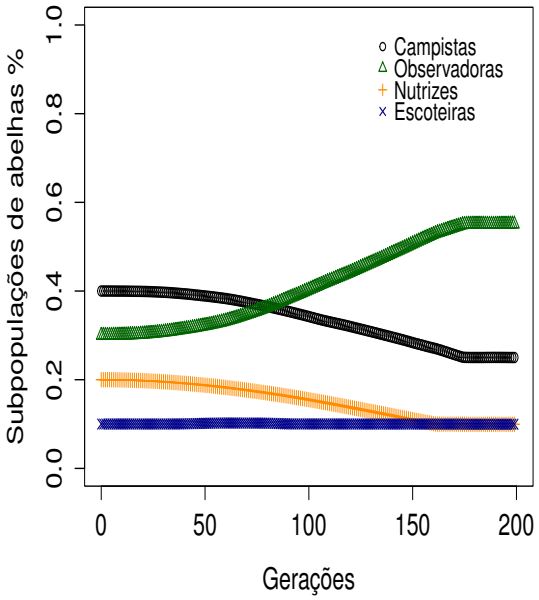
(a) Soluções obtidas por cada algoritmo

(b) Variação da proporção de cada abelha

Figura 6.6: Problema DTLZ2

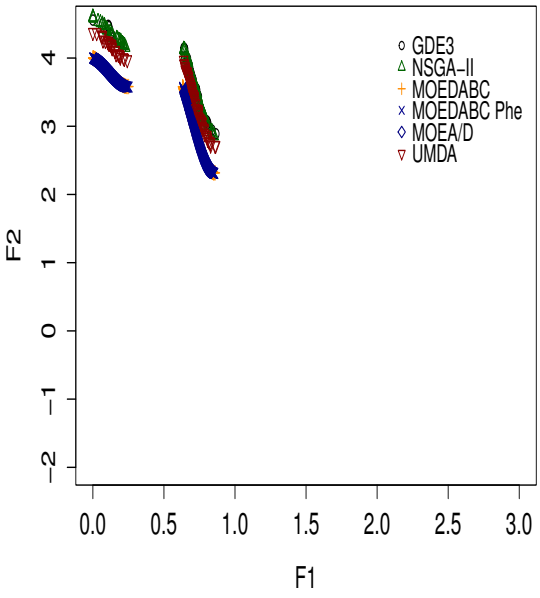


(a) Soluções obtidas por cada algoritmo

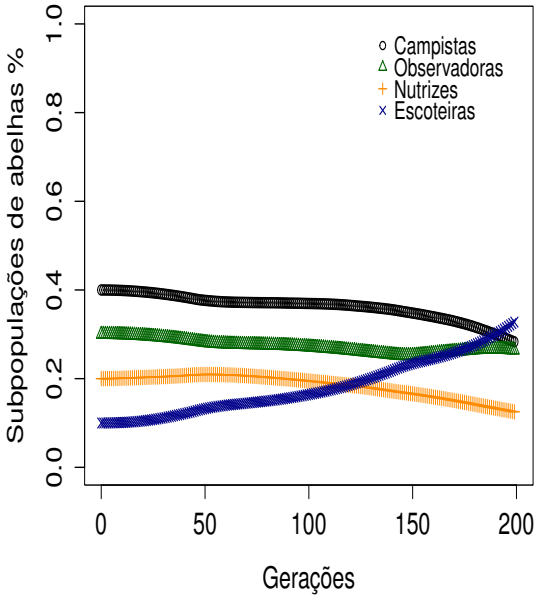


(b) Variação da proporção de cada abelha

Figura 6.7: Problema DTLZ4



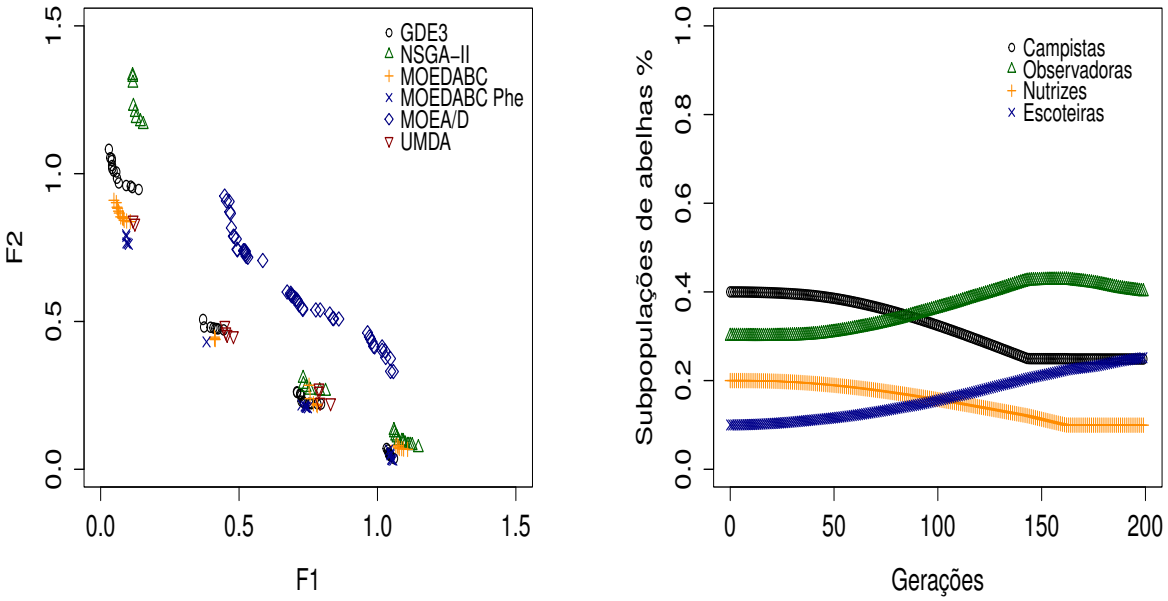
(a) Soluções obtidas por cada algoritmo



(b) Variação da proporção de cada abelha

Figura 6.8: Problema DTLZ7

Por fim, analisando os resultados obtidos nos problemas da classe UF pode-se observar que apesar do número menor de soluções os resultados obtidos pelo MOEDABC^{Phe} domina boa parte das outras soluções obtidas pelos demais métodos com relação ao problema UF1 na figura 6.9. Notou-se também mudanças na proporção de cada tipo de abelha da colônia havendo um acréscimo do número de escoteiras e uma diminuição das demais.

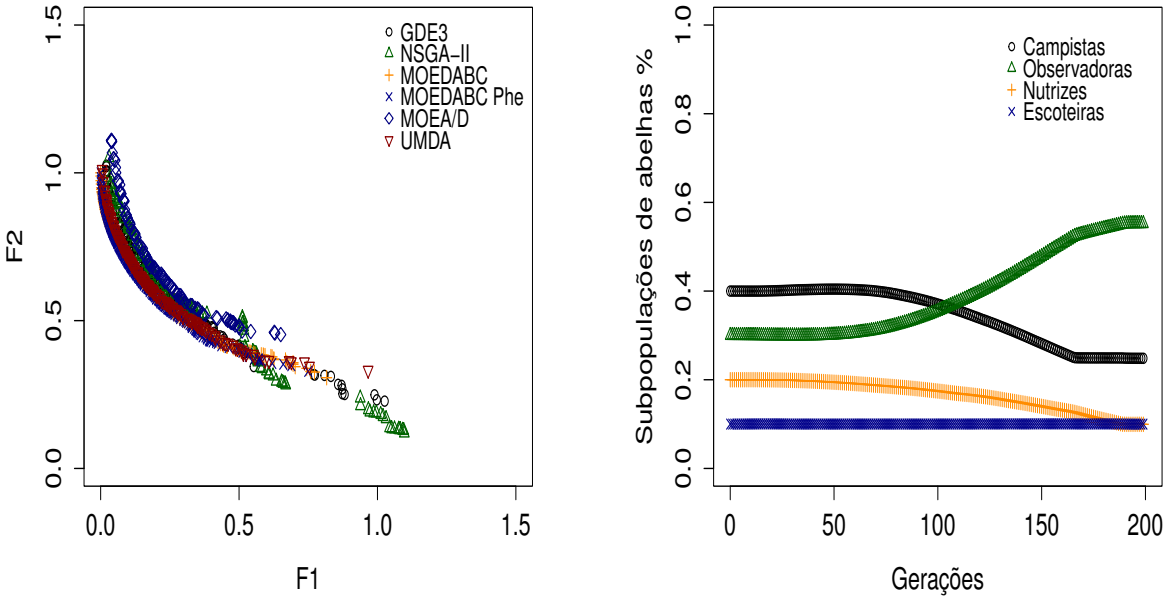


(a) Soluções obtidas por cada algoritmo

(b) Variação da proporção de cada abelha

Figura 6.9: Problema UF1

Com relação ao problema UF2 da figura 6.10 observou-se uma maior concentração das soluções obtidas pelo MOEDABC^{Phe}. Ressalta-se também um aumento da proporção de abelhas observadoras e uma diminuição das demais.

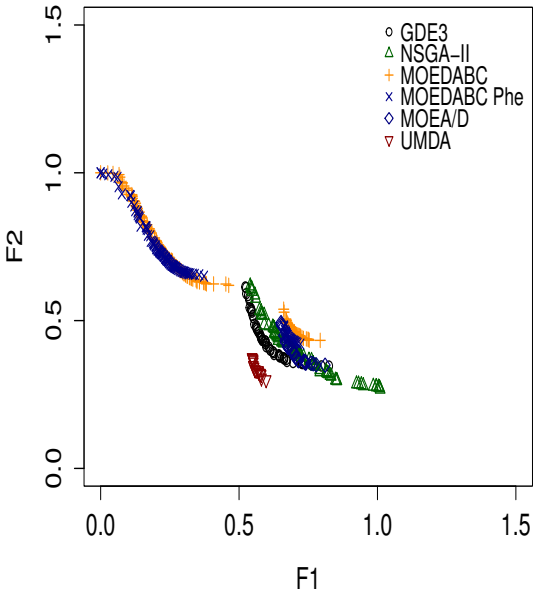


(a) Soluções obtidas por cada algoritmo

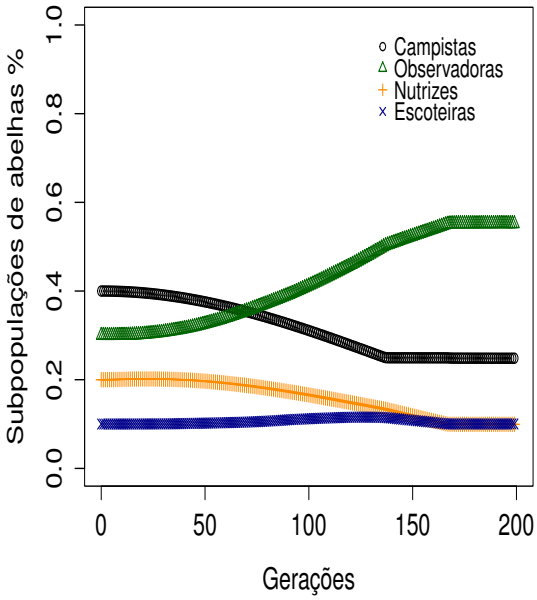
(b) Variação da proporção de cada abelha

Figura 6.10: Problema UF2

Na figura 6.11 observou-se um maior número de soluções do MOEDABC^{Phe} porém algumas destas são dominadas principalmente pelo UMDA$_c^{GC}$ na região onde $F1$ é maior que 0.5. Já na figura 6.12 as soluções estão muito aglomeradas, porém nota-se que o MOEDABC^{Phe} apresentou melhor desempenho em relação ao MOEDABC e ao UMDA$_c^{GC}$, observou-se também um grande aumento do número de escoteiras.

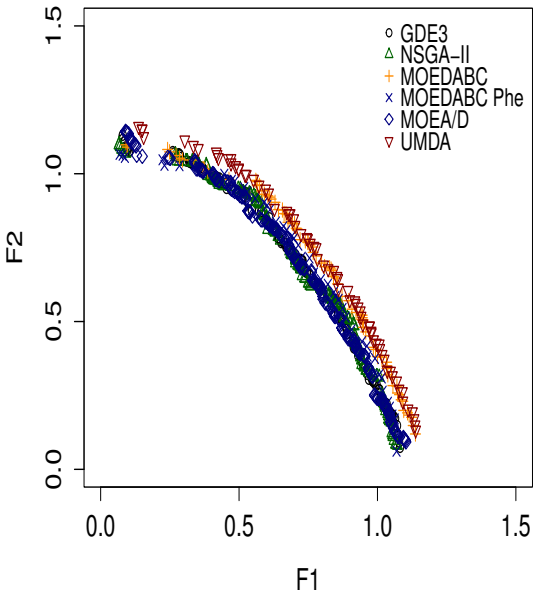


(a) Soluções obtidas por cada algoritmo

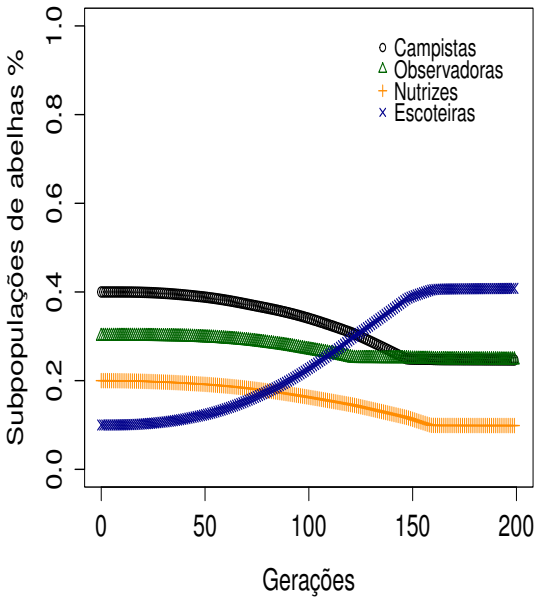


(b) Variação da proporção de cada abelha

Figura 6.11: Problema UF3



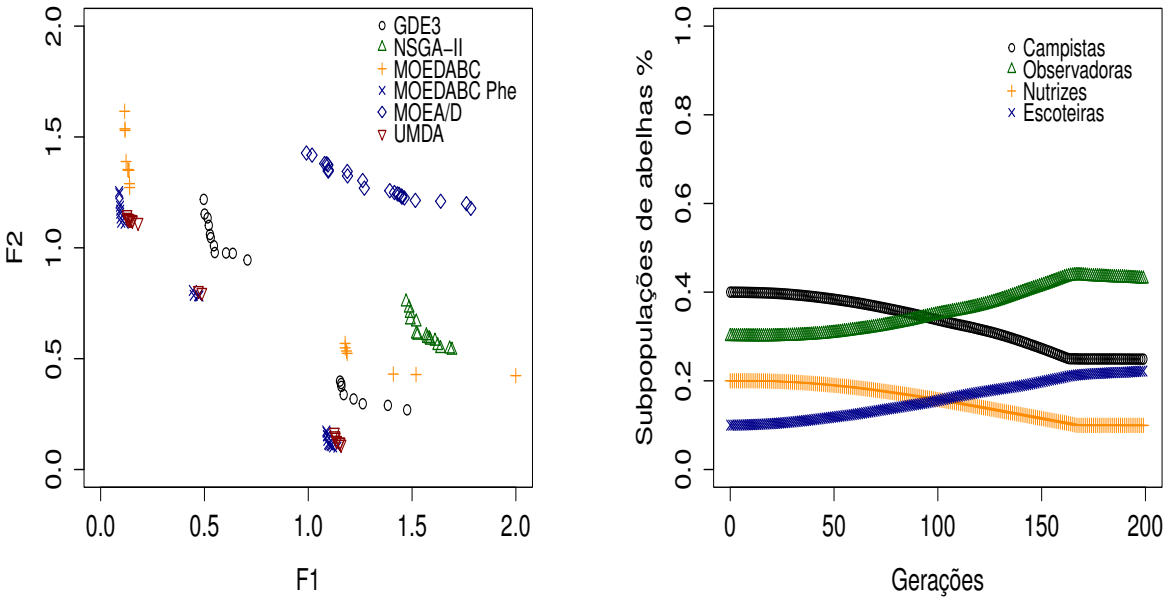
(a) Soluções obtidas por cada algoritmo



(b) Variação da proporção de cada abelha

Figura 6.12: Problema UF4

Finalmente, na figura 6.13 notou-se que o MOEDABC^{Phe} obteve um menor número de soluções, porém essas dominam a maioria das outras soluções. Neste gráfico também verificou-se que o UMDA$_c^{GC}$ obteve bons resultados. Com relação as subpopulações de abelhas do MOEDABC^{Phe} notou-se uma maior predominância de abelhas observadoras e escoteiras nas últimas gerações.



(a) Soluções obtidas por cada algoritmo

(b) Variação da proporção de cada abelha

Figura 6.13: Problema UF6

6.3 Considerações Finais

Neste capítulo foram analisados os resultados obtidos pelos algoritmos GDE3, NSGA-II, e MOEA/D, presentes no framework MOEA, além dos algoritmos MOEDABC, MOEDABC^{Phe} e UMDA$_c^{GC}$ desenvolvidos neste trabalho. Nestas análises foram utilizados os indicadores *Additive Epsilon Indicator*, *Hypervolume*, *Inverted Generational Distance*, *Maximum Pareto Front Error* e *Spacing*. Como pode-se notar o algoritmo MOEDABC^{Phe} apresentou bom desempenho, além disso pode-se observar também que a utilização do feromônio no controle da proporção de cada tipo de abelha proporcionou ao MOEDABC^{Phe} uma vantagem em relação aos demais, permitindo esse ajustar a utilização dos métodos utilizados por cada tipo de abelha conforme a natureza do problema e as soluções até então encontradas pelo algoritmo.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste trabalho um novo algoritmo híbrido denominado *Multiobjective Estimation of Distribution Algorithm based on Bee Colonies and Clusters* (MOEDABC^{Phe}) é proposto. De fato, este incorpora métodos baseados em uma colônia de abelhas artificial com um feromônio para o controle das populações, algoritmos de estimação de distribuição (EDAs), representados aqui por cada tipo de abelha da colônia (abelhas campistas, observadoras, nutrizes e escoteiras), e a clusterização do espaço objetivo. Para gerar as novas soluções o MOEDABC^{Phe} se baseia nas funções distribuições de probabilidade Gaussianas por meio dos algoritmos UMDA$_c^G$ e RECEDA, e também do operador de mutação de Cauchy para melhorar a diversidade de soluções. Por combinar EDAs, a clusterização do espaço objetivo, os métodos *Nondominated Sorting* e *Crowding-distance assignment* do NSGA-II foi possível extrair maiores informações sobre os problemas de otimização abordados, permitindo assim uma maior eficiência na solução de problemas com variáveis de decisão de larga escala. Com relação aos problemas testes verificou-se a partir de 5 indicadores de desempenho que comparados aos algoritmos do estado da arte GDE3, MOEA/D e NSGA-II as estratégias incorporadas no MOEDABC^{Phe} permitiram obter resultados bem competitivos, indicando este algoritmo como uma ferramenta útil na resolução dos problemas considerados.

Cabe ressaltar também que a adoção de algoritmos híbridos bio-inspirados em colônias artificiais de abelhas permitiu adotar certos tipos de estratégias as quais podem ser utilizadas em outros algoritmos de forma a permitir a esses maior flexibilidade na resolução de problemas complexos. Essa maior flexibilidade pôde ser observada comparando aos algoritmos MOEDABC e UMDA$_c^{GC}$, os quais foram desenvolvidos a partir de alguns métodos utilizados pelo MOEDABC^{Phe}. Neste caso pode-se verificar que a utilização de uma abordagem baseada na diferenciação de funções de cada abelha de uma colônia artificial juntamente com um feromônio permitiu alcançar melhores resultados em muitos dos problemas abordados.

7.1 Trabalhos Futuros

Nesta seção sugerimos para trabalhos futuros a utilização de métodos mais rápidos para o cálculo da média, dos desvios padrões, da matriz de covariância e demais métodos usados pelos EDAs na extração de informações das populações, visto que estes cálculos exigem grande esforço computacional. Além disso propomos a adoção dos EDAs em sistemas distribuídos os quais podem se beneficiar das informações obtidas das populações de soluções produzidas por outros EDAs de forma que a partir da troca de informações estatísticas destas populações seja possível a resolução de problemas de larga escala em menor tempo. Propomos também um estudo dos efeitos de um maior número de populações de soluções no MOEDABC^{Phe}, de um método dinâmico para definição do número de *clusters* e a comparação do MOEDABC^{Phe} com o algoritmo MO-CMA-ES (Igel et al., 2007) o qual utiliza uma estratégia evolutiva de adaptação da matriz de covariância.

Apêndice A

Publicações

Nesta seção apresenta-se o artigo que foi produzido a partir deste trabalho e publicado em congresso:

1. **Título**: A Multi-Objective Estimation of Distribution Algorithm Based on Artificial Bee Colony

   Autores: Fabiano T. Novais, Lucas S. Batista, Agnaldo J. Rocha e Frederico G. Guimarães

   Evento: 1^{st} BRICS Countries Congress (BRICS-CCI) and 11^{th} Brazilian Congress (CBIC) on Computational Intelligence.

   Data: 08 a 11 de setembro de 2013

   Local: Porto de Galinhas, Recife, Brasil

Apêndice B

Resultados Completos

Tabela B.1: Problemas da classe DTLZ: Indicador Additive Epsilon Indicator

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
DTLZ2	Média	0,17079001	0,23694714	0,05842147	0,05541717	0,31646254	0,03730651
	Desvio Padrão	0,01352085	0,02895074	0,00552306	0,00470971	0,03999618	0,00558727
	Máximo	0,20276968	0,30253770	0,07301215	0,07050690	0,37943917	0,05042845
	Minímo	0,14174081	0,17185603	0,04673785	0,04753632	0,22599901	0,02772399
	Indicador	<	<	<	-	<	>
DTLZ4	Média	0,19332278	0,23149025	0,08773977	0,05227302	0,37811449	0,14810274
	Desvio Padrão	0,02020709	0,03464209	0,03459692	0,03230816	0,19353383	0,03350167
	Máximo	0,25860568	0,30899502	0,24790670	0,20342334	1,04428713	0,30448988
	Minímo	0,15742952	0,16638835	0,03624877	0,01042457	0,19183310	0,08770081
	Indicador	<	<	<	-	<	<
DTLZ7	Média	0,26822925	0,45221889	0,04653806	0,04192632	1,50686400	0,57765233
	Desvio Padrão	0,01875463	0,02986673	0,00693133	0,00701966	0,17458449	0,03401086
	Máximo	0,31712511	0,54628962	0,06188732	0,05466295	2,00048933	0,67204500
	Minímo	0,23006334	0,38833285	0,03158673	0,02764069	1,05781755	0,50663861
	Indicador	<	<	=	-	<	<

Tabela B.2: Problemas da classe DTLZ: Indicador Hypervolume

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
DTLZ2	Média	0,25281300	0,14845605	0,44562394	0,45349793	0,09167596	0,47707676
	Desvio Padrão	0,03882101	0,04508675	0,03381205	0,03046702	0,04207755	0,03061557
	Máximo	0,34612540	0,27304037	0,52041938	0,52922901	0,20096853	0,54675729
	Minímo	0,20493721	0,08117015	0,40278708	0,41182212	0,03380760	0,44408392
	Indicador	$<$	$<$	$<$	-	$<$	$>$
DTLZ4	Média	0,24661321	0,18066883	0,44149890	0,47048745	0,09988863	0,37331145
	Desvio Padrão	0,03931362	0,05310804	0,03818094	0,03807675	0,07084024	0,04372449
	Máximo	0,34415081	0,32175919	0,52699171	0,55371620	0,24508541	0,46442796
	Minímo	0,16876084	0,09870004	0,29064364	0,35306771	0,00000000	0,26228487
	Indicador	$<$	$<$	$=$	-	$<$	$<$
DTLZ7	Média	0,12698803	0,04806630	0,33067517	0,33247288	0,00000000	0,01935849
	Desvio Padrão	0,00902517	0,01133946	0,01289914	0,01204136	0,00000000	0,00823478
	Máximo	0,15362775	0,07495666	0,36471249	0,36595697	0,00000000	0,05551299
	Minímo	0,11227062	0,01802002	0,29427415	0,31034072	0,00000000	0,01024576
	Indicador	$<$	$<$	$=$	-	$<$	$<$

Tabela B.3: Problemas da classe DTLZ: Indicador Inverted Distance

Problema	**Medida**	**GDE3**	**NSGA-II**	**MOEDABC**	**MOEDABC^{Phe}**	**MOEA/D**	**UMDA$_c^{GC}$**
DTLZ2	Média	0,00690893	0,01065518	0,00190675	0,00175469	0,01328663	0,00156659
	Desvio Padrão	0,00067730	0,00142607	0,00022722	0,00020194	0,00217758	0,00008293
	Máximo	0,00847531	0,01367840	0,00254981	0,00223210	0,01763560	0,00171490
	Minímo	0,00539620	0,00746023	0,00146917	0,00141152	0,00950267	0,00139130
	Indicador	$<$	$<$	$<$	-	$<$	$>$
DTLZ4	Média	0,01144992	0,01511209	0,00619604	0,00411335	0,02397387	0,01078501
	Desvio Padrão	0,00178191	0,00306360	0,00387063	0,00385307	0,01330851	0,00295872
	Máximo	0,01659673	0,02233356	0,02551684	0,02696492	0,07116855	0,02001197
	Minímo	0,00789580	0,00900469	0,00258126	0,00155662	0,01155224	0,00556478
	Indicador	$<$	$<$	$<$	-	$<$	$<$
DTLZ7	Média	0,00970058	0,01871048	0,00147322	0,00145690	0,06692153	0,02472814
	Desvio Padrão	0,00065070	0,00188144	0,00033722	0,00019210	0,00774285	0,00163729
	Máximo	0,01168410	0,02394671	0,00250307	0,00194297	0,08778509	0,02910487
	Minímo	0,00831904	0,01497039	0,00107101	0,00111212	0,04686434	0,02144481
	Indicador	$<$	$<$	$=$	-	$<$	$<$

Tabela B.4: Problemas da classe DTLZ: Indicador Maximum Pareto Front Error

Problema	**Medida**	**GDE3**	**NSGA-II**	**MOEDABC**	**MOEDABC^{Phe}**	**MOEA/D**	**UMDA$_c^{GC}$**
DTLZ2	Média	0,22160380	0,47719530	0,09219509	0,09095446	0,44556085	0,03048205
	Desvio Padrão	0,02177034	0,09403792	0,01684708	0,01803495	0,08076568	0,02804002
	Máximo	0,29194859	0,69367746	0,13563531	0,13456831	0,65000583	0,11829801
	Minímo	0,18771389	0,29343550	0,04648000	0,05093350	0,30561043	0,00000000
	Indicador	$<$	$<$	$=$	-	$<$	$>$
DTLZ4	Média	0,32510141	0,41511814	0,11379538	0,07497564	0,78843361	0,15819542
	Desvio Padrão	0,06085743	0,08401782	0,06884594	0,07672996	1,51008156	0,05887268
	Máximo	0,48298529	0,73137577	0,36627954	0,29903977	7,32189182	0,31205401
	Minímo	0,21783141	0,26562568	0,00000000	0,00000000	0,25502902	0,01931988
	Indicador	$<$	$<$	$<$	-	$<$	$<$
DTLZ7	Média	0,30101218	0,59288424	0,08002601	0,07518378	1,53514891	0,58682666
	Desvio Padrão	0,02536351	0,06029566	0,05248540	0,05749586	0,45679667	0,05056647
	Máximo	0,36292056	0,71666671	0,27049146	0,29140820	3,10749281	0,69905535
	Minímo	0,25538952	0,44894959	0,00000000	0,00000000	1,02701627	0,38484426
	Indicador	$<$	$<$	$=$	-	$<$	$<$

Tabela B.5: Problemas da classe DTLZ: Indicador Spacing

Problema	Medida	GDE3	NSGA-II	MOEDABC	MOEDABC^{Phe}	MOEA/D	UMDA$_c^{GC}$
DTLZ2	Média	0,04283012	0,04888421	0,03744231	0,03660193	0,07878651	0,03617616
	Desvio Padrão	0,00323944	0,00322132	0,00216900	0,00181815	0,01021221	0,00249319
	Máximo	0,05456485	0,05485854	0,04142703	0,04006093	0,10716708	0,04160699
	Minímo	0,03558562	0,04192874	0,03363698	0,03279491	0,05671940	0,03020482
	Indicador	$<$	$<$	$<$	-	$<$	$=$
DTLZ4	Média	0,05249415	0,05016292	0,05077417	0,05102663	0,19102923	0,05709188
	Desvio Padrão	0,00555535	0,00445161	0,01622307	0,01061010	0,38165996	0,01653324
	Máximo	0,06571493	0,06144075	0,08216497	0,07033600	1,87566838	0,09057774
	Minímo	0,04270149	0,04079325	0,00826093	0,00808254	0,06195859	0,02732743
	Indicador	$=$	$=$	$=$	-	$>$	$=$
DTLZ7	Média	0,04004814	0,04108498	0,04630605	0,04735361	0,15098147	0,05523589
	Desvio Padrão	0,00342432	0,00481261	0,00581173	0,00586937	0,12097661	0,01311190
	Máximo	0,04711174	0,05388428	0,06097635	0,06700324	0,79109626	0,12796511
	Minímo	0,03469269	0,03271273	0,03606650	0,03882538	0,06505490	0,04109124
	Indicador	$>$	$>$	$=$	-	$<$	$<$

Referências Bibliográficas

Ahn, Chang Wook e Ramakrishna, R. S. (2007). Multiobjective real-coded bayesian optimization algorithmrevisited: diversity preservation. *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, p. 593–600, New York, NY, USA. ACM. ISBN 978-1-59593-697-4. doi: 10.1145/1276958.1277079. URL http://doi.acm.org/10.1145/1276958.1277079.

alRifaie, Mohammad Majid; Bishop, John Mark e Caines, Suzanne. (2012). Creativity and autonomy in swarm intelligence systems. *Cognitive Computation*, v. 4, n. 3, p. 320–331.

Armañanzas, Rubé n; Inza, Iñ aki; Santana, Roberto; Saeys, Yvan; Flores, Jose Luis; Lozano, Jose Antonio; Van dePeer, Yves; Blanco, Rosa; Robles, Victor; Bielza, Concha e Larrañaga, Pedro. (2008). A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, v. 1, n. 6, p. 1–12. ISSN 1756-0381. URL http://dx.doi.org/10.1186/1756-0381-1-6.

Baluja, Shumeet e Davies, Scott. (1997). Combining multiple optimization runs with optimal dependency trees. Relatório técnico.

Bleuler, Stefan; Laumanns, Marco; Thiele, Lothar e Zitzler, Eckart. (2003). Pisa-a platform and programming language independent interface for search algorithms. *Evolutionary multi-criterion optimization*, p. 494–508. Springer, (2003).

Bonabeau, Eric e Meyer, Christopher. (2001). Swarm intelligence: A whole new way to think about business. *Harvard business review*, v. 79, n. 5, p. 106–115.

Cahon, Sébastien; Melab, Nordine e Talbi, E-G. (2004). Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, v. 10, n. 3, p. 357–380.

Coello, Carlos A Coello; Lamont, Gary B e Veldhuizen, David A Van. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

Costa, Mario e Minisci, Edmondo. (2003). Moped: A multi-objective parzen-based estimation of distribution algorithm for continuous problems. Fonseca, CarlosM.; Fleming, PeterJ.; Zitzler, Eckart; Thiele, Lothar e Deb, Kalyanmoy, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, p. 282–294. Springer Berlin Heidelberg. ISBN 978-3-540-01869-8. doi: 10.1007/3-540-36970-8_20. URL http://dx.doi.org/10.1007/3-540-36970-8_20.

De Oliveira, Ioná Maghali Santos e Schirru, Roberto. (2011). Swarm intelligence of artificial bees applied to in-core fuel management optimization. *Annals of Nuclear Energy*, v. 38, n. 5, p. 1039–1045.

Deb, Kalyanmoy; Agrawal, Samir; Pratap, Amrit e Meyarivan, Tanaka. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, v. 1917, p. 849–858.

DeLong, Elizabeth R; DeLong, David M e Clarke-Pearson, Daniel L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, p. 837–845.

Dong, Weishan e Yao, Xin. September(2007). Covariance matrix repairing in Gaussian based EDAs. *2007 IEEE Congress on Evolutionary Computation*, p. 415–422. doi: 10.1109/CEC.2007.4424501. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4424501.

Dorigo, M. e Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, p. 3 vol. (xxxvii+2348), (1999). doi: 10.1109/CEC.1999.782657.

Durillo, Juan J; Nebro, Antonio J; Luna, Francisco; Dorronsoro, Bernabé e Alba, Enrique. (2006). jmetal: a java framework for developing multi-objective optimization metaheuristics. *Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10*.

Engelbrecht, Andries P. (2006). *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons. ISBN 0470091916.

Fayad, Mohamed e Schmidt, Douglas C. (1997). Object-oriented application frameworks. *Communications of the ACM*, v. 40, n. 10, p. 32–38.

Hadka, David. *MOEA Framework A*, (2011). URL https://sourceforge.net/projects/moeaframework/files/MOEAFramework-2.0/MOEAFramework-2.0-Manual.pdf/download.

Harik, Georges e Harik, Georges. (1999). Linkage learning via probabilistic modeling in the ecga. Relatório técnico.

Hedayatzadeh, Ramin; Hasanizadeh, Bahareh; Akbari, Reza e Ziarati, Koorush. August(2010). A multi-objective Artificial Bee Colony for optimizing multi-objective problems. *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, v. 2, n. x, p. V5–277–V5–281. doi: 10.1109/ICACTE.2010.5579761. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5579761.

Hernández-Díaz, Alfredo G.; Santana-Quintero, Luis V.; Coello Coello, Carlos A.; Molina, Julián e Caballero, Rafael. August(2011). Improving the efficiency of -dominance based grids. *Inf. Sci.*, v. 181, n. 15, p. 3101–3129. ISSN 0020-0255.

doi: 10.1016/j.ins.2011.02.030. URL http://dx.doi.org/10.1016/j.ins.2011.02.030.

Hinchey, M.G.; Sterritt, R. e Rouff, C. (2007). Swarms and swarm intelligence. *Computer*, v. 40, n. 4, p. 111–113. ISSN 0018-9162. doi: 10.1109/MC.2007.144.

Igel, Christian; Hansen, Nikolaus e Roth, Stefan. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, v. 15, n. 1, p. 1–28.

Igel, Christian; Heidrich-Meisner, Verena e Glasmachers, Tobias. (2008). Shark. *The Journal of Machine Learning Research*, v. 9, p. 993–996.

Jaszkiewicz, A. (2005). Momhlib++: Multiple objective metaheuristics library in c++. *URL: http://www-idss. cs. put. poznan. pl/˜ jaszkiewicz/MOMHLib abgerufen am*, v. 22.

Kanungo, Tapas; Mount, David M; Netanyahu, Nathan S; Piatko, Christine D; Silverman, Ruth e Wu, Angela Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 24, n. 7, p. 881–892.

Karaboga, Dervis e Basturk, Bahriye. November(2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *J. of Global Optimization*, v. 39, n. 3, p. 459–471. ISSN 0925-5001. doi: 10.1007/s10898-007-9149-x. URL http://dx.doi.org/10.1007/s10898-007-9149-x.

Kennedy, J. e Eberhart, R. nov/dec(1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, p. 1942 –1948 vol.4, nov/dec(1995). doi: 10.1109/ICNN.1995.488968.

Kıran, MustafaServet; Özceylan, Eren; Gündüz, Mesut e Paksoy, Turan. (2012). Swarm intelligence approaches to estimate electricity energy demand in turkey. *Knowledge-Based Systems*, v. 36.

Knowles, Joshua e Corne, David. (2002). On metrics for comparing nondominated sets. *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, p. 711–716. IEEE, (2002).

Kukkonen, Saku e Lampinen, Jouni. (2005). Gde3: The third evolution step of generalized differential evolution. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, p. 443–450. IEEE, (2005).

Larrañaga, Pedro; Etxeberria, Ramon; Lozano, José A. e Peña, José M. (2000). Optimization in continuous domains by learning and simulation of Gaussian networks. Wu, A. S., editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, p. 201–204, Las Vegas, Nevada, USA.

Laumanns, Marco; Thiele, Lothar; Deb, Kalyanmoy e Zitzler, Eckart. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, v. 10, n. 3, p. 263–282.

Luo, Na e Qian, Feng. aug.(2009). Evolutionary algorithm using kernel density estimation model in continuous domain. *Asian Control Conference, 2009. ASCC 2009. 7th*, p. 1526 –1531, aug.(2009).

Luo, Na e Qian, Feng. (2010). Estimation of distribution algorithm sampling under gaussian and cauchy distribution in continuous domain. *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, p. 1716–1720, (2010). doi: 10.1109/ICCA.2010.5524432.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Symposium on Mathematics, Statistics and Probability*, p. 281–297. URL http://portal.acm.org/citation.cfm?doid=1068009.1068122.

Mühlenbein, Heinz e Mahnig, Thilo. (1999). Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, v. 7.

Ngatchou, Patrick; Zarei, Anahita e El-Sharkawi, MA. (2005). Pareto multi objective optimization. *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*, p. 84–91. IEEE, (2005).

Parejo, José Antonio; Ruiz-Cortés, Antonio; Lozano, Sebastián e Fernandez, Pablo. (2012). Metaheuristic optimization frameworks: a survey and benchmarking. *Soft Computing*, v. 16, n. 3, p. 527–561.

Paul, Topon Kumar e Iba, Hitoshi. (2003). Real-coded estimation of distribution algorithm. *Proceedings of The Fifth Metaheuristics International Conference*. Citeseer, (2003).

Pelikan, Martin; Hauschild, MW e Lobo, FG. (2012). Introduction to estimation of distribution algorithms. *MEDAL Report*, v. 0, n. 2012003. URL http://medal-lab.org/files/2012003.pdf.

Pelikan, Martin e Mühlenbein, Heinz. The bivariate marginal distribution algorithm, (1999).

Pelikan, Martin; Pelikan, Martin; Sastry, Kumara; Sastry, Kumara; Goldberg, David E. e Goldberg, David E. (2005). Multiobjective hboa, clustering, and scalability. Relatório técnico, In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2005.

Rao, SS. (2009). Engineering optimization: theory and practice. v. 0. URL http://books.google.com/books?hl=en&lr=&id=YNt34dvnQLEC&oi=fnd&pg=PA63&dq=Engineering+Optimization:+Theory+and+Practice&ots=FuHw0VyWUQ&sig=z0Vp_RqpVeLrpBww_igOHOGzmtg.

Robinson, Gene E. (1992). Regulation of division of labor in insect societies. *Annual review of entomology*, v. 37, n. 1, p. 637–665.

Schott, Jason R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Relatório técnico, DTIC Document.

Sebag, Michèle e Ducoulombier, Antoine. Extending population-based incremental learning to continuous search spaces, (1998).

Shim, Vui Ann; Tan, KC e Tan, KK. (2012). A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization. *Evolutionary Computation (CEC), 2012 IEEE Congress on*, p. 1–8. IEEE, (2012).

Srinivas, N. e Deb, Kalyanmoy. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, p. 221–248.

Steuer, Ralph E e Choo, Eng-Ung. (1983). An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical programming*, v. 26, n. 3, p. 326–344.

Thampi, Sabu M. (2009). Swarm intelligence. *CoRR*, v. abs/0910.4116.

Theodoridis, S. e Koutroumbas, K. (2008). *Pattern Recognition*. Elsevier Science. ISBN 9780080949123. URL http://books.google.com.br/books?id=QgD-3Tcj8DkC.

Van Veldhuizen, David A e Lamont, Gary B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. Relatório técnico, Citeseer.

Veldhuizen, David A. Van e Veldhuizen, David A. Van. (1999). Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. Relatório técnico, Evolutionary Computation.

Vira, Chankong e Haimes, Yacov Y. (1983). *Multiobjective decision making: theory and methodology*. Number 8. North-Holland.

Vo, Christopher; Panait, Liviu e Luke, Sean. (2009). Cooperative coevolution and univariate estimation of distribution algorithms. *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, p. 141–150. ACM, (2009).

Votano, JR; Parham, M e Hall, LH. (2011). Computação Evolucionária em Problemas de Engenharia. *Chemistry &*, v. 0, p. 392. URL http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract.

Waldock, Antony e Corne, David. (2010). Multi-objective probability collectives. *Applications of Evolutionary Computation*, p. 461–470. Springer.

Wang, Hui; Liu, Yong; Liu, Yong e Zeng, Sanyou. (2007). A hybrid particle swarm algorithm with cauchy mutation. *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, p. 356–360, (2007). doi: 10.1109/SIS.2007.367959.

Zhang, Qingfu e Li, Hui. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, v. 11, n. 6, p. 712–731.

Zhang, Qingfu; Sun, Jianyong e Tsang, Edward. (2007). Combinations of estimation of distribution algorithms and other techniques. *International Journal of Automation and Computing*, v. 4, n. 3, p. 273–280.

Zhang, Qingfu; Zhou, Aimin; Zhao, Shizheng; Suganthan, Ponnuthurai Nagaratnam; Liu, Wudong e Tiwari, Santosh. (2008). Multiobjective optimization test instances for the cec 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report*.

Zhou, Aimin; Qu, Bo-Yang; Li, Hui; Zhao, Shi-Zheng; Suganthan, Ponnuthurai Nagaratnam e Zhang, Qingfu. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, v. 1, n. 1, p. 32–49.

Zhou, Aimin; Zhang, Qingfu; Jin, Yaochu e Sendhoff, Bernhard. (2008). Combination of eda and de for continuous biobjective optimization. *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, p. 1447–1454. IEEE, (2008).

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

Zitzler, Eckart; Laumanns, Marco e Bleuler, Stefan. (2004). A tutorial on evolutionary multiobjective optimization. *Metaheuristics for Multiobjective Optimisation*, p. 3–37. Springer.

Zitzler, Eckart; Laumanns, Marco; Thiele, Lothar; Fonseca, Carlos M e daFonseca, Viviane Grunert. (2002). Why quality assessment of multiobjective optimizers is difficult. *GECCO*, volume 2, p. 666–673, (2002).

Zitzler, Eckart e Thiele, Lothar. (1998). Multiobjective optimization using evolutionary algorithms - a comparative case study. p. 292–301. Springer, (1998).

Zitzler, Eckart; Thiele, Lothar; Laumanns, Marco; Fonseca, Carlos M e Da Fonseca, Viviane Grunert. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *Evolutionary Computation, IEEE Transactions on*, v. 7, n. 2, p. 117–132.