
Christopher Renkavieski

*Evolução Diferencial com Parametrização Adaptativa Aplicada
a Problemas Contínuos Sem Restrições*

Joinville
2019

UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Christopher Renkavieski

**EVOLUÇÃO DIFERENCIAL COM PARAMETRIZAÇÃO
ADAPTATIVA APLICADA A PROBLEMAS CONTÍNUOS
SEM RESTRIÇÕES**

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina
como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Dr. Rafael Stubs Parpinelli
Orientador

Joinville, Junho de 2019

**EVOLUÇÃO DIFERENCIAL COM PARAMETRIZAÇÃO
ADAPTATIVA APLICADA A PROBLEMAS CONTÍNUOS
SEM RESTRIÇÕES**

Christopher Renkavieski

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação Integral do CCT/UDESC.

Banca Examinadora

Dr. Rafael Stubs Parpinelli - UDESC
(orientador)

MSc. Leanderson André - UNIVILLE

MSc. Sandro Roberto Loiola de Menezes -
TOTVS

Agradecimentos

Agradeço a meus pais, por todo o apoio e incentivo fornecidos ao longo de toda minha vida. Ao meu orientador, pela oportunidade, pelos esclarecimentos e também pelas críticas ao longo do trabalho. Aos membros do COCA, peja ajuda em diversos pontos do trabalho. Aos meus amigos, que me ajudaram e enfrentaram esta jornada ao meu lado.

“Consider again that dot. That’s here. That’s home. That’s us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every “superstar,” every “supreme leader,” every saint and sinner in the history of our species lived there - on a mote of dust suspended in a sunbeam.”

Carl Sagan

Resumo

O controle de parâmetros é um dos principais desafios ao se aplicar uma meta heurística a um problema de otimização. Devido a isso, múltiplas técnicas já foram propostas para fazer a auto adaptação destes parâmetros, de modo que eles sejam ajustados pelo próprio processo de otimização. Uma das principais meta heurísticas encontradas na literatura é a Evolução Diferencial, aplicável a problemas de domínio contínuo e que possui um grande potencial para a auto adaptação de seus parâmetros, com múltiplos algoritmos já propostos para este fim. Este trabalho se propõe a selecionar, analisar e avaliar um conjunto destes algoritmos, realizando comparações entre eles e também propondo modificações que visem aprimorar seu desempenho. Foram avaliados os algoritmos SHADE, L-SHADE e EB-L-SHADE, além do DE canônico, e foram propostos os algoritmos A-SHADE, EB-A-SHADE e EB-A-SHADE-M, baseados no L-SHADE e no EB-L-SHADE. Estes algoritmos foram avaliados por meio do conjunto de *benchmark* do CEC-2013 em 100 dimensões, e observou-se que os algoritmos adaptativos apresentaram resultados superiores em relação ao DE canônico. Observou-se também que o A-SHADE e suas variações tiveram resultados competitivos, sendo superiores ao L-SHADE nos experimentos realizados.

Palavras-chaves: Computação Evolutiva, Evolução Diferencial, Parametrização Adaptativa

Abstract

Parameter control is one of the main challenges when applying a meta heuristic to an optimization problem. Because of that, many techniques have been proposed to self adapt these parameters, so that they can be adjusted by the optimization process itself. One of the main meta heuristics found in literature is the Differential Evolution, which can be applied to continuous problems and has been shown to have a strong self adaptation potential, with many algorithms made for that end already published. This work is intended to select, analyze and evaluate a set of these algorithms, comparing them and proposing possible modifications aiming to improve their performance. The following algorithms were evaluated: SHADE, L-SHADE and EB-L-SHADE, including the canonical version of the DE, and the following algorithms were proposed: A-SHADE, EB-A-SHADE and EB-A-SHADE-M, these being based on L-SHADE and EB-L-SHADE. These algorithms were evaluated with the CEC-2013 benchmark set with 100 dimensions, and it was observed that the adaptive algorithms performed better than the canonical DE. It was also observed that A-SHADE and its variations had competitive results, being superior to L-SHADE in the experiments performed.

Keywords: Evolutionary Computing, Differential Evolution, Adaptive Parameters

Sumário

Lista de Figuras	8
Lista de Tabelas	10
Lista de Abreviaturas	11
1 Introdução	13
1.1 Objetivos	15
1.1.1 Objetivos específicos	15
1.2 Metodologia	16
1.3 Organização do Texto	16
2 Fundamentação Teórica	17
2.1 Problemas de otimização	17
2.1.1 Classificação quanto à presença de restrições e função objetivo . . .	17
2.1.2 Classificação quanto à modalidade	18
2.1.3 Classificação quanto ao número de dimensões	19
2.2 Algoritmos evolutivos para otimização	20
2.2.1 Evolução Diferencial	22
2.3 Parametrização em técnicas de otimização	24
2.4 Evolução Diferencial com parametrização adaptativa	27
2.4.1 Visão Geral das Técnicas Utilizadas	34
2.5 Algoritmos Selecionados	35
2.5.1 JADE	36
2.5.2 SHADE	38

2.5.3	L-SHADE	42
2.5.4	EB-L-SHADE	42
2.6	Considerações Parciais	44
3	Trabalhos Relacionados	45
3.1	Considerações Parciais	47
4	Metodologia	48
4.1	Algoritmos escolhidos	48
4.2	Alterações avaliadas	48
4.2.1	Detalhes e justificativas	49
4.3	Diversidade	53
4.4	Paralelismo	55
4.5	Análises estatísticas	55
4.6	Considerações Parciais	56
5	Funções para Otimização Contínua sem Restrições	57
5.1	Definição das funções	57
5.1.1	Funções simples	57
5.1.2	Funções compostas	60
5.2	Características das funções	63
5.3	Considerações Parciais	64
6	Experimentos, Resultados e Análises	65
6.1	Procolo de Experimentos	65
6.2	Resultados e Análises	66
6.3	Considerações Parciais	80
7	Considerações Finais	81

Referências	83
A Apêndice: Boxplots	88

Lista de Figuras

2.1	Funções Bohachevsky, Ackley e Himmelblau, respectivamente	19
2.2	Fluxograma básico para um algoritmo evolutivo.	21
2.3	Fluxograma para o algoritmo DE.	23
2.4	Taxonomia para o ajuste <i>offline</i> e <i>online</i> de parâmetros.	26
2.5	Número de algoritmos que utilizam cada uma das técnicas de controle citadas na seção 2.4	35
2.6	Fluxograma para o algoritmo JADE.	38
2.7	Memórias históricas M_{CR} e M_F	39
2.8	Fluxograma para o algoritmo SHADE.	41
4.1	Tamanho da população em relação ao número de gerações, para o L-SHADE em 100 dimensões.	50
4.2	Tamanho da população em relação ao número de gerações, com a redução acelerada de NP em 100 dimensões.	51
4.3	Fluxograma para o algoritmo EB-A-SHADE-M.	53
6.1	Gráficos das funções 12 e 16, respectivamente	79
6.2	Gráficos das funções 19 e 28, respectivamente	79
A.1	Boxplots dos experimentos com a função 1.	88
A.2	Boxplots dos experimentos com a função 2.	88
A.3	Boxplots dos experimentos com a função 3.	89
A.4	Boxplots dos experimentos com a função 4.	89
A.5	Boxplots dos experimentos com a função 5.	89
A.6	Boxplots dos experimentos com a função 6.	89

A.7	Boxplots dos experimentos com a função 7.	90
A.8	Boxplots dos experimentos com a função 8.	90
A.9	Boxplots dos experimentos com a função 9.	90
A.10	Boxplots dos experimentos com a função 10.	90
A.11	Boxplots dos experimentos com a função 11.	91
A.12	Boxplots dos experimentos com a função 12.	91
A.13	Boxplots dos experimentos com a função 13.	91
A.14	Boxplots dos experimentos com a função 14.	91
A.15	Boxplots dos experimentos com a função 15.	92
A.16	Boxplots dos experimentos com a função 16.	92
A.17	Boxplots dos experimentos com a função 17.	92
A.18	Boxplots dos experimentos com a função 18.	92
A.19	Boxplots dos experimentos com a função 19.	93
A.20	Boxplots dos experimentos com a função 20.	93
A.21	Boxplots dos experimentos com a função 21.	93
A.22	Boxplots dos experimentos com a função 22.	93
A.23	Boxplots dos experimentos com a função 23.	94
A.24	Boxplots dos experimentos com a função 24.	94
A.25	Boxplots dos experimentos com a função 25.	94
A.26	Boxplots dos experimentos com a função 26.	94
A.27	Boxplots dos experimentos com a função 27.	95
A.28	Boxplots dos experimentos com a função 28.	95

Lista de Tabelas

2.1	Cronologia dos diferentes algoritmos de DE adaptativos pesquisados.	28
6.1	Média e desvio padrão obtidos para os experimentos em 100 dimensões.	67
6.1	Média e desvio padrão obtidos para os experimentos em 100 dimensões.	68
6.2	Tempos totais de uma execução de cada algoritmo, em segundos, de forma sequencial e paralela	69
6.3	Gráficos e convergência e diversidade para os experimentos executados.	70
6.3	Gráficos e convergência e diversidade para os experimentos executados.	71
6.3	Gráficos e convergência e diversidade para os experimentos executados.	72
6.3	Gráficos e convergência e diversidade para os experimentos executados.	73
6.3	Gráficos e convergência e diversidade para os experimentos executados.	74
6.3	Gráficos e convergência e diversidade para os experimentos executados.	75
6.3	Gráficos e convergência e diversidade para os experimentos executados.	76

Lista de Abreviaturas

ABC	<i>Artificial Bee Colony</i>
ACO	<i>Ant Colony Optimization</i>
AdapSS-JADE	<i>Adaptive Strategy Selection – JADE</i>
ADE	<i>Adaptive Differential Evolution</i>
CA	<i>Cultural Algorithm</i>
CEC	<i>Congress on Evolutionary Computing</i>
COCA	Grupo de Computação Cognitiva Aplicada
CoDE	<i>Composite Differential Evolution</i>
COP	<i>Constrained Optimization Problem</i>
CSP	<i>Constraint-Satisfaction Problem</i>
DE	<i>Differential Evolution</i>
DEPD	<i>Differential Evolution using Pre-Calculated Differential</i>
DESAP	<i>Differential Evolution with Self-Adapting Populations</i>
EPSDE	<i>Ensemble of mutation strategies and parameters in DE</i>
ESADE	<i>Evolutionary Self-Adaptive Differential Evolution</i>
FADE	<i>Fuzzy Adaptive Differential Evolution</i>
FOP	<i>Free Optimization Problem</i>
GA	<i>Genetic Algorithm</i>
GAO	<i>Greedy Approximate Oracle method</i>
GP	<i>Genetic Programming</i>

jDElscop	<i>jDE for large-scale optimization problems</i>
L-SHADE	<i>SHADE with Linear Population Size Reduction</i>
LCG	<i>Linear Congruential Generator</i>
LSGOjDE	<i>Large Scale Global Optimization jDE</i>
MDE-pBX	<i>Modified DE with p-best crossover</i>
p-ADE	<i>pbest vector-based self-Adaptive Differential Evolution</i>
PDE	<i>Pareto Differential Evolution</i>
PSO	<i>Particle Swarm Optimization</i>
SaDE	<i>Self-adaptive Differential Evolution</i>
SaDE-MMTS	<i>SaDE with Modified Multi-Trajectory Search</i>
SDE	<i>Self-adaptive Differential Evolution</i>
SHADE	<i>Success-History based Adaptive Differential Evolution</i>
SHADE-ILS	<i>SHADE with Iterative Local Search</i>
SPDE	<i>Self-Adaptive Pareto Differential Evolution</i>
TCC	Trabalho de Conclusão de Curso

1 Introdução

A área da Computação Natural possui três vertentes: a primeira é o uso de materiais da natureza, como o DNA, para realizar computação, a segunda é o uso da computação para simular fenômenos naturais, e a terceira é o desenvolvimento de algoritmos que utilizam a natureza como inspiração (BRABAZON; O’NEILL; MCGARRAGHY, 2015). O contexto deste trabalho se encaixa na terceira destas vertentes.

A inspiração pode vir de diversas fontes na natureza, e é independente do problema que se deseja solucionar, o que permite a aplicação de algoritmos bioinspirados a diversos domínios de pesquisa (ANDRÉ, 2015). Podem ser destacados dois grandes grupos de algoritmos bioinspirados: Inteligência de Enxame e Computação Evolutiva (PARPINELLI et al., 2018).

A inteligência de enxame é caracterizada pelo comportamento coletivo de múltiplos indivíduos que interagem entre si. Entre os algoritmos de Inteligência de Enxame encontrados na literatura, podem-se destacar a Otimização por Enxame de Partículas (PSO), com inspiração no movimento coordenado de cardumes de peixes e em grupos de aves (KENNEDY; EBERHART, 1995), a técnica de Colônias Artificiais de Abelhas (ABC), baseada no comportamento de abelhas em sua busca por alimentos (KARABOGA; BASTURK, 2007) e a Otimização por Colônias de Formigas (ACO), inspirada pela construção de caminhos de formigas (DORIGO; BIRATTARI; STUTZLE, 2006).

A segunda principal família de algoritmos bioinspirados é a Computação Evolutiva, que tem como inspiração a teoria Neodarwinista, onde os seres vivos evoluem por meio de mutação, recombinação e seleção natural (KUTSCHERA; NIKLAS, 2004). Assim como na inteligência de enxame, estes algoritmos trabalham com uma população de soluções possíveis para um determinado problema e esta população evolui ao longo de um número de gerações. Nestes algoritmos, a função a ser otimizada compõe a função de *fitness*, que guia a evolução. Entre as técnicas da Computação Evolutiva, está a Evolução Diferencial (DE), tema deste trabalho, canonicamente aplicada a problemas de domínio real (STORN; PRICE, 1997). Outras técnicas evolutivas comuns são os Algoritmos Genéticos (GA) (HOLLAND, 1984), a Programação Genética (GP) (KOZA, 1992), os

Algoritmos Culturais (CA) (REYNOLDS, 1999), entre outros.

Todos os algoritmos bioinspirados possuem características em comum, como a presença de componentes probabilísticos e a necessidade de ajuste de parâmetros (PARPINELLI et al., 2018). A escolha de parâmetros, em particular, afeta diretamente a qualidade da solução encontrada (EIBEN; HINTERDING; MICHALEWICZ, 1999), isto é, o quão próxima a solução está do ponto ótimo do problema. Devido a isso, uma parametrização adequada é um dos pontos chave na execução de um algoritmo bioinspirado. Este ajuste de parâmetros pode ser realizado de duas formas: *offline*, em que os parâmetros são definidos antes da execução, e *online*, em que os parâmetros são modificados ao longo da execução (PARPINELLI et al., 2018).

No ajuste *offline*, a escolha dos parâmetros é um problema combinatorial, que se torna mais difícil quanto maior for a quantidade de parâmetros do algoritmo a ser utilizado. É necessária a realização de testes prévios para se ter referência dos valores dos parâmetros, e estes não podem ser generalizados para todos os problemas (PARPINELLI et al., 2018). Além disso, valores de parâmetros bons para uma dada etapa do processo podem não ser bons para as outras etapas. Devido a estes fatores, o desenvolvimento e aperfeiçoamento de técnicas de controle *online* de parâmetros é de grande interesse para a área de otimização.

A Evolução diferencial (DE) é uma das técnicas de Computação Evolutiva. Este algoritmo foi elaborado tendo como objetivo a otimização de problemas contínuos, que são problemas cujo domínio pode assumir qualquer valor real dentro de um intervalo. No DE, os indivíduos são representados como vetores de valores reais que representam possíveis soluções para um problema e passam por operações de mutação, *crossover* (recombinação) e seleção. O DE possui quatro principais parâmetros que devem ser ajustados: o tamanho da população NP , o fator de mutação F , a taxa de *crossover* CR e a estratégia de mutação utilizada, sendo que esta última não é somente um valor, mas sim o algoritmo que será utilizado para aplicar a mutação aos indivíduos (STORN; PRICE, 1997). A escolha do DE neste trabalho se dá por duas principais razões: ele é adequado a problemas de otimização contínua, e possui um grande potencial para auto-adaptação de seus parâmetros, já apresentando múltiplas variações na literatura que realizam esta adaptação. Entre os algoritmos de DE adaptativos, podem se destacar: SaDE (QIN; HUANG; SUGANTHAN, 2009), jDE (BREST et al., 2006), JADE (ZHANG; SANDERSON, 2009), SHADE (TANABE; FUKUNAGA, 2013) e L-SHADE (TANABE; FUKUNAGA,

2014).

Neste trabalho, os algoritmos SHADE e L-SHADE servirão de base para as análises e modificações propostas. Esta escolha vêm do fato de o SHADE apresentar grande potencial, que fica evidente por possuir múltiplos algoritmos elaborados com base nele. Um destes é o L-SHADE, que também foi escolhido devido aos resultados competitivos que ele apresenta, possuindo status de estado da arte.

A contribuição deste trabalho está na avaliação da eficácia e da qualidade da solução obtida pela aplicação das técnicas de Evolução Diferencial auto-adaptativa a diferentes problemas de otimização, além da análise do comportamento destas técnicas e da avaliação de possíveis pontos em que elas podem ser aprimoradas.

1.1 Objetivos

Este trabalho tem como objetivo geral a aplicação e análise de técnicas de Evolução Diferencial com parametrização adaptativa a diferentes problemas de otimização contínua sem restrições.

1.1.1 Objetivos específicos

- Estudar a técnica de Evolução Diferencial e seu contexto na área da Computação Evolutiva;
- Fazer um levantamento dos diferentes algoritmos para a parametrização adaptativa aplicados à Evolução Diferencial;
- Implementar diferentes algoritmos de Evolução Diferencial com parametrização adaptativa;
- Selecionar os algoritmos que servirão de base para as modificações, no caso o SHADE e o L-SHADE;
- Propor modificações nos algoritmos com o intuito de viabilizar melhorias;
- Realizar experimentos e comparar a eficácia dos algoritmos selecionados e suas modificações.

1.2 Metodologia

Inicialmente será realizada uma revisão bibliográfica sobre a técnica de Evolução Diferencial, e seu contexto dentro das áreas de Computação Evolutiva e Computação Natural. Serão também estudadas as diferentes técnicas de parametrização adaptativa aplicadas à Evolução Diferencial e as comparações já realizadas entre eles e disponíveis na literatura.

Na sequência do trabalho, a versão canônica da Evolução Diferencial e um conjunto dos algoritmos adaptativos estudados serão implementados e aplicados a problemas contínuos sem restrições. Os algoritmos implementados serão aplicados a funções de *benchmark* da literatura, e os resultados obtidos serão avaliados e comparados, permitindo identificar aqueles que apresentarem os melhores resultados.

Além da comparação dos resultados, será também realizada a análise do comportamento dos algoritmos, com o objetivo de identificar pontos em que eles podem ser aperfeiçoados. Quando possível, serão elaboradas modificações nos algoritmos, com base nas análises feitas, e estes algoritmos modificados serão também avaliados.

1.3 Organização do Texto

Este trabalho está organizado em sete capítulos. No capítulo 1, o capítulo presente, os temas do trabalho são introduzidos, são especificados seus objetivos gerais e específicos, e a metodologia seguida é explicada. O capítulo 2 apresenta toda a revisão teórica necessária para este trabalho. No capítulo 3, são apresentados outros trabalhos da literatura que estão relacionados a este. O capítulo 4 explica as metodologias dos experimentos realizados. O capítulo 5 apresenta as funções de *benchmark* utilizadas para avaliar os algoritmos. No capítulo 6, são apresentados os experimentos realizados, seus protocolos, resultados obtidos e análise destes resultados. O capítulo 7 apresenta as considerações obtidas neste trabalho e apresenta os próximos passos a ser seguidos.

2 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica referente ao escopo do desenvolvimento deste trabalho. Ele está organizado da seguinte forma: a seção 2.1 apresenta o conceito de problemas de otimização e como eles podem ser classificados. A seção 2.2 introduz, de modo geral, os algoritmos evolutivos, para em seguida explicar em mais detalhes o algoritmo de Evolução Diferencial (DE). Na seção 2.3 é explicada a importância da parametrização em técnicas de otimização, diferenciando parametrização *offline* e *online* e apresentando uma taxonomia para as diferentes categorias de parametrização encontradas na literatura. Já na seção 2.4, são apresentadas diferentes formas pelas quais a adaptação *online* de parâmetros já foi aplicada ao algoritmo DE na literatura. Dentre os algoritmos apresentados na seção 2.4, os escolhidos para implementação são apresentados em detalhes na seção 2.5. Finalizando o capítulo, a seção 2.6 apresenta as considerações gerais da seção.

2.1 Problemas de otimização

Os problemas de otimização encontrados na literatura, sejam eles do mundo real ou *benchmarks*, podem ser classificados de diferentes formas, sendo as principais: presença ou não de restrições e função objetivo, modalidade, número de dimensões e quantidade de objetivos.

2.1.1 Classificação quanto à presença de restrições e função objetivo

A primeira forma de se classificar um problema de otimização é quanto à presença ou não de restrições e função objetivo, havendo assim quatro possíveis categorias dentro desta classificação (EIBEN; RUTTKAT, 1997). Destas quatro, o caso em que não há função objetivo nem restrições é chamado de “Espaço de busca livre”, e não possui relevância em problemas de otimização. Sendo assim, as três categorias com relevância para este estudo, de acordo com Eiben e Ruttkat (1997), são descritas da seguinte forma:

- ***Free Optimization Problem (FOP)***: Um FOP é caracterizado por um par $\langle S, f \rangle$, onde S é um espaço de busca sem restrições, e f é o valor de uma função objetivo que deve ser maximizada ou minimizada em S . Uma solução para um FOP é um valor $s \in S$ que possua um valor f ótimo. Encontrar o ponto mínimo de uma função contínua é um exemplo de FOP.
- ***Constraint-Satisfaction Problem (CSP)***: Um CSP é um par $\langle S, \phi \rangle$, onde S é um espaço de busca livre, e ϕ é uma função booleana em S . Uma solução para um CSP é um valor $s \in S$ tal que $\phi(s) = \text{true}$. Um exemplo de problema nesta categoria é o problema da satisfazibilidade booleana.
- ***Constrained Optimization Problem (COP)***: Um COP é caracterizado por uma tripla $\langle S, f, \phi \rangle$, onde S é um espaço de busca livre, f é o valor de uma função objetivo que deve ser otimizada em S , e ϕ é uma função booleana em S . Uma solução para um COP é um valor $s \in S$, tal que $\phi(s) = \text{true}$ e f apresente um valor ótimo. Maximizar lucro enquanto se minimiza custos de produção, com restrição nos recursos disponíveis, é um exemplo de COP.

Neste trabalho, as técnicas de otimização utilizadas serão aplicadas a problemas do tipo FOP, onde o objetivo será somente encontrar o ponto ótimo de uma dada função.

2.1.2 Classificação quanto à modalidade

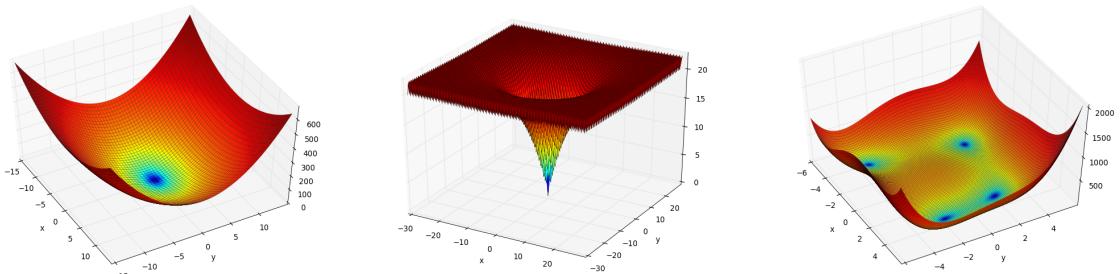
Se um dado ponto no espaço de busca de uma função apresentar um valor objetivo maior ou menor que todos os pontos na sua vizinhança, diz-se que este ponto é um máximo ou mínimo local. Se este ponto apresentar também o maior ou menor valor para a função objetivo em todo o espaço de busca, ele é um máximo ou mínimo global. Estes pontos podem também ser chamados de ótimos locais e globais, respectivamente, dependendo se o objetivo do problema for encontrar o ponto máximo ou mínimo da função.

A modalidade de uma função diz respeito à quantidade de ótimos locais que esta função possui. Diz-se que é unimodal uma função que apresenta somente um ótimo local, que neste caso será também o ótimo global, e diz que é multimodal uma função que apresentar múltiplos ótimos locais (JAMIL; YANG, 2013).

Em uma função multimodal, é possível que vários pontos ótimos apresentem o mesmo valor, sendo assim todos ótimos globais. É também possível que somente um destes ótimos seja global, com os outros ótimos locais apresentando valores piores que este segundo o objetivo do problema (minimizar ou maximizar). E pode-se também ter funções que apresentem múltiplos ótimos globais e múltiplos ótimos locais piores que estes.

Três exemplos de funções com diferentes modalidades são apresentados na figura 2.1. Esta figura mostra, respectivamente, os gráficos das funções Bohachevsky (unimodal), Ackley (multimodal com um ótimo global) e Himmelblau (multimodal com quatro ótimos globais).

Figura 2.1: Funções Bohachevsky, Ackley e Himmelblau, respectivamente



Fonte: (DEAP Project, 2014).

Existem dois tipos de objetivos quando se trabalha com funções multimodais: otimização global e otimização multimodal. Na otimização global, o objetivo é encontrar o ótimo global do problema, ou um dos ótimos globais caso hajam múltiplos. Já na otimização multimodal, o objetivo é encontrar todos os ótimos locais do problema.

Neste trabalho, as técnicas de otimização serão aplicadas tanto a funções unimodais quanto multimodais, tendo a otimização global como objetivo.

2.1.3 Classificação quanto ao número de dimensões

O número de dimensões de uma função, também chamada de dimensionalidade, se refere a quantidade de variáveis que esta função deve receber de entrada para que o seu retorno possa ser computado. Uma função $f(x)$ é unidimensional, pois possui apenas uma variável ou dimensão. Já uma função $f(x, y)$ possui duas dimensões.

Diz-se que uma função é escalável se o seu número de dimensões for variável, isto é, se ela for computável com diferentes quantidades de variáveis de entrada. A função

$f(x) = x^2$, por exemplo, não é escalável, enquanto a função $f(x_0 \dots x_n) = \sum_{i=0}^n x_i^2$ é escalável.

De modo geral, um problema se torna mais difícil quanto maior for a sua dimensionalidade, pois o seu espaço de busca aumenta exponencialmente com o seu número de dimensões (JAMIL; YANG, 2013).

2.2 Algoritmos evolutivos para otimização

A teoria Neodarwinista explica como espécies de indivíduos se modificam com o passar do tempo por meio da seleção natural (KUTSCHERA; NIKLAS, 2004). A Computação Evolutiva consiste em uma família de algoritmos que se utiliza desta teoria como inspiração (BRABAZON; O'NEILL; MCGARRAGHY, 2015).

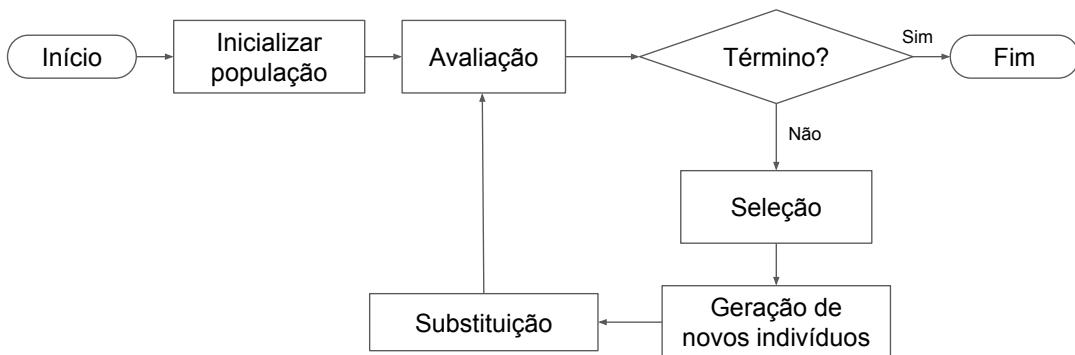
Embora existam múltiplos algoritmos evolutivos, há um conjunto de características que todos compartilham. São elas: indivíduos, população, função *fitness*, operadores evolutivos e seleção (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

- **Indivíduos:** um indivíduo representa uma possível solução ao problema. Por exemplo, caso se deseje otimizar uma função $f(x_1 \dots x_n)$, uma forma de se representar um indivíduo é com um vetor $(x_1 \dots x_n)$. Da genética, diz-se que cada x_i é um gene, e o vetor de genes é um cromossomo, e o indivíduo pode ser composto por um ou mais cromossomos.
- **População:** assim como na biologia, uma população é um conjunto de indivíduos que interagem e trocam informações entre si.
- **Função *fitness*:** o *fitness* de um indivíduo é a medida do quanto adaptado ele está ao problema sendo otimizado. Uma forma comum de se medir o *fitness* é aplicando a função objetivo aos genes do indivíduo. Dependendo do problema e da forma como a operação de seleção é implementada, um bom indivíduo pode ser caracterizado por um valor de *fitness* maior ou menor que o valor de um indivíduo ruim.
- **Operadores evolutivos:** são os operadores que modificam os indivíduos de uma dada geração, dando origem aos indivíduos que poderão fazer parte da próxima geração. Entre os operadores evolutivos, os mais comuns são a mutação e o *crossover*:

- **Mutação:** produz modificações aleatórias nos genes de um indivíduo, analógicamente ao conceito de mutação genética da biologia. Esta operação tem por característica aumentar a diversidade entre os indivíduos de uma população.
- **Crossover:** também chamada de “recombinação”, é a operação responsável por mesclar aleatoriamente os genes de dois indivíduos, produzindo indivíduos novos. Por combinar informações de indivíduos já presentes na população, esta operação tende a diminuir a diversidade entre eles.
- **Seleção:** é a operação responsável por selecionar, em uma população, os indivíduos que irão gerar descendentes para a próxima geração. Esta seleção é realizada com base no *fitness*, sendo que indivíduos com um valor de *fitness* melhor têm maior probabilidade de ser selecionados.

Um fluxograma básico para as operações de um algoritmo evolutivo é apresentado na figura 2.2. As etapas do fluxograma são: uma população inicial é gerada de forma aleatória e em seguida o *fitness* dos seus indivíduos é avaliado. Os indivíduos são então selecionados com base em seu *fitness* e novos indivíduos são gerados a partir destes por meio dos operadores evolutivos, como mutação e crossover. Estes novos indivíduos vão então substituir indivíduos da geração atual, dando origem à próxima geração, que é também avaliada. Neste ponto, se o critério de parada não for satisfeito, retorna-se à etapa de seleção. Um critério de parada que pode ser utilizado é o número de gerações, por exemplo.

Figura 2.2: Fluxograma básico para um algoritmo evolutivo.



Fonte: adaptado de (BRABAZON; O’NEILL; MCGARRAGHY, 2015).

Os detalhes de cada etapa deste fluxograma, assim como a sua ordem exata, são específicos para cada algoritmo evolutivo, de modo que a figura 2.2 representa apenas

uma visão geral desta família de algoritmos. Na Evolução Diferencial (DE), por exemplo, a etapa de seleção ocorre após a criação dos novos indivíduos, sendo ela o fator de decisão para a etapa de substituição (STORN; PRICE, 1997).

Outro conceito importante da computação evolutiva é a diversidade genética ou dispersão da população. A diversidade é uma medida do quanto espalhados estão os indivíduos no espaço de busca, de modo que uma baixa diversidade indica que a população está convergindo ou convergida, enquanto uma alta diversidade indica uma população espalhada pelo espaço (BRABAZON; O’NEILL; MCGARRAGHY, 2015). Baixa diversidade é considerada um dos principais responsáveis por convergência prematura da população a um ótimo local, o que mostra a importância de manter a diversidade da população, principalmente nos estágios iniciais da evolução (GUPTA; GHAFIR, 2012). Uma diversidade alta permite a melhor exploração do espaço de busca, por caracterizar uma população mais espalhada por ele. Uma diversidade baixa, por outro lado, indica a intensificação da solução, com a população realizando a busca em uma pequena região do espaço de busca.

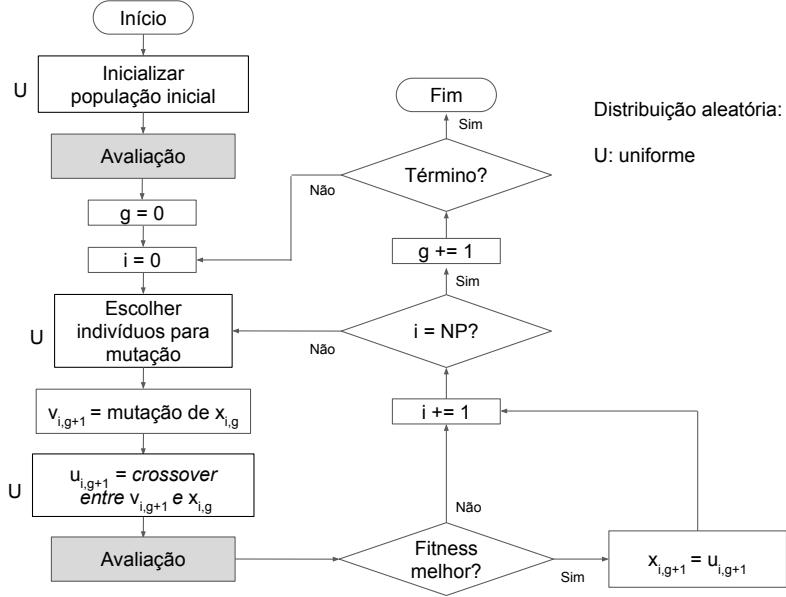
Existem múltiplos algoritmos evolutivos, podendo-se destacar os Algoritmos Genéticos (GA), a Evolução Diferencial (DE), a Programação Genética (GP) e os Algoritmos Culturais (CA). O foco deste trabalho é o DE, que é explicado em detalhes na seção 2.2.1.

2.2.1 Evolução Diferencial

A Evolução Diferencial (*Differential Evolution – DE*) é uma técnica de otimização pertencente à família dos algoritmos evolutivos, proposta por Storn e Price (1997). A DE foi desenvolvida primariamente para a solução de problemas de otimização contínua, podendo ser aplicada mesmo se a função a ser otimizada for não-diferenciável, não-linear e multimodal (STORN; PRICE, 1997).

O algoritmo DE utiliza uma população de NP indivíduos e as operações de mutação, *crossover* e seleção, nesta ordem. A figura 2.3 apresenta o fluxograma da Evolução Diferencial. Nele, as operações destacadas com a letra “U” representam passos onde é utilizado um algoritmo para geração de números aleatórios com distribuição uniforme. As caixas com fundo em destaque indicam as etapas do algoritmo em que foi aplicado paralelismo, como discutido no capítulo 4.

Figura 2.3: Fluxograma para o algoritmo DE.



Fonte: elaborado a partir de (STORN; PRICE, 1997).

Após a criação e avaliação da população inicial, o *loop* evolutivo é iniciado. Para cada indivíduo $x_{i,g}$ da população, onde g é a geração atual, são aplicados os processos de mutação e *crossover* para gerar o indivíduo de teste $u_{i,g+1}$. Este indivíduo é então avaliado e seu *fitness* é comparado ao do seu pai $x_{i,g}$. Caso haja melhora no *fitness*, $u_{i,g+1}$ substituirá $x_{i,g}$ na geração $g + 1$. As etapas mutação, *crossover* e seleção são explicadas em mais detalhes nas seções seguintes.

Mutação

Para cada indivíduo $x_{i,g}$, sendo g a geração atual e $i \in \{1 \dots NP\}$, um vetor doador $v_{i,g+1}$ é gerado. Existem múltiplas estratégias de mutação que podem ser utilizadas para gerar este vetor doador, sendo DE/rand/1/bin a estratégia originalmente proposta por Storn e Price (1997), descrita como:

$$v_{i,g+1} = x_{r1,g} + F \times (x_{r2,g} - x_{r3,g}) \quad (2.1)$$

onde $r_1, r_2, r_3 \in \{1 \dots NP\}$ são selecionados aleatoriamente com distribuição uniforme, mutualmente diferentes e diferentes de i . Além disso, $F \in [0, 2]$ é um valor real e constante ao longo do processo de evolução (STORN; PRICE, 1997). O valor de F determina o quão distante o indivíduo doador estará do seu pai. Um F alto produzirá indivíduos mais

distantes, favorecendo a exploração, enquanto um F baixo produzirá indivíduos mais próximos, favorecendo a intensificação.

Crossover

A operação de *crossover* é realizada entre o indivíduo pai $x_{i,g}$ e o indivíduo doador $v_{i,g+1}$, dando origem ao indivíduo de teste $u_{i,g+1}$. O indivíduo de teste é formado por uma mescla entre os genes dos indivíduos envolvidos da seguinte forma:

$$u_{ji,g+1} = \begin{cases} v_{ji,g+1} & \text{se } rand(j) \leq CR \text{ ou } j = rand(i) \\ x_{ji,g} & \text{se } rand(j) > CR \text{ e } j \neq rand(i) \end{cases} \quad (2.2)$$

onde $j \in \{1..D\}$, sendo D o número de dimensões do indivíduo, e $CR \in [0, 1]$ a constante de *crossover*. Tem-se também $rand(j) \in [0, 1]$, um valor real aleatório de distribuição uniforme, calculado para cada valor de j , e $rand(i) \in \{1..D\}$, um valor inteiro aleatório também de distribuição uniforme, que tem por objetivo garantir que o indivíduo de teste $u_{i,g+1}$ receberá ao menos um gene de $u_{i,g+1}$ (STORN; PRICE, 1997). Quanto maior o valor de CR , mais genes do indivíduo doador estarão presentes no indivíduo de teste, de modo que um CR próximo a 1 favorece a exploração do espaço de busca.

Seleção

A DE canônica utiliza uma forma de seleção gulosa, em que o *fitness* do indivíduo de teste $u_{i,g+1}$ é comparado ao *fitness* do indivíduo pai $x_{i,g}$. Caso o indivíduo de teste apresente um *fitness* melhor, este substituirá o indivíduo pai na geração $g + 1$ (STORN; PRICE, 1997).

2.3 Parametrização em técnicas de otimização

Quando se pretende aplicar uma heurística de otimização a um problema, seja qual for esta heurística, um dos maiores problemas encontrados é a definição dos parâmetros do algoritmo.

A quantidade de parâmetros a ser ajustada varia de acordo com o algoritmo. O algoritmo de Colônias de Abelha Artificiais (ABC), por exemplo, possui dois parâmetros,

enquanto o algoritmo de Otimização por Enxame de Partículas (PSO) apresenta quatro (ANDRÉ; PARPINELLI, 2014).

O ajuste dos parâmetros influencia a direção da busca por regiões do espaço com boas soluções, afetando assim diretamente a qualidade da solução encontrada pelo algoritmo (ANDRÉ; PARPINELLI, 2014). Paralelamente ao espaço de busca do problema sendo otimizado, existe o espaço de busca dos parâmetros do algoritmo, que é também um problema de otimização (ANDRÉ; PARPINELLI, 2014).

Eiben, Hinterding e Michalewicz (1999) determinam duas principais formas de se escolher os parâmetros para uma heurística de otimização: *parameter tuning* e *parameter control*, também chamadas de controle *offline* e *online*, respectivamente (PARPINELLI et al., 2018).

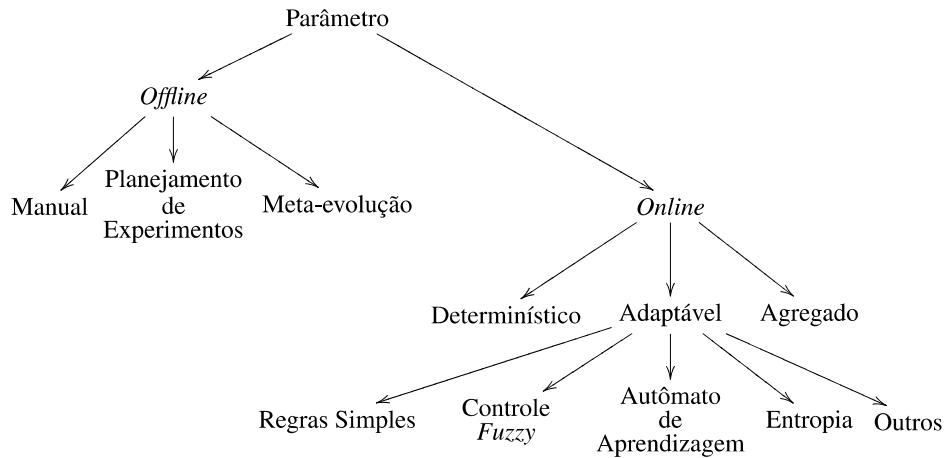
O controle *offline* é caracterizado pela escolha manual de parâmetros anterior à execução do algoritmo, sendo que estes parâmetros se mantêm fixos ao longo da execução. Eiben, Hinterding e Michalewicz (1999) argumentam que qualquer tentativa de se encontrar parâmetros ótimos desta forma está fracassada *a priori*, destacando algumas razões para este raciocínio:

- Os diferentes parâmetros não são independentes entre si, de modo que não se pode otimizar um por vez, e tentar todas as combinações possíveis é inviável;
- O processo de se otimizar os parâmetros consome muito tempo, independente das interações que existem entre eles;
- Múltiplas heurísticas de otimização, como as evolutivas, são intrinsecamente dinâmicas, de modo que os valores adequados para os parâmetros podem variar ao longo da execução. Por exemplo, em um GA, pode-se desejar probabilidades maiores de mutação no início da evolução, para que o espaço de busca seja melhor explorado, enquanto no final do processo, uma probabilidade menor pode ser desejada, para refinar a solução encontrada com uma busca local.

Os problemas encontrados no ajuste *offline* de parâmetros podem ser solucionados por meio do ajuste *online*, em que o valor dos parâmetros pode variar ao longo da busca. Existem diferentes formas de se realizar o controle *online*, com Eiben, Hinterding e Michalewicz (1999) e Zhang et al. (2012) definindo diferentes taxonomias para esta

parametrização. André e Parpinelli (2014) usam como base estas duas taxonomias para propor a que é apresentada na figura 2.4.

Figura 2.4: Taxonomia para o ajuste *offline* e *online* de parâmetros.



Fonte: (ANDRÉ; PARPINELLI, 2014).

Com base nesta taxonomia, as formas de ajuste possíveis para o controle *offline* são:

- **Manual:** os parâmetros são determinados pelo usuário, antes da execução do algoritmo, com base na sua experiência pessoal.
- **Planejamento de Experimentos:** são realizados experimentos prévios, com diferentes configurações de parâmetros, e uma combinação é escolhida por meio da análise dos resultados desses experimentos.
- **Meta-evolução:** um algoritmo é aplicado previamente para evoluir os valores dos parâmetros.

No controle *online*, por sua vez, têm-se os ajustes:

- **Determinístico:** são utilizadas regras determinísticas para alterar os valores dos parâmetros durante a execução, sem usar informações do processo. Frequentemente é utilizada alguma forma de escalonamento, de modo que os valores variem com o tempo, seguindo alguma função. Dentre os algoritmos explicados na seção seguinte, o L-SHADE utiliza este tipo de ajuste para o tamanho da sua população.
- **Adaptável:** informações do processo de busca são utilizadas para ajustar a variação dos parâmetros. Isto pode ser feito de múltiplas formas, sendo algumas delas:

- **Regras Simples:** a adaptação dos parâmetros é realizada por meio de regras simples, implementadas por funções que recebem como entrada algum *feedback* da evolução. Estas funções podem ter diferentes formas, e o *feedback* pode ser o *fitness* da população, a sua diversidade, entre outros. Este tipo de controle é o mais comum entre os algoritmos explicados na próxima seção, sendo usado, por exemplo, para adaptar os valores de F e CR no JADE, SHADE e L-SHADE.
- **Controle Fuzzy:** os valores dos parâmetros são tratados com diferentes graus de pertinência, e é utilizado um mecanismo de inferência para avaliar o estado atual do sistema. Estes são conceitos da lógica *fuzzy*, que difere da lógica tradicional por utilizar graus de incerteza. Este é o tipo de controle utilizado pelo FADE, por exemplo.
- **Autômato de Aprendizagem:** é utilizado um mecanismo para tomada de decisão que pode operar em ambientes estocásticos e aprimora sua eficácia por meio de um processo de aprendizagem. Tais mecanismos são chamados “Autômatos de aprendizagem”, e, neste contexto, podem ser utilizados para selecionar valores para os parâmetros. O AdapSS-JADE utiliza este tipo de controle para a seleção da estratégia de mutação.
- **Entropia:** o conceito físico de entropia é utilizado para analisar o grau de estocasticidade da população de soluções, fornecendo informações que podem ser utilizadas para ajustar os parâmetros do algoritmo.
- **Agregado:** os parâmetros são codificados juntamente da solução, e são otimizados pelo próprio processo de evolução, por meio das rotinas do algoritmo. SDE e jDE são algoritmos citados na próxima seção que utilizam controle agregado.

Esta taxonomia será utilizada neste trabalho para classificar os diferentes algoritmos de DE adaptativa estudados.

2.4 Evolução Diferencial com parametrização adaptativa

A literatura apresenta múltiplas variações com parametrização *online* do algoritmo DE. A tabela 2.1 apresenta uma cronologia dos algoritmos desta classe pesquisados pelo autor,

e cada um dos algoritmos é explicado brevemente na sequência.

Tabela 2.1: Cronologia dos diferentes algoritmos de DE adaptativos pesquisados.

Ano	Nome	Params. Adapt.	Características	Referência
2002	SPDE	F, CR	Pareto DE F e CR herdados dos pais	Abbass (2002)
2004	DEPD DE_n	F	População extra Diferenciais prévias	Ali e Törn (2004)
2005	FADE	F, CR	Lógica Fuzzy	Liu e Lampinen (2005)
2005	SaDE	F, CR Mut. estrat.	Dist. normal para F e CR Adapta probs. para mut. estrat.	Qin e Suganthan (2005)
2005	SDE	F, CR	F herdado dos pais Dist. normal para CR	Omran, Salman e Engelbrecht (2005)
2006	DESAP	F, CR, NP	NP em função de um gene	Teo (2006)
2006	jDE	F, CR	F e CR podem ser herdados F_l, F_u, τ_1, τ_2	Brest et al. (2006)
2008	dynNP-DE	F, CR, NP	jDE com NP dinâmico.	Brest e Maučec (2008)
2009	ESADE	F, CR	F evoluído em pop. extra Nova mutação	Epitropakis, Plagianakos e Vrahatis (2009)
2009	JADE	F, CR	Memória opcional	Zhang e Sanderson (2009)
2011	AdapSS-JADE	Mut. estrat.	JADE com seleção de estratégias de mutação	Gong et al. (2011)
2011	EPSDE	F, CR Mut. estrat.	Param. herdados com base em sucesso	Mallipeddi et al. (2011)
2011	jDElscop	F, CR, NP Mut. estrat.	Redução de NP Três estratégias de mut.	Brest e Maučec (2011)
2011	p-ADE	W, K, F, CR	Nova estrat. de mutação Inercial, cognitiva, social	Bi e Xiao (2011)
2011	SaDE-MMTS	F, CR	Mut. JADE Large Scale Optimization	Zhao, Suganthan e Das (2011)
2011	CoDE	Mut. estrat.	3 configurações para parâmetros	Wang, Cai e Zhang (2011)
2012	MDE-pBX	F, CR	Semelhante ao JADE Nova mutação e crossover	Islam et al. (2012)
2013	SHADE	F, CR	Aprimoramento do JADE Múltiplas médias de F e CR	Tanabe e Fukunaga (2013)
2013	ADE	F, CR	Funções seno e cosseno	Huang e Chen (2013)
2014	L-SHADE	F, CR, NP	SHADE com redução linear de NP	Tanabe e Fukunaga (2014)
2016	iL-SHADE	F, CR, NP	Aprimoramento do L-SHADE Memória de F e CR	Brest, Maučec e Bošković (2016)
2016	L-SHADE-EpSin	F, CR, NP	Variação senoidal de F Busca local	Awad et al. (2016)
2017	L-SHADE-cnEpSin	F, CR, NP Crossover	L-SHADE-EpSin com matriz de covariância para crossover Busca local	Awad, Ali e Suganthan (2017)
2018	SHADE-ILS	F, CR	Critério de reinício	Molina, LaTorre e Herrera (2018)
2018	LSGOjDE	Mut. estrat.	Máximos variáveis p. F e CR Redução de NP	Maučec et al. (2018)
2018	EB-L-SHADE	F, CR, NP Mut. estrat.	L-SHADE com duas estratégias de mutação	Mohamed, Hadi e Jambi (2018)

Fonte: elaborado pelo autor.

A *Self-Adaptive Pareto Differential Evolution* (SPDE) é uma variação auto-adaptativa da *Pareto Differential Evolution* (PDE), um algoritmo de DE voltado à otimização de problemas com múltiplos objetivos, e que usa para isso o conceito de otimalidade de Pareto. Na SPDE, cada indivíduo possui os seus próprios valores dos parâmetros F e CR , e estes são herdados dos seus pais, de forma similar a como os indivíduos doadores são gerados a partir de três indivíduos na DE canônica (ABBASS, 2002). Sendo assim, a SPDE utiliza um controle *online* agregado.

Differential Evolution using Pre-Calculated Differential (DEPD) e DE_n são duas modificações ao algoritmo DE propostas por Ali e Törn (2004) e que podem ser

utilizadas em conjunto. Nestes algoritmos, o fator de mutação F é escalonado em função da diferença entre maior e menor *fitness* da população. O DEPD introduz o conceito de diferenciais pré-calculadas, em que a diferença entre indivíduos calculada na etapa da mutação da primeira geração é guardada em um vetor e reutilizada na mutação das próximas R gerações, ponto em que estas diferenciais serão atualizadas, com este ciclo sendo mantido até o fim da evolução. Já o DE_n consiste em utilizar uma população auxiliar que pode realizar trocas de indivíduos com a população principal de três formas distintas, gerando assim os algoritmos DE1, DE2 e DE3 (ALI; TÖRN, 2004). Como a adaptação do fator F usa o *fitness* como parâmetro, o controle *online* deste algoritmo é classificado como adaptável – regra simples.

O *Fuzzy Adaptive Differential Evolution* (FADE) usa o conceito de lógica *Fuzzyy* para adaptar os parâmetros F e CR da Evolução Diferencial, utilizando-se para isto de dois controladores lógicos *Fuzzy*. Estes controladores recebem como entrada a raiz quadrada média das diferenças entre duas gerações consecutivas dos valores da função e dos membros da população, e retornam os parâmetros F e CR (LIU; LAMPINEN, 2005). Este algoritmo utiliza controle de parâmetros *online* adaptável – controle *Fuzzy*.

O algoritmo *Self-adaptive Differential Evolution* (SaDE) utiliza duas estratégias de mutação, que podem ser selecionadas com probabilidades p_1 e p_2 , sendo que estes valores são adaptados com base no sucesso de cada estratégia nas gerações passadas. O SaDE também modifica os valores dos parâmetros F e CR , sendo que cada um deles é gerado aleatoriamente com distribuições normais diferentes, porém com valores de média e desvio padrão que se mantém constantes ao longo da evolução (QIN; SUGANTHAN, 2005). Assim, pode-se dizer que o SaDE utiliza duas formas de controle *online*: adaptável – regras simples para as estratégias de mutação, e determinístico para os parâmetros F e CR .

O *Self-adaptive Differential Evolution* (SDE) foi elaborado com base no SPDE, utilizando conceitos parecidos, porém para problemas de objetivo único. Nele, o parâmetro F de cada indivíduo é herdado dos seus pais, enquanto o valor de CR é gerado por meio de uma distribuição normal para cada indivíduo (OMRAN; SALMAN; ENGELBRECHT, 2005). Assim, tem-se um controle *online* determinístico para o parâmetro CR , e um agregado para F .

No algoritmo *Differential Evolution with Self-Adapting Populations* (DESAP),

é realizada a adaptação tanto dos parâmetros F e CR quanto do tamanho da população NP , sendo que o valor inicial de NP é calculado a partir do número de dimensões do problema. Além de F e CR , cada indivíduo também carrega um parâmetro π , sendo que estes três parâmetros são herdados dos seus pais. Ao fim de uma geração, NP é calculado a partir dos valores de π dos indivíduos (TEO, 2006). Este algoritmo utiliza controle de parâmetros *online* do tipo agregado.

No algoritmo jDE, a adaptação dos parâmetros F e CR pode ocorrer de duas formas: eles podem ser herdados de um dos pais do indivíduo ou recalculados independentemente dos parâmetros deste pai. As probabilidades de F e CR serem recalculados são τ_1 e τ_2 , respectivamente. Caso F seja recalculado, este cálculo usará os parâmetros F_l e F_u (BREST et al., 2006). Sendo assim, o jDE utiliza os controles *online* agregado e adaptável – regras simples.

O dynNP-DE é uma variação do jDE que adapta também o valor do tamanho da população, NP . Neste algoritmo, o tamanho da população é reduzido pela metade periodicamente ao longo da evolução, sendo que é aplicado o mesmo número de avaliações de função *fitness* para cada tamanho da população (BREST; MAUČEC, 2008). Além das formas de controle *online* já utilizadas no jDE, o dynNP-DE utiliza também um controle determinístico para o tamanho da população.

O algoritmo *Evolutionary Self-Adaptive Differential Evolution* (ESADE) utiliza uma população extra, de tamanho 6 e com indivíduos unidimensionais, para evoluir o parâmetro F , enquanto o parâmetro CR é gerado aleatoriamente, a partir de uma distribuição normal, para cada geração (EPITROPAKIS; PLAGIANAKOS; VRAHATIS, 2009). Assim, tem-se um controle *online* agregado para o valor de F , e determinístico para o valor de CR .

No JADE, os parâmetros F e CR de cada indivíduo são calculados aleatoriamente a partir de uma distribuição de Cauchy com média μ_F e normal com média μ_{CR} , respectivamente. Estas médias são inicializadas em 0.5 e recalculadas a cada geração a partir dos valores de F e CR dos indivíduos que geraram filhos aceitos para a próxima geração. O JADE utiliza uma estratégia de mutação própria, que utiliza um indivíduo aleatório entre os $p\%$, e permite também o uso de uma população extra que pode ter seus indivíduos selecionados para a mutação (ZHANG; SANDERSON, 2009). O controle *online* utilizado pelo JADE é classificado como adaptável – regras simples.

O *Adaptive Strategy Selection – JADE* (AdapSS-JADE) modifica o JADE adicionando a ele mais estratégias de mutação e um mecanismo para dinamicamente selecionar a melhor estratégia. São utilizadas 4 estratégias e um mecanismo de aprendizado para determinar a probabilidade de cada estratégia ser selecionada (GONG et al., 2011). Além do controle *online* utilizado pelo JADE para os parâmetros F e CR , o AdapSS-JADE utiliza um controle adaptável – autômato de aprendizagem para a estratégia de mutação.

O *Ensemble of mutation strategies and parameters in DE* (EPSDE) utiliza um conjunto de estratégias de mutação e de valores pré-definidos para F e CR . Na população inicial, cada indivíduo recebe um elemento aleatório de cada um destes três conjuntos, e gera seus descendentes com base neles. Conjuntos de valores que geraram descendentes de sucesso são passados para estes, de modo que o algoritmo evolui seus parâmetros em conjunto aos seus indivíduos (MALLIPEDDI et al., 2011). O EPSDE utiliza um controle de parâmetros *online* agregado.

O *jDE for large-scale optimization problems* (jDElscop) é uma modificação do jDE que utiliza um mecanismo para redução de NP semelhante ao do dynNP-DE. No jDElscop, os valores de ambos F e CR são calculados com base em parâmetros extras, diferente do jDE que os utiliza somente para o cálculo de F . Outras diferenças deste algoritmo são o uso de três estratégias de mutação e de um mecanismo para inverter o sinal de F (BREST; MAUČEC, 2011). O controle *online* deste algoritmo é agregado para os parâmetros F e CR , e determinístico para NP e para a estratégia de mutação.

No algoritmo *pbest vector-based self-Adaptive Differential Evolution* (p-ADE), é introduzida uma nova estratégia de mutação que utiliza o melhor indivíduo global e também a melhor solução que o indivíduo sendo modificado já apresentou ao longo da evolução. Esta estratégia também utiliza dois novos parâmetros, W e K , que correspondem à partes inercial e social da mutação, enquanto F corresponde à parte cognitiva. Estes três parâmetros, além do CR , são escalonados em relação aos valores de *fitness* da população (BI; XIAO, 2011). O controle *online* de parâmetros do p-ADE é classificado como adaptável – regras simples.

O algoritmo *SaDE with Modified Multi-Trajectory Search* (SaDE-MMTS) utiliza a adaptação de parâmetros do SaDE, adicionando a ele a estratégia de mutação do JADE. O SaDE-MMTS também utiliza uma busca de multi-trajetórias, sendo que os agentes de busca deste algoritmo são selecionados da população do SaDE, e as avaliações

de função *fitness* são distribuídas entre estes dois algoritmos com base em suas taxas de sucesso (ZHAO; SUGANTHAN; DAS, 2011). Além da parametrização *online* utilizada pelo SaDE, a parametrização do SaDE-MMTS é adaptável – regras simples para a busca de multi-trajetórias.

O *Composite Differential Evolution* (CoDE) utiliza 3 diferentes estratégias de mutação e 3 combinações de parâmetros F e CR . Nas etapa de mutação e *crossover*, cada indivíduo gera 3 vetores de teste, sendo um para cada estratégia de mutação, e cada um deles usa uma das três combinações de parâmetros, selecionada aleatoriamente. Somente o melhor dos três indivíduos de teste será avaliado para entrar ou não na próxima geração (WANG; CAI; ZHANG, 2011). O CoDE utiliza uma parametrização *online* do tipo determinística.

O *Modified DE with p-best crossover* (MDE-pBX) tem como inspiração o JADE, utilizando um mecanismo de adaptação semelhante para os parâmetros F e CR . O MDE-pBX utiliza também uma nova estratégia de mutação, que utiliza um parâmetro q , e uma forma diferente de *crossover*, que utiliza um parâmetro p , além do CR (ISLAM et al., 2012). No MDE-pBX, o controle *online* de F e CR é do tipo adaptável – regras simples, assim como no JADE. O controle de p , por sua vez, é realizado de forma determinística, enquanto q é mantido constante ao longo da evolução.

O algoritmo *Success-History based Adaptive Differential Evolution* (SHADE) foi elaborado a partir do JADE, sendo um aprimoramento deste. O SHADE utiliza as mesmas técnicas de adaptação do JADE, sendo as principais diferenças: são usadas múltiplas médias para os parâmetros F e CR , no lugar de apenas uma, e o valor de p para a mutação é gerado aleatoriamente para cada indivíduo, no lugar de ser fixo (TANABE; FUKUNAGA, 2013). Sendo assim, além do controle *online* utilizado pelo JADE, o SHADE também utiliza um controle determinístico para o valor de p .

No algoritmo *Adaptive Differential Evolution* (ADE) realiza um escalonamento nos parâmetros F e CR utilizando, para isso, composições com funções seno e cosseno. As funções elaboradas utilizam dois novos parâmetros, α e β , que são mantidos constantes (HUANG; CHEN, 2013). O controle *online* de parâmetros utilizado pelo ADE é classificado como determinístico.

O algoritmo SHADE with Linear Population Size Reduction (L-SHADE) foi elaborado como um aprimoramento do SHADE, acrescentando a este um mecanismo

para redução linear do valor de NP ao longo da evolução (TANABE; FUKUNAGA, 2014). Sendo assim, além dos controles *online* de parâmetros já utilizados pelo SHADE, o L-SHADE também utiliza um controle determinístico para o valor de NP .

O iL-SHADE, por sua vez, foi elaborado como um aprimoramento do L-SHADE. Nele, o valor inicial das médias de CR é aumentado de 0.5 para 0.8, e são alteradas as formas como as médias de F e CR são calculadas e atualizadas de uma geração para a próxima. É alterada também a forma como o parâmetro p é calculado, sendo agora atualizado uma vez a cada geração, no lugar de ser calculado para cada indivíduo (BREST; MAUČEC; BOŠKOVIĆ, 2016). Apesar das alterações, a classificação do controle *online* dos parâmetros do iL-SHADE continua a mesma do L-SHADE: adaptável – regras simples para F e CR , e determinístico para p e NP .

O L-SHADE-EpSin é uma modificação do L-SHADE que modifica a adaptação do parâmetro F e utiliza uma busca local quando a evolução está próxima do fim. O parâmetro F é adaptado a partir de duas funções senoides durante as primeiras 50% avaliações de função, sendo que uma delas utiliza um parâmetro adaptativo, enquanto a segunda metade da evolução utiliza a adaptação padrão do L-SHADE para o F . Já a busca local é aplicada a uma nova população de 10 indivíduos aleatórios, quando a população principal atinge o tamanho de 20 indivíduos. Caso a busca local produza indivíduos melhores que os da população original, os 10 piores indivíduos desta população são substituídos pelos encontrados pela busca local (AWAD et al., 2016). Para o L-SHADE-EpSin, o controle *online* de parâmetros é o mesmo do L-SHADE, com adição do controle determinístico para a adaptação do F na primeira metade da evolução.

O L-SHADE-cnEpSin utiliza como base o L-SHADE-EpSin, fazendo duas alterações: não é realizada a busca local próxima ao final da evolução, e é utilizada uma segunda forma de *crossover*, que usa uma matriz de covariância para transformar os indivíduos participantes da operação. A operação de *crossover* a ser utilizada para cada indivíduo é determinada por meio de uma probabilidade fixa (AWAD; ALI; SUGANTHAN, 2017). Além do controle *online* de parâmetros utilizado pelo L-SHADE-EpSin, este algoritmo também utiliza um controle determinístico para a escolha da operação de *crossover*.

O SHADE *with Iterative Local Search* (SHADE-ILS) utiliza a versão original do SHADE em conjunto com duas técnicas de busca local e um mecanismo de reinício que é aplicado quando a solução melhora menos de 5% ao longo de três gerações. A escolha

da técnica de busca local a ser utilizada em uma geração usa como base a melhora de *fitness* causada por ambas as técnicas nas gerações anteriores. (MOLINA; LATORRE; HERRERA, 2018). O SHADE-ILS utiliza parametrização *online* do tipo adaptável – regras simples para a probabilidade de escolha de cada técnica de busca local.

O *Large Scale Global Optimization* jDE (LSGOjDE) utiliza os mecanismos de adaptação do jDE, sendo suas principais diferenças: são utilizadas três estratégias de mutação, sendo que o valor máximo permitido para os valores F e CR varia de acordo com a estratégia usada, e é incorporado um mecanismo para redução de NP . Uma das novas estratégias utilizadas é uma busca local cujo intervalo de busca é adaptado em relação ao seu sucesso ou fracasso nas gerações anteriores. A probabilidade de seleção de cada estratégia varia de forma determinística ao longo das gerações da evolução (MAUČEC et al., 2018). Além da classificação do jDE, a parametrização *online* do LSGOjDE é do tipo determinística para a redução de NP e para a escolha da estratégia de mutação, e adaptável – regras simples para o intervalo da busca local.

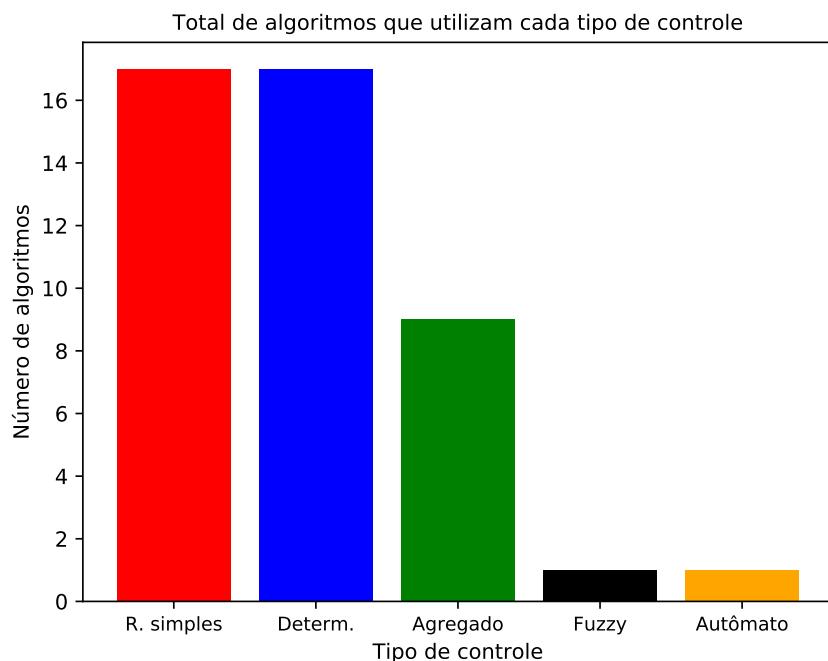
O EB-L-SHADE utiliza, em conjunto à mutação current-to-pbest/1/bin, uma versão modificada desta chamada current-to-ordpbest/1/bin. Esta segunda mutação tem a mesma forma da primeira, com a diferença de que, nela, a ordem dos indivíduos pode ser alterada de acordo com o *fitness* relativo entre eles. A escolha de qual mutação será aplicada a cada indivíduo é feita a partir de uma probabilidade que é adaptada de forma semelhante à adaptação de F e CR utilizada pelo L-SHADE (MOHAMED; HADI; JAMBI, 2018). Neste algoritmo, o controle *online* de parâmetros segue a mesma classificação do L-SHADE, com a adição do controle adaptável - regras simples para a escolha da estratégia de mutação.

2.4.1 Visão Geral das Técnicas Utilizadas

A figura 2.5 apresenta o número de algoritmos que utiliza cada uma das técnicas de controle *online* citadas na seção anterior. Foram inclusas somente as técnicas que são utilizadas em ao menos um algoritmo. Caso um algoritmo utilize mais de uma técnica, todas as técnicas utilizadas por ele são contabilizadas neste gráfico.

Pode-se ver na figura 2.5 que as técnicas mais utilizadas são a de regras simples e determinístico. Isto se dá pela popularidade do SHADE, que utiliza estes tipos de controle e serviu de inspiração para múltiplos outros algoritmos. O controle agregado

Figura 2.5: Número de algoritmos que utilizam cada uma das técnicas de controle citadas na seção 2.4



Fonte: autoria própria.

também é bastante utilizado, principalmente pelos algoritmos anteriores ao SHADE na linha do tempo. Já os controles *fuzzy* e com autômato de aprendizagem são utilizados por somente um algoritmo cada, e uma hipótese para isto é o alto nível de complexidade destas técnicas, principalmente em comparação às outras mais utilizadas.

2.5 Algoritmos Selecionados

Dentre os algoritmos apresentados na seção 2.4, o SHADE, o L-SHADE e o EB-L-SHADE foram selecionados para implementação e avaliação. Esta escolha vêm da variedade de algoritmos elaborados com base no SHADE, o que evidencia o seu potencial, sendo o L-SHADE e o EB-L-SHADE duas destas variações no estado da arte.

Estes algoritmos são explicados em detalhes nas subseções seguintes. Também é apresentado em detalhes o algoritmo JADE, pois ele é a base para o SHADE, e portanto o seu entendimento auxilia na compreensão do SHADE.

2.5.1 JADE

O algoritmo JADE (ZHANG; SANDERSON, 2009) segue o princípio básico do DE canônico, porém difere deste em três pontos: uso de um histórico de indivíduos A , estratégia de mutação, e adaptação dos parâmetros F e CR . Cada um destes pontos é detalhado nos parágrafos seguintes.

O histórico A funciona da seguinte forma: toda vez que um indivíduo da população produz um descendente que o substitui, este indivíduo é adicionado a A . Ao fim de uma geração, se o tamanho do histórico $|A|$ for maior que NP , indivíduos de A são removidos, de forma aleatória, até que $|A| \leq NP$.

A estratégia de mutação utilizada pelo JADE tem o nome DE/current-to-pbest/1/bin, e é dada pela seguinte equação:

$$v_{i,g+1} = x_{i,g} + F_i(x_{best,g}^p - x_{i,g}) + F_i(x_{r1,g} - x'_{r2,g}) \quad (2.3)$$

onde $x_{i,g}$ é o indivíduo pai, F_i é o seu valor para o parâmetro F , $x_{best,g}^p$ é um indivíduo selecionado aleatoriamente entre os $p\%$ melhores da geração atual, $x_{r1,g}$ é um indivíduo selecionado aleatoriamente da população principal, e $x'_{r2,g}$ é um indivíduo selecionado aleatoriamente da união entre a população principal e o histórico. Na ausência de histórico, $x'_{r2,g}$ é selecionado da população principal somente. Os indivíduos $x_{r1,g}$ e $x'_{r2,g}$ devem ser diferentes entre si, e diferentes de x_i .

A adaptação dos parâmetros, por sua, vez, é feita da seguinte forma: a cada geração, os valores de F e CR para um dado indivíduo i são calculados de acordo com as equações:

$$F_i = randc(\mu_F, 0.1) \quad (2.4)$$

$$CR_i = randn(\mu_{CR}, 0.1) \quad (2.5)$$

onde $randc(\mu_F, 0.1)$ indica um valor gerado aleatoriamente com uma distribuição de Cauchy, com média μ_F e variância 0.1. Já $randn(\mu_{CR}, 0.1)$ é um valor gerado com distribuição normal de média μ_{CR} e desvio padrão 0.1. O valor de CR_i é truncado entre $[0, 1]$, enquanto F_i é truncado para 1 caso $F_i > 1$ ou recalculado caso $F_i \leq 0$. As médias μ_F e μ_{CR}

inicializadas em 0.5 e recalculadas ao fim de cada geração de acordo com as equações:

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F) \quad (2.6)$$

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}) \quad (2.7)$$

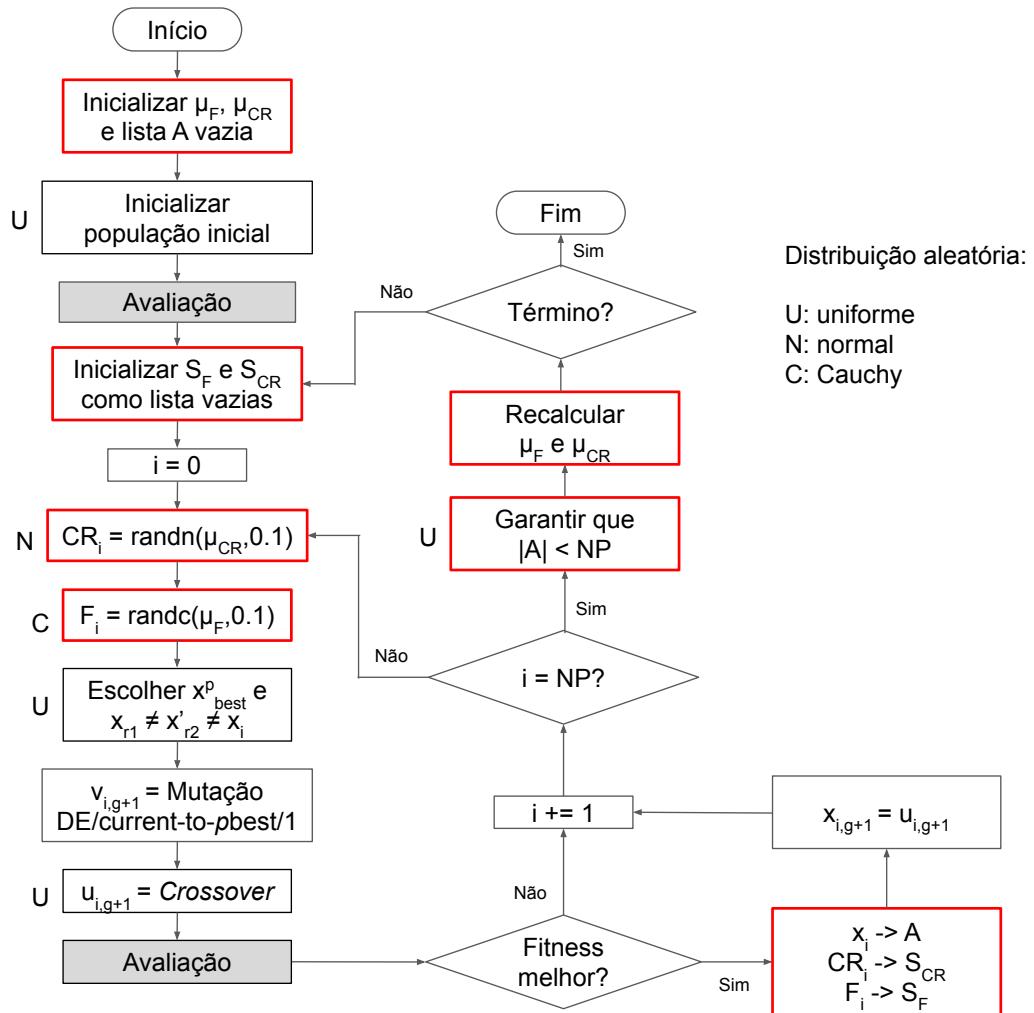
onde c é uma constante entre 0 e 1, S_F e S_{CR} são listas que contém, respectivamente, os valores dos parâmetros F e CR de todos os indivíduos que geraram descendentes de sucesso na geração atual. O termo $mean_A(S_{CR})$ na equação 2.7 representa a média aritmética de todos os valores de S_{CR} , enquanto $mean_L(S_F)$, na equação 2.6, representa a média de Lehmer dos valores de S_F . A média de Lehmer é calculada de acordo com a equação:

$$mean_L(S_F) = \frac{\sum_{k=1}^{k \leq |S_F|} F_k^2}{\sum_{k=1}^{k \leq |S_F|} F_k} \quad (2.8)$$

O fluxograma apresentado na figura 2.6 resume o funcionamento do algoritmo JADE. Nele, as etapas em que há geração de números aleatórios estão destacadas com as letras “U”, “N” e “C”, representando, respectivamente, distribuições uniforme, normal e de Cauchy. Estão também destacadas, com borda vermelha, as etapas em que o JADE se diferencia do DE canônico. As etapas em que foi aplicado paralelismo nos experimentos estão com o fundo destacado.

O JADE introduz dois novos parâmetros: p e c . Para o valor de p , Zhang e Sanderson (2009) recomendam $p \in [0.05, 0.2]$, isto é, p englobe de 5% a 20% da população. Já para c , a recomendação é também $c \in [0.05, 0.2]$, de modo que o tempo de vida de um dado valor de μ_F ou μ_{CR} fique entre 5 e 20 gerações. Os valores iniciais para μ_F e μ_{CR} são $\mu_F = \mu_{CR} = 0.5$. Segundo Zhang e Sanderson (2009), o uso da distribuição de Cauchy para a geração de F tem por objetivo aumentar a diversidade dos fatores de mutação em comparação a uma distribuição normal, o que torna uma convergência prematura menos provável.

Figura 2.6: Fluxograma para o algoritmo JADE.



Fonte: elaborado a partir de (ZHANG; SANDERSON, 2009).

2.5.2 SHADE

O algoritmo SHADE (TANABE; FUKUNAGA, 2013) foi elaborado com base no JADE, portanto somente os pontos em que eles diferem serão detalhados nesta seção. O SHADE difere do JADE nos seguintes pontos: auto-adaptação do parâmetro p , uso de múltiplas médias para os parâmetros F e CR , e a forma como estas médias são atualizadas a cada geração. Os parágrafos seguintes detalham cada uma destas diferenças.

Enquanto no JADE o parâmetro p , utilizado para determinar o tamanho do conjunto de $p\%$ melhores indivíduos, é mantido constante, no SHADE ele é recalculado para cada indivíduo. Este cálculo é feito da seguinte forma:

$$p = \text{randu}[p_{min}, 0.2] \quad (2.9)$$

onde p_{min} é o valor mínimo para p , definido como $p_{min} = 2/NP$, ou seja, ao menos 2 indivíduos farão parte do conjunto dos $p\%$ melhores. Já $randu[p_{min}, 0.2]$ indica um valor aleatório com distribuição uniforme entre p_{min} e 0.2.

No lugar das médias μ_F e μ_{CR} , o SHADE utiliza duas listas, M_F e M_{CR} , de tamanho H , servindo como memórias para múltiplas médias dos parâmetros F e CR . A figura 2.7 ilustra estas memórias.

Figura 2.7: Memórias históricas M_{CR} e M_F .

Index	1	2	...	$H - 1$	H
M_{CR}	$M_{CR,1}$	$M_{CR,2}$...	$M_{CR,H-1}$	$M_{CR,H}$
M_F	$M_{F,1}$	$M_{F,2}$...	$M_{F,H-1}$	$M_{F,H}$

Fonte: (TANABE; FUKUNAGA, 2013).

A cada geração, os valores de F e CR de cada indivíduo são calculados primeiramente selecionando um valor r_i aleatório no intervalo $[1, H]$. Os valores são então calculados de acordo com as equações:

$$F_i = randc(M_{F,r_i}, 0.1) \quad (2.10)$$

$$CR_i = randn(M_{CR,r_i}, 0.1) \quad (2.11)$$

onde, assim como no JADE, $randc$ e $randn$ representam valores aleatórios gerados com distribuições de Cauchy e normal, respectivamente. Caso os valores de F_i ou CR_i sejam gerados fora do intervalo $[0, 1]$, eles são reajustados da mesma forma que no JADE (TANABE; FUKUNAGA, 2013).

Ao fim de uma geração, os valores de índice k das memórias M_F e M_{CR} serão atualizados. Este índice é inicializado com o valor 0, e é incrementado em 1 toda vez que as memórias forem atualizadas. A atualização da posição k das memórias é dada por:

$$M_{F,k,g+1} = \begin{cases} mean_{WL}(S_F) & \text{se } S_F \neq \emptyset \\ M_{F,k,g} & \text{caso contrário} \end{cases} \quad (2.12)$$

$$M_{CR,k,g+1} = \begin{cases} mean_{WA}(S_{CR}) & \text{se } S_{CR} \neq \emptyset \\ M_{CR,k,g} & \text{caso contrário} \end{cases} \quad (2.13)$$

onde, como o JADE, S_F e S_{CR} são listas que contém os parâmetros F e CR de todos os indivíduos que geraram descendentes de sucesso na geração atual. São utilizadas as médias ponderadas $mean_{WL}$ (Lehmer) e $mean_{WA}$ (aritmética) para a atualização dos valores nos históricos. O cálculo destas médias é realizado de acordo com as equações:

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{k \leq |S_F|} w_k \cdot S_{F,k}^2}{\sum_{k=1}^{k \leq |S_F|} w_k \cdot S_{F,k}} \quad (2.14)$$

$$mean_{WA}(S_{CR}) = \sum_{k=1}^{k \leq |S_{CR}|} w_k \cdot S_{CR,k} \quad (2.15)$$

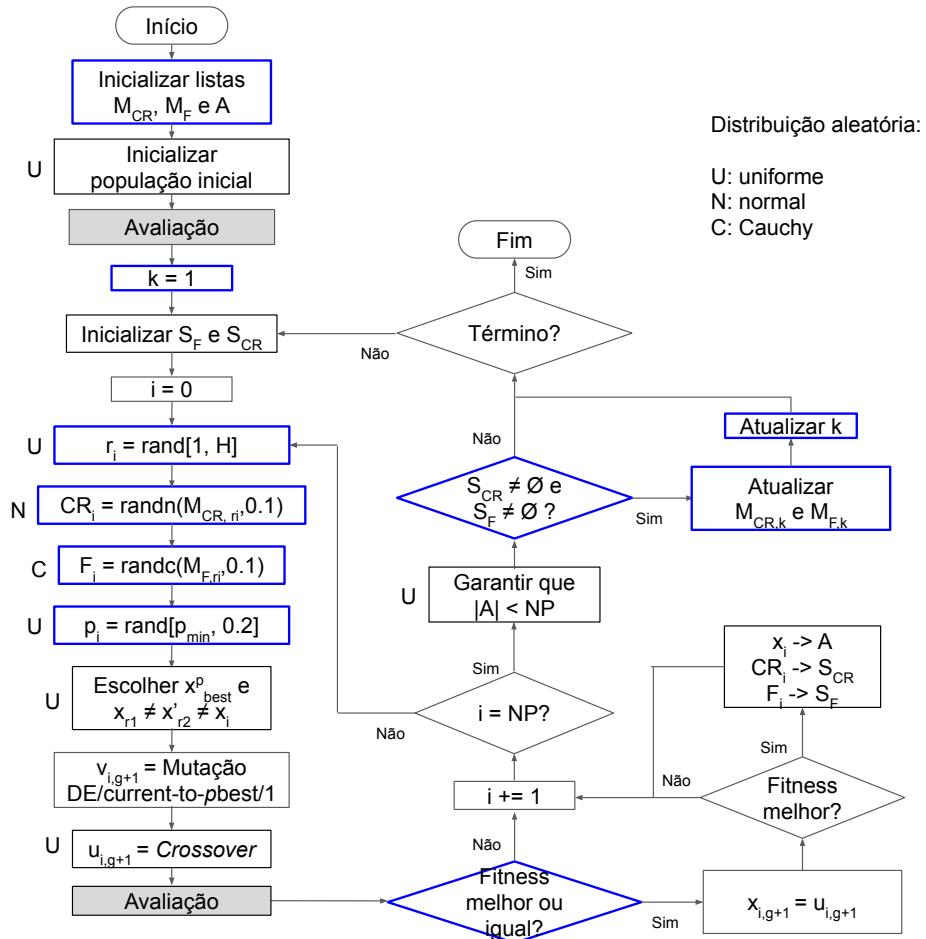
$$w_k = \frac{\Delta f_k}{\sum_{j=1}^{j \leq |S_{CR}|} \Delta f_j} \quad (2.16)$$

onde $\Delta f_k = |f(u_{k,g+1}) - f(x_{k,g})|$, com $x_{k,g}$ um indivíduo da população atual, e $u_{k,g+1}$ o indivíduo de teste que o substituiu.

O fluxograma apresentado na figura 2.8 resume o funcionamento do algoritmo SHADE. Nela, as etapas em que há geração de números aleatórios estão destacadas com as letras “U”, “N” e “C”, representando distribuições uniforme, normal e de Cauchy, respectivamente. Estão também destacadas, com borda em azul, as etapas em que o SHADE se diferencia do JADE. Já as etapas com fundo destacado são aquelas em que foi aplicado paralelismo nos experimentos realizados.

Dos parâmetros adicionados pelo JADE, o SHADE remove o parâmetro c , e altera p para que este não seja mais definido pelo usuário. O SHADE introduz um novo parâmetro, H , que representa o tamanho das memórias M_F e M_{CR} . Tanabe e Fukunaga (2013) apresentam um estudo do impacto do valor de H , e chegam à conclusão que 30, 50 e 100 são valores adequados. É utilizado o valor inicial 0.5 para todas as entradas das memórias M_F e M_{CR} .

Figura 2.8: Fluxograma para o algoritmo SHADE.



Fonte: elaborado a partir de (TANABE; FUKUNAGA, 2013).

SHADE 1.1

O SHADE 1.1 é uma atualização do SHADE apresentada por seus autores (TANABE; FUKUNAGA, 2014). Esta modificação afeta poucos pontos do algoritmo, de modo que não é considerada um algoritmo novo. É, porém, nesta versão do algoritmo que o L-SHADE é baseado.

Os pontos em que o SHADE 1.1 difere do SHADE original são:

- Caso todos os valores de S_{CR} sejam iguais a 0 em uma geração, o valor de M_{CR} atual é atualizado para um valor especial \perp , e este valor não é mais alterado pelo resto do processo. Caso um indivíduo selecione um valor r_i tal que $M_{CR,r_i} = \perp$, o seu valor de CR será igual a 0;
- Em casos diferentes do descrito acima, a atualização do valor atual de M_{CR} passa a ser feita por meio da média ponderada de Lehmer, da mesma forma em que M_F

é atualizado na versão original do SHADE, como descrito na equação 2.14;

- O valor de p , utilizado na mutação, deixa de ser gerado para cada indivíduo, e passa a ser um parâmetro fixo e universal, definido no início da evolução, como no JADE.

2.5.3 L-SHADE

O L-SHADE adiciona um esquema de redução linear do tamanho da população NP ao SHADE 1.1. Isto é feito da seguinte maneira: ao final de cada geração, o valor de NP é atualizado de acordo com a fórmula:

$$NP_{g+1} = \text{round} \left[\frac{NP^f - NP^i}{MAX_NFE} \cdot NFE + NP^i \right] \quad (2.17)$$

onde NP^f e NP^i são os valores final e inicial de NP , respectivamente, MAX_NFE é o valor total de avaliações de função, e NFE é o valor atual.

Caso haja redução de população ao fim de uma geração, os indivíduos removidos são aqueles com os piores valores de *fitness*.

Outra diferença menor do L-SHADE é em relação ao tamanho do arquivo externo A utilizado para a mutação. Enquanto no SHADE e no JADE tem-se $|A| = NP$, no L-SHADE tem-se que $|A| = r^{arc} \cdot NP_g$, isto é, o tamanho do arquivo, em uma dada geração, é o valor de NP nesta geração multiplicado por uma constante r^{arc} .

2.5.4 EB-L-SHADE

O EB-L-SHADE é idêntico ao L-SHADE, exceto em ponto: o uso de uma segunda estratégia de mutação, em conjunto da estratégia current-to-pbest/1/bin do L-SHADE. Esta nova estratégia é chamada current-to-ord_pbest/1/bin, e funciona de forma semelhante à já utilizada pelo algoritmo. Nela, primeiramente são selecionados três indivíduos aleatórios, sendo um deles do conjunto dos $p\%$ melhores indivíduos, um da população principal, e um da união entre a população principal e o arquivo externo. Estes três indivíduos são então ordenados em relação ao *fitness*, sendo $x_{ord_pbest}^G$ o melhor dos três, $x_{ord_pmedian}^G$ o mediano, e $x_{ord_pworst}^G$ o pior. A mutação é então aplicada da seguinte forma:

$$v_i^{g+1} = x_i^g + F_i^g \cdot (x_{ord_pbest}^g - x_i^g) + F_i^g \cdot (x_{ord_pmedian}^g - x_{ord_pworst}^g) \quad (2.18)$$

onde x_i^g é o indivíduo atual, e F_i^G é o seu valor para o parâmetro F .

Para a escolha de qual estratégia será aplicada a cada indivíduo, é utilizado um mecanismo para adaptação da probabilidade de qual estratégia será escolhida, com base na eficácia de cada uma delas ao longo de uma geração. É usada uma memória M_{FCP} , de mesmo tamanho que as memórias M_F e M_{CRE} , e é utilizado o mesmo mecanismo destas para escolher qual de seus elementos será utilizado pelos indivíduos, e qual será atualizado ao fim de cada geração.

Para atualizar o elemento i de M_{FCP} , ao fim de uma geração, primeiramente é calculada a eficácia de cada mutação da seguinte forma:

$$\omega_{m1} = \sum_{j=1}^n f(x_i^g) - f(u_i^g) \quad (2.19)$$

onde $f(x_i^g)$ é a função *fitness* aplicada ao indivíduo original, e $f(u_i^g)$ é a função aplicada ao indivíduo de teste, e o somatório é calculado sobre os indivíduos de teste gerados pela mutação $m1$. Como o somatório das probabilidades de cada mutação deve ser igual a 100%, tem-se que $\omega_{m2} = 1 - \omega_{m1}$.

Na sequência, a taxa de aprimoramento Δ_m desta mutação é calculada de acordo com:

$$\Delta_{m1} = \min \left(0.8, \max \left(0.2, \frac{\omega_{m1}}{\omega_{m1} - \omega_{m2}} \right) \right) \quad (2.20)$$

com 0.2 e 0.8 sendo os valores mínimos e máximos para as probabilidades de cada mutação.

O valor de $M_{FCP,i}^{g+1}$ é então calculado como:

$$M_{FCP,i}^{g+1} = (1 - c)M_{FCP,i}^g + c\Delta_{m1} \quad (2.21)$$

onde c é uma taxa de aprendizado que determina o quanto rapidamente o valor será atualizado ao longo das gerações (MOHAMED; HADI; JAMBI, 2018).

2.6 Considerações Parciais

Este capítulo apresentou uma visão geral sobre os diferentes temas tratados neste trabalho: problemas de otimização, algoritmos evolutivos, com destaque à Evolução Diferencial, técnicas de parametrização e os principais algoritmos de *Differential Evolution* (DE) com parametrização adaptativa encontrados na literatura.

A revisão apresentada neste capítulo permitiu identificar os principais algoritmos de DE com auto ajuste de parâmetros, fornecendo um vetor para as etapas seguintes do trabalho. Foi identificado que o algoritmo SHADE apresenta um grande potencial, evidenciado pela quantidade de algoritmos elaborados a partir dele. Entre estes algoritmos, estão o L-SHADE e o EB-L-SHADE, que são estado da arte. Atenção especial merece ser dada ao EB-L-SHADE, devido à adaptação da estratégia de mutação, o que não é feito nos algoritmos que deram origem a ele. Devido a isto, estes algoritmos foram selecionados para implementação, avaliação e proposição de modificações, como será discutido no Capítulo 4.

O contínuo interesse científico na auto-adaptação dos parâmetros do DE é confirmado pela grande quantidade de algoritmos propostos para este fim, como por exemplo o SHADE-ILS, o LSGOjDE e o EB-L-SHADE, propostos em 2018, indicando que esta é uma área ativa de pesquisa.

3 Trabalhos Relacionados

Este capítulo apresenta uma relação de trabalhos publicados na literatura e que estão relacionados ao tema presente. Neles, são exploradas diferentes técnicas de parametrização, são realizadas análises sobre algum algoritmo de DE adaptativo ou são apresentadas aplicações de algum destes algoritmos a um problema real. Não estão inclusos neste capítulo trabalhos que apenas propõe uma nova forma de adaptação dos parâmetros do DE, pois estes trabalhos já são apresentados na seção 2.4.

André e Parpinelli (2014) apresentam uma visão geral sobre as diferentes formas de parametrização *online* e suas aplicações em algoritmos evolutivos e de inteligência de enxame. Além de propor a taxonomia para parametrização apresentada na figura 2.4, são revisadas e classificadas diferentes técnicas de parametrização *online* apresentadas na literatura para algoritmos de Evolução Diferencial, Algoritmos Genéticos, Otimização por Colônias Artificiais de Abelhas, Otimização por Colônias de Formigas, entre outros.

Gosh et al. (2017) propõem uma modificação para o DE utilizando seleção baseada em distância, com o objetivo de otimizar problemas com presença de ruido. Além de utilizar o centroide da população na operação de mutação, Gosh et al. (2017) utilizam uma variação da operação de seleção clássica onde, caso o indivíduo de teste possua um *fitness* pior que o do seu pai, este ainda possui uma probabilidade p de entrar na população, dada por:

$$p_s = e^{-\frac{\Delta f}{Dist}} \quad (3.1)$$

onde Δf é a diferença absoluta entre o *fitness* dos indivíduos pai e de teste, e $Dist$ é a distância entre eles no espaço de busca. Gosh et al. (2017) avaliam o uso desta seleção nos algoritmos SHADE e L-SHADE aplicados a funções com presença de ruido, e foi observada uma melhora na eficácia destes algoritmos.

Tanabe e Fukunaga (2015) avaliam a parametrização de versões com reinício dos algoritmos DE, SHADE e L-SHADE para cenários com diferentes custos computacionais: altos ($10^2 \times D$ avaliações de *fitness*), médios ($10^4 \times D$ avaliações) e baixos ($10^5 \times D$ avaliações), onde D é a dimensionalidade do problema. Foi utilizada a ferramenta para

configuração de algoritmos SMAC para determinar o tamanho da população e os valores iniciais dos parâmetros para cada um dos três cenários, considerando problemas com até 20 dimensões. Todas as configurações foram aplicadas a todos os cenários, e foi observado que, em cada cenário, os melhores resultados foram obtidos com as configurações projetadas para ele. Observou-se também que, para custos computacionais altos, o DE apresentou melhores resultados, enquanto que, para custos médios e baixos, SHADE e L-SHADE foram melhores (TANABE; FUKUNAGA, 2015).

Skanderova (2017) apresenta uma análise de como a adaptação *online* de parâmetros influencia o espalhamento de genes positivos em uma população. Para isto, foram utilizadas redes temporais para avaliar os algoritmos DE e jDE, sendo comparadas características como entropia e tempo de espera de cada rede. Foi observado que, no jDE, bons genes se espalham pela rede mais rapidamente que no DE canônico (SKANDEROVA, 2017).

Senkerik et al. (2018) analisam o impacto do uso de séries caóticas no lugar de geradores de números pseudo-aleatórios nos algoritmos jDE, SHADE e duas variações do DE tradicional (DE/Rand e DE/Best). As comparações foram realizadas entre 9 séries caóticas e um gerador de números pseudo-aleatórios tradicional, o algoritmo *Linear Congruential Generator* (LCG). Os testes foram realizados em problemas de *benchmark* com 10 dimensões e populações de 50 indivíduos e, de modo geral, foi observado que as séries caóticas aprimoram os resultados dos algoritmos, em comparação ao LCG (SENKERIK et al., 2018).

Piotrowski e Napiorkowski (2018) argumenta contra a complexidade excessiva apresentada por algumas meta-heurísticas, como por exemplo o algoritmo L-SHADE-EpSin. (AWAD et al., 2016). Neste trabalho, é feita uma análise detalhada da busca local utilizada pelo L-SHADE-EpSin, mostrando que esta apresenta uma forte tendência em direção à origem do sistema de coordenadas, o que torna os seus resultados não confiáveis. A partir deste e de outro exemplo, Piotrowski e Napiorkowski (2018) discute como o acréscimo contínuo de sub-rotinas a algoritmos já estabelecidos pode trazer efeitos indesejados, como uma busca tendenciosa, de modo que deve-se ter cautela quando se realiza este tipo de operação.

Tanabe e Fukunaga (2016) questionam como um DE adaptativo ótimo deveria se comportar, e o quão próximos deste ótimo teórico estão os melhores algoritmos de DE

adaptativos atuais. Para isto, é proposto o *framework Greedy Approximate Oracle method* (GAO), que simula a escolha de parâmetros ótimos a cada passo ao gerar indivíduos de teste para todos as combinações de valores de F e CR pertencentes a um conjunto de tamanho λ , gerado de forma aleatória, e usar somente o melhor indivíduo produzido desta forma. Este *framework* é utilizado para avaliar os algoritmos jDE, EPSDE, JADE, MDE e SHADE, além de outros algoritmos que não pertencem à Evolução Diferencial. O GAO apresenta eficácia superior a todos os algoritmos de DE avaliados, mostrando que ainda existe potencial para aprimoramentos destes algoritmos (TANABE; FUKUNAGA, 2016).

3.1 Considerações Parciais

Este capítulo apresentou um conjunto de trabalhos que abordam temas de Evolução Diferencial e parametrização adaptativa. Com estes trabalhos, pode-se observar que a pesquisa nestes temas vai além da simples proposta de novas técnicas de parametrização *online*, havendo também trabalhos com foco na análise de comportamento e eficácia dos algoritmos (TANABE; FUKUNAGA, 2015; SKANDEROVA, 2017; TANABE; FUKUNAGA, 2016) e avaliação do impacto de uma alteração que possa ser aplicada a mais de um algoritmo (GOSH et al., 2017; SENKERIK et al., 2018).

A análise fornecida por Tanabe e Fukunaga (2016) não apenas mostra que ainda há potencial para o aprimoramento dos mecanismos de adaptação de parâmetros no DE, como fornece *insights* de como estes valores podem variar ao longo da evolução, o que pode ser levado em consideração para a elaboração de novos algoritmos.

Já a análise de Tanabe e Fukunaga (2015) reforça que, mesmo que um algoritmo seja adequado a uma situação, não necessariamente ele será adequado a outras situações, mostrando que mesmo o DE canônico pode ser competitivo ou até superior a variações mais complexas em certos cenários.

4 Metodologia

Este capítulo apresenta os algoritmos desenvolvidos e avaliados, com a motivação destas escolhas sendo apresentadas na seção 4.1. Já na seção 4.2, são apresentadas quais as alterações a estes algoritmos foram propostas e avaliadas.

Também neste capítulo, na seção 4.3 é apresentada a métrica para medida de diversidade utilizada nos experimentos, e não seção 4.5 são apresentados os testes estatísticos aplicados aos resultados dos experimentos, juntamente da motivação para a sua aplicação. A seção 4.6 apresenta as considerações parciais para este capítulo.

4.1 Algoritmos escolhidos

Foram utilizados como base para comparação os algoritmos DE canônico, SHADE, L-SHADE e o EB-L-SHADE. A escolha do L-SHADE foi motivada por seu status como estado da arte, enquanto o EB-L-SHADE foi avaliado por ter apresentado bons resultados em relação ao L-SHADE na literatura (MOHAMED; HADI; JAMBI, 2018), além de que a modificação proposta por ele foi utilizada em conjunto com outras modificações, como explicado na seção 4.2. Já o DE canônico e SHADE foram incluídos por serem diferentes passos na construção do L-SHADE, e espera-se que os resultados obtidos comprovem esta evolução.

4.2 Alterações avaliadas

Além dos algoritmos mencionados, foram avaliadas três modificações para o L-SHADE.

Elas são:

- Decaimento alternativo para o tamanho da população;
- Uso conjunto de duas estratégias de mutação: DE/current-to-pbest/1/bin e DE/current-to-ord_pbest/1/bin, como no EB-L-SHADE;

- Uso de uma variação da rotina de mutação não uniforme de Michalewicz (MICHALEWICZ, 1996), aplicada aos indivíduos de teste produzidos pela evolução;

Cada uma destas modificações são explicadas e justificadas nas subseções seguintes.

4.2.1 Detalhes e justificativas

Decaimento alternativo de NP

O decaimento linear de população, utilizado pelo L-SHADE, apresenta comportamentos não ideais quando o algoritmo é aplicado a problemas de dimensionalidade alta, como 100 dimensões. Para este caso, considerando os valores inicial e final de NP recomendados pelo autor (TANABE; FUKUNAGA, 2014), o tamanho da população varia entre 1800 a 4 indivíduos, linearmente em relação às avaliações de função. Com $10^4 \times D$ avaliações de função disponíveis, o algoritmo será executado por aproximadamente 3400 gerações. Metade das avaliações de função são atingidas na geração 385, com uma população de 903 indivíduos. Aproximadamente 44,6% das avaliações de função são aplicadas a uma população maior de 1000 indivíduos, e 72,4% das avaliações são aplicadas a uma população maior de 500 indivíduos. Assim, muitas avaliações de função são desperdiçadas com tamanhos de população grandes, de modo que sobram poucas avaliações para os valores menores de NP .

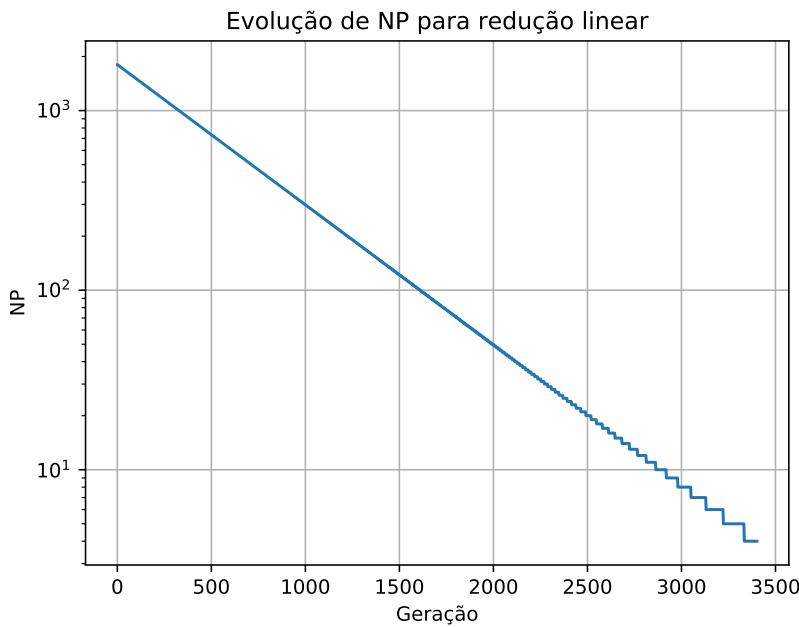
A figura 4.1 mostra a evolução de NP em relação as gerações para o L-SHADE em 100 dimensões. Nela, a área abaixo da curva representa o número de avaliações de função realizados. Pode-se perceber que a maior parte das avaliações são aplicadas a populações grandes, enquanto os valores menores de NP têm poucas avaliações a disposição.

Uma alternativa para a redução linear de NP é dada pelo uso da equação

$$NP_i = NP_0 \left(\frac{NP_f}{NP_0} \right)^{\frac{i}{MAX_NFE}} \quad (4.1)$$

onde MAX_NFE é o número total de avaliações de função disponíveis. Utilizando os mesmos valores do L-SHADE para NP_0 e NP_f , em 100 dimensões, a evolução terá um total de aproximadamente 40.000 gerações, sendo que, em aproximadamente 5.000 delas, o tamanho da população será o mínimo. Isso gera um problema oposto ao anterior, em

Figura 4.1: Tamanho da população em relação ao número de gerações, para o L-SHADE em 100 dimensões.



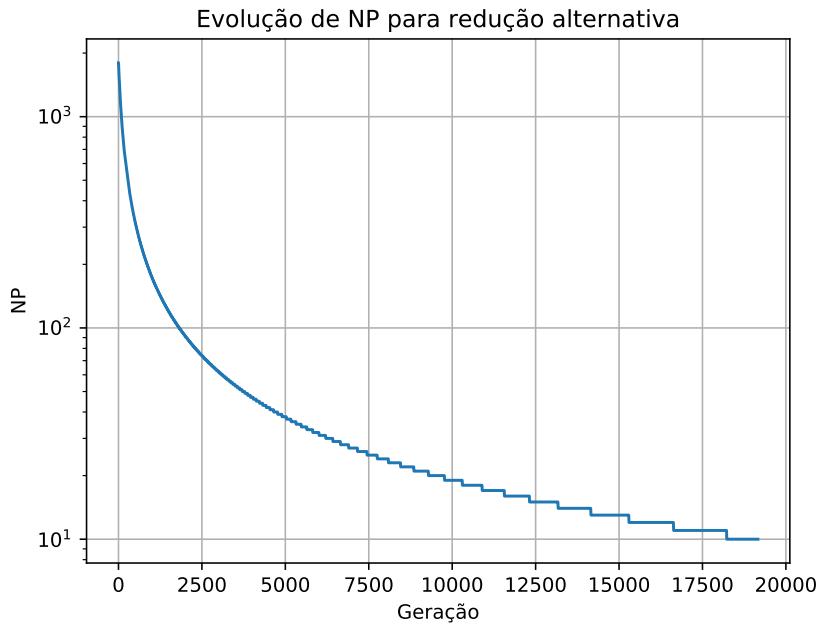
Fonte: autoria própria.

que uma grande quantidade de avaliações de função são utilizadas com uma população muito pequena. Alterando-se o valor de NP final para 10, no entanto, reduz-se o número de gerações para aproximadamente 19.100, com aproximadamente 1.000 delas com população de tamanho mínimo. Nestas condições, aproximadamente 11,5% das avaliações são aplicadas a uma população maior de 1.000 indivíduos, e 24,7% aplicadas a uma população maior de 500 indivíduos, reduzindo o desperdício de avaliações com valores de NP grandes.

A figura 4.2 mostra a evolução de NP seguindo a dinâmica descrita acima, para 100 dimensões. Nela, é possível observar uma distribuição mais equilibrada das avaliações de função para os diferentes tamanhos de população, em comparação à redução linear, mostrada na figura 4.1.

Com a alteração do método de redução de população, espera-se diminuir o desperdício de avaliações de função no início da busca, fornecendo a ela um número maior de gerações para convergir ao final da busca. Esta modificação do L-SHADE será referida como A-SHADE (*SHADE with Alternative population size reduction*) no restante deste trabalho.

Figura 4.2: Tamanho da população em relação ao número de gerações, com a redução acelerada de NP em 100 dimensões.



Fonte: autoria própria.

Uso conjunto de duas estratégias de mutação

Outro algoritmo que se pretende avaliar é o EB-L-SHADE em conjunto da redução alternativa de NP descrita anteriormente. O uso conjunto das mutações current-to-pbest e current-to-ord_pbest se mostrou competitivo (MOHAMED; HADI; JAMBI, 2018), e portanto pretende-se também avaliar o impacto desta hibridização de estratégias de mutação aplicadas ao A-SHADE. No resto deste trabalho, este algoritmo será referido como EB-A-SHADE.

Mutação não-uniforme de Michalewicz

A mutação não-uniforme é um operador que, quando aplicado a um gene de um indivíduo, aplica uma perturbação ao valor deste gene (MICHALEWICZ, 1996). Sendo v_k o valor original do gene, o seu valor modificado v'_k , após a aplicação do operador, será:

$$v'_k = \begin{cases} v_k + \Delta(t, UB - v_k) & \text{se } pr \leq 0.5 \\ v_k - \Delta(t, v_k - LB) & \text{caso contrário} \end{cases} \quad (4.2)$$

onde LB e UB são os valores mínimo e máximo para gene, e pr é um valor aleatório gerado uniformemente entre 0 e 1. A função $\Delta(t, y)$ é definida como:

$$\Delta(t, y) = y \cdot (1 - r^{(1 - \frac{t}{T})^b}) \quad (4.3)$$

onde r é outro valor aleatório gerado uniformemente entre 0 e 1, T é o número máximo de iterações, t é a geração atual, e b é um parâmetro do sistema que determina o grau de dependência com o número da iteração. Conforme indicado em Michalewicz (1996), é usado $b = 5$.

O efeito deste operador é explorar o espaço de busca uniformemente durante o início da busca, e explorar localmente ao fim. Isto acontece porque a função $\Delta(t, y)$ retorna um valor no intervalo $[0, y]$, porém o valor retornado tende a 0 conforme t se aproxima de T .

O objetivo do uso deste operador é avaliar se ele causa um impacto positivo na diversidade da população, reduzindo a convergência prematura dela. Para isso, ele será aplicado aos indivíduos de teste, com uma probabilidade, após a primeira mutação e o *crossover*, afetando um número pré-determinado de genes.

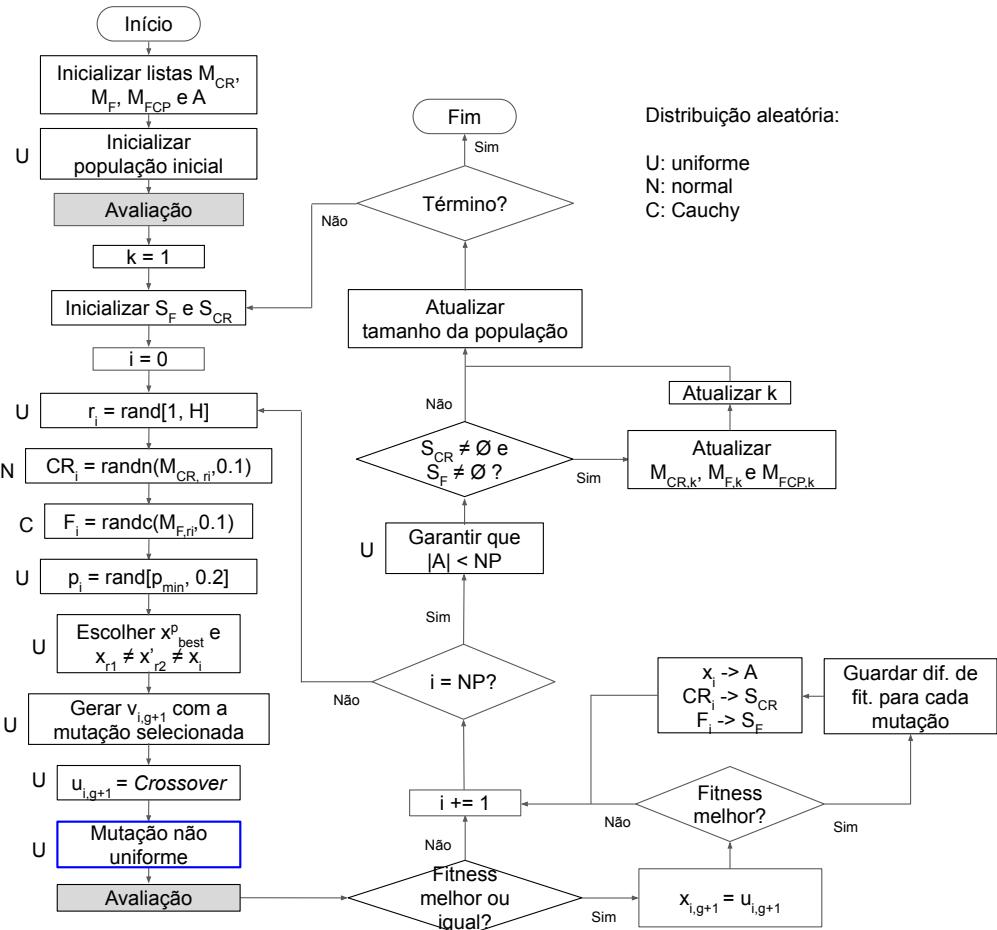
A busca global que este operador realiza nas primeiras gerações, no entanto, não é interessante para o algoritmo avaliado, pois ela poderia impedir os indivíduos de teste de gerar soluções melhores que seus progenitores. Devido a isto, o operador será aplicado com uma modificação na potência $1 - \frac{t}{T}$ na fórmula do $\Delta(t, y)$, de modo a gerar somente perturbações pequenas no valor dos genes. Para este fim, o valor de $\frac{t}{T}$ será substituído por valores entre 0.75 e 1, que equivalem ao último quarto da evolução com o operador tradicional. Este valor irá variar linearmente de 0.75 a 1, e em seguida de volta a 0.75, também linearmente, completando assim um ciclo. O número de ciclos cl será um parâmetro fixo ao longo da evolução.

Neste trabalho, esta variação da mutação não uniforme será aplicada ao EB-A-SHADE, produzindo o algoritmo EB-A-SHADE-M. Esta operação será aplicada aos indivíduos de teste, após a etapa de *crossover*. Neste algoritmo, define-se como p_M a probabilidade de um indivíduo de teste ser selecionado para a mutação não uniforme, e como p_G a probabilidade de um gene deste indivíduo ser modificado pela mutação. A quantidade de genes selecionados é feita de forma determinística, isto é, se $D = 100$ e $p_G = 0.1$, então exatamente 10 genes aleatórios do indivíduo serão selecionados para a

mutação.

A figura 4.3 apresenta o fluxograma do EB-A-SHADE-M. Nele, a etapa em que a mutação não uniforme é aplicada está destacada com borda azul. As etapas em que há geração de números aleatórios estão destacadas com as letras “U”, “N” e “C”, representando, respectivamente, distribuições uniforme, normal e de Cauchy. As etapas em que foi aplicado paralelismo nos experimentos estão com o fundo destacado.

Figura 4.3: Fluxograma para o algoritmo EB-A-SHADE-M.



Fonte: elaborado a partir de (MOHAMED; HADI; JAMBI, 2018) e adaptado pelo autor.

4.3 Diversidade

Nos experimentos realizados, a diversidade genética populacional é calculada ao final de cada geração, para fins de análise de comportamento dos algoritmos.

Existem múltiplas formas de se calcular a diversidade, sendo a mais comum a distância euclideana média de todos para todos os indivíduos. Esta métrica, no entanto,

apresenta complexidade quadrática em relação ao tamanho da população, sendo assim um cálculo computacionalmente custoso. Outra técnica para se calcular a diversidade é apresentada por Morrison e Jong (2002), que faz uma analogia com a propriedade física do momento de inércia: tratando cada indivíduo como um ponto de mesma massa espalhados pelo espaço, o momento de inércia deste conjunto em relação ao seu centro de massa pode ser utilizado como uma métrica para diversidade.

Para este cálculo, primeiro deve-se determinar o centro de massa da população.

Cada coordenada c_i do centro de massa é dada por:

$$c_i = \frac{1}{NP} \sum_{j=1}^{NP} x_{ij} \quad (4.4)$$

com j representando o indivíduo de índice j na população. A diversidade ou momento de inércia pode então ser calculada como (MORRISON; JONG, 2002):

$$I = \sum_{i=1}^D \sum_{j=1}^N (x_{ij} - c_i)^2 \quad (4.5)$$

Esta medida tem a vantagem de apresentar complexidade linear em relação a NP, sendo assim computacionalmente menos custosa para se calcular. Ela tem, porém, a desvantagem de apresentar uma grande dispersão nos seus valores, que modo que uma análise da variação da diversidade ao longo da evolução se torna difícil. Para resolver este problema, será utilizada neste trabalho uma versão modificada desta métrica de diversidade, dada pela fórmula:

$$I' = \frac{1}{D} \cdot \sum_{i=1}^D \sqrt{\frac{1}{N-1} \sum_{j=1}^N (x_{ij} - c_i)^2} \quad (4.6)$$

Com esta modificação, a diversidade pode ser vista como o desvio padrão médio de cada dimensão. Esta nova métrica mantém o custo computacional da métrica anterior, porém apresentando um comportamento semelhante ao obtido pela distância euclidiana média entre indivíduos.

4.4 Paralelismo

Com o objetivo acelerar a execução dos experimentos, foi utilizado processamento paralelo a nível de CPU nos algoritmos.

O paralelismo foi feito por meio de *OpenMP* (OpenMP Architecture Review Board, 2008) em C++, e foi aplicado à avaliação da população, de modo que o *fitness* de até n indivíduos é calculado simultaneamente, sendo n o número de *threads* utilizadas.

4.5 Análises estatísticas

Quando é necessário comparar conjuntos independentes de dados, não é possível saber, de imediato, se existe ou não diferença real entre estes conjuntos. Isto é, existe a possibilidade de que os valores dos conjuntos tenham origem de uma mesma distribuição, com as diferenças entre eles sendo causadas somente pela aleatoriedade da distribuição.

Para que se possa dizer com confiança se dois ou mais conjuntos de dados têm origem de distribuições diferentes, é necessária a aplicação de análises estatísticas a estes conjuntos. O retorno de uma destas análises é um $p - value$, que representa a probabilidade de os conjuntos observados terem como origem uma mesma distribuição. Assim, um $p - value$ pequeno indica uma probabilidade baixa, o que indica que os conjuntos provavelmente vêm de distribuições diferentes, enquanto um $p - value$ alto indica o contrário.

Para as análises estatísticas realizadas neste trabalho, será utilizado um grau de confiança de 95%, ou seja, serão considerados como estatisticamente diferentes conjuntos que apresentarem $p - value < 0.05$.

Dois testes estatísticos são utilizados neste trabalho. O primeiro deles é o teste de Kruskal-Vallis, que é aplicado simultaneamente a todos os conjuntos que devem ser comparados. Este teste retorna um $p - value$ único, que indica se existe ou não alguma diferença estatística entre algum dos conjuntos analisados (OSTERTAGOVA; OSTERTAG; KOVÁČ, 2014), porém não indica especificamente entre quais conjuntos existe diferença.

O segundo teste aplicado é o teste *post-hoc* de Dunn. Esta análise é realizada após o teste de Kruskal-Vallis, somente caso este aponte alguma diferença entre os conjun-

tos. O teste de Dunn, então, retorna um *p-value* para cada par de conjuntos, permitindo identificar exatamente quais deles são estatisticamente diferentes de quais outros (DUNN, 1961).

4.6 Considerações Parciais

Neste capítulo, foram apresentados os algoritmos escolhidos para experimentação, juntamente com as modificações propostas e as justificativas para estas modificações. Foram apresentadas também outros pontos da metodologia do trabalho, como a técnica de paralelismo utilizada nos experimentos, e as análises estatísticas aplicadas aos resultados obtidos, além da justificativa para estas análises.

Com as modificações propostas, têm-se definidos exatamente quais algoritmos vão ser avaliados, e com as análises estatísticas definidas, têm-se como eles serão comparados. Os pontos que restam detalhar são as funções de *benchmark* às quais os algoritmos serão submetidos, e os parâmetros exatos utilizados nos experimentos. Estes dois pontos são discutidos nos próximos dois capítulos.

5 Funções para Otimização Contínua sem Restrições

Para a avaliação dos algoritmos, foram utilizadas as funções de *benchmark* do evento *Congress on Evolutionary Computing* (CEC) 2013 (LIANG et al., 2013). O conjunto completo de *benchmarks* foi utilizado para o TCC-2.

Todas as funções do conjunto de *benchmark* do CEC 2013 são deslocadas, e parte delas são também rotacionadas ou manipuladas de alguma outra forma (LIANG et al., 2013). As fórmulas apresentadas na seção 5.1 são as formas básicas de cada uma das funções, sem estas transformações. Liang et al. (2013) expõe exatamente de que forma cada uma das funções foi transformada.

A seção 5.2 explica as características principais destas funções, com relação à modalidade, separabilidade, entre outras. Na seção 5.3, são apresentadas as considerações parciais para o conteúdo apresentado neste capítulo.

5.1 Definição das funções

Nesta seção, são apresentadas as versões básicas de cada uma das funções de *benchmark* utilizadas. Nelas, D é o número de dimensões do problema.

5.1.1 Funções simples

1 – Esfera:

$$f(x) = \sum_{i=1}^D x_i^2 \quad (5.1)$$

Ótimo global: -1400.0.

2 – Elipse altamente condicionada (rotacionada):

$$f(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2 \quad (5.2)$$

Ótimo global: -1300.0.

3 – Bent cigar (rotacionada):

$$f(x) = x_1^2 + 10^6 \sum_{i=2}^D x_i^2 \quad (5.3)$$

Ótimo global: -1200.0.

4 – Discus (rotacionada):

$$f(x) = 10^6 x_1^2 + \sum_{i=2}^D x_i^2 \quad (5.4)$$

Ótimo global: -1100.0.

5 – Different powers:

$$f(x) = \sqrt{\sum_{i=1}^D |x_i|^{2+4\frac{i-1}{D-1}}} \quad (5.5)$$

Ótimo global: -1000.0.

6 – Rosenbrock (rotacionada):

$$f(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \quad (5.6)$$

Ótimo global: -900.0.

7 – Schaffer F7 (rotacionada):

$$f(x) = \left(\frac{1}{D-1} \sum_{i=1}^{D-1} (\sqrt{x_i} + \sqrt{x_i} \sin^2(50x_i^{0.2})) \right)^2 \quad (5.7)$$

Ótimo global: -800.0.

8 – Ackley (rotacionada):

$$f(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e \quad (5.8)$$

Ótimo global: -700.0.

9 – Weierstrass (rotacionada):

$$f(x) = \sum_{i=1}^D \left[\sum_{k=0}^{kmax} (a^k \cos(2\pi b^k (x_i + 0.5))) - D \sum_{k=0}^{kmax} (a^k \cos(2\pi b^k \cdot 0.5)) \right] \quad (5.9)$$

com $a = 0.5$, $b = 3$ e $kmax = 20$. Ótimo global: -600.0.

10 – Griewank (rotacionada):

$$f(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (5.10)$$

Ótimo global: -500.0.

11 – Rastrigin:

$$f(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (5.11)$$

Ótimo global: -400.0.

12 – Rastrigin (rotacionada):

Mesma fórmula base da função 11 (equação 5.11). Ótimo global: -300.0.

13 – Rastrigin (rotacionada e com descontinuidades):

Mesma fórmula base da função 11 (equação 5.11). Ótimo global: -200.0.

14 – Schwefel:

$$f(x) = 418.9829 \times D - \sum_{i=1}^D x_i \sin(\sqrt{|x_i|}) \quad (5.12)$$

Ótimo global: -100.0.

15 – Schwefel (rotacionada):

Mesma fórmula base da função 14 (equação 5.12). Ótimo global: 100.0.

16 – Katsuura (rotacionada):

$$f(x) = \frac{10}{D^2} \prod_{i=1}^D \left(1 + i \sum_{j=1}^{32} \frac{2^j x_i - \text{round}(2^j x_i)}{2^j} \right)^{\frac{10}{D+2}} - \frac{10}{D^2} \quad (5.13)$$

Ótimo global: 200.0.

17 – Lunacek Bi_Rastrigin:

$$f(x) = \min \left(\sum_{i=1}^D (x_i - \mu_0)^2, dD + s \sum_{i=1}^D (x_i - \mu_1)^2 + 10(D - \sum_{i=1}^D \cos(2\pi(x_i - \mu_0))) \right) \quad (5.14)$$

Com $\mu_0 = 2.5$, $\mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}$, $s = 1 - \frac{1}{2\sqrt{D+20}-8.2}$ e $d = 1$. Ótimo global: 300.0.

18 – Lunacek Bi_Rastrigin (rotacionada):

Mesma fórmula base da função 17 (equação 5.14). Ótimo global: 400.0.

19 – Griewank expandida somada à função Rosenbrok:

Sendo $g_1(x)$ a função Griewank, dada pela equação 5.10, e $g_2(x)$ a função Rosenbrock, dada pela equação 5.6, a função 19 é definida como:

$$f(x) = g_1(g_2(x_1, x_2)) + g_1(g_2(x_2, x_3)) + \dots + g_1(g_2(x_{D-1}, x_D)) + g_1(g_2(x_D, x_1)) \quad (5.15)$$

Ótimo global: 500.0.

20 – Schaffer F6 expandida:

Sendo a função Schaffer F6: $g(y, z) = 0.5 + \frac{\sin^2(\sqrt{y+z^2}) - 0.5}{(1 + 0.001(x^2+y^2))^2}$, a função Schaffer F6 expandida é dada por:

$$f(x) = g(x_1, x_2) + g(x_2, x_3) + \dots + g(x_{D-1}, x_D) + g(x_D, x_1) \quad (5.16)$$

Ótimo global: 600.0.

5.1.2 Funções compostas

Cada uma das funções compostas do conjunto é gerada a partir da combinação de um número n das funções simples definidas anteriormente. A função $f(x)$ resultante da com-

binação é gerada da seguinte forma:

$$f(x) = \sum_{i=1}^n (\omega_i \times [\lambda_i g_i(x) + bias_i]) \quad (5.17)$$

onde $g_i(x)$ é uma das funções simples utilizadas, λ_i é utilizado para controlar a altura de cada função, e ω_i é um peso normalizado aplicado a cada função. O peso ω_i é normalizado seguindo $\omega_i = w_i / \sum_{i=1}^n w_i$, onde w_i é dado por:

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^D (x_j - o_{ij})^2}} \exp\left(-\frac{\sum_{j=1}^D (x_j - o_{ij})^2}{2D\sigma_i^2}\right) \quad (5.18)$$

onde o_i é a posição ótima deslocada de cada $g_i(x)$ e σ_i controla a cobertura de cada $g_i(x)$.

As funções compostas utilizadas são:

21 – Função composta 1 (rotacionada, n = 5):

$$\sigma = [10, 20, 30, 40, 50]$$

$$\lambda = [1, 1e-6, 1e-26, 1e-6, 0.1]$$

$$bias = [0, 100, 200, 300, 400]$$

g_1 : Rosenbrock rotacionada.

g_2 : *Different powers* rotacionada.

g_3 : *Bent cigar* rotacionada.

g_4 : Discus rotacionada.

g_5 : Esfera.

Ótimo global: 700.0.

22 – Função composta 2 (não rotacionada, n = 3):

$$\sigma = [20, 20, 20]$$

$$\lambda = [1, 1, 1]$$

$$bias = [0, 100, 200]$$

g_{1-3} : Schwefel.

Ótimo global: 800.0.

23 – Função composta 3 (rotacionada, n = 3):

$$\sigma = [20, 20, 20]$$

$$\lambda = [1, 1, 1]$$

$$bias = [0, 100, 200]$$

g_{1-3} : Schwefel rotacionada.

Ótimo global: 900.0.

24 – Função composta 4 (rotacionada, n = 3):

$$\sigma = [20, 20, 20]$$

$$\lambda = [0.25, 1, 2.5]$$

$$bias = [0, 100, 200]$$

g_1 : Schwefel rotacionada.

g_2 : Rastrigin rotacionada.

g_3 : Weierstrass rotacionada.

Ótimo global: 1000.0.

25 – Função composta 5 (rotacionada, n = 3):

$$\sigma = [10, 30, 50]$$

$$\lambda = [0.25, 1, 2.5]$$

$$bias = [0, 100, 200]$$

g_1 : Schwefel rotacionada.

g_2 : Rastrigin rotacionada.

g_3 : Weierstrass rotacionada.

Ótimo global: 1100.0.

26 – Função composta 6 (rotacionada, n = 5):

$$\sigma = [10, 10, 10, 10, 10]$$

$$\lambda = [0.25, 1, 1e - 7, 2.5, 10]$$

$$bias = [0, 100, 200, 300, 400]$$

g_1 : Schwefel rotacionada.

g_2 : Rastrigin rotacionada.

g_3 : Elipse altamente condicionada rotacionada.

g_4 : Weierstrass rotacionada.

g_5 : Griewank rotacionada.

Ótimo global: 1200.0.

27 – Função composta 7 (rotacionada, n = 5):

$$\sigma = [10, 10, 10, 10, 10]$$

$$\lambda = [100, 10, 2.5, 25, 0.1]$$

$$bias = [0, 100, 200, 300, 400]$$

g_1 : Griewank rotacionada.

g_2 : Rastrigin rotacionada.

g_3 : Schwefel rotacionada.

g_4 : Weierstrass rotacionada.

g_5 : Esfera.

Ótimo global: 1300.0.

28 – Função composta 8 (rotacionada, n = 5):

$$\sigma = [10, 20, 30, 40, 50]$$

$$\lambda = [2.5, 2.5e - 3, 2.5, 5e - 4, 0.1]$$

$$bias = [0, 100, 200, 300, 400]$$

g_1 : Griewank expandida somada à função Rosenbrok rotacionada.

g_2 : Schaffers F7 rotacionada.

g_3 : Schwefel rotacionada.

g_4 : Schaffers F6 expandida rotacionada.

g_5 : Esfera.

Ótimo global: 1400.0.

5.2 Características das funções

As funções deste conjunto de *benchmark* são problemas de otimização contínuos e sem restrições, sendo assim classificados como FOP, como é discutido na seção 2.1.1.

Quanto à modalidade, tem-se que as funções 1 – 5 são unimodais, enquanto as funções 6 – 28 são multimodais com um único ótimo global. Outra característica importante é a separabilidade, que é a possibilidade de se otimizar uma dimensão de forma independente das outras em uma função. Quando à separabilidade, as funções 1, 5, 11 e 22 são separáveis, enquanto as demais não são separáveis.

Uma característica comum às funções compostas (21 – 28) é a presença de diferentes propriedades em torno de diferentes ótimos locais. Devido a isto, uma estratégia de busca que funcione para escapar de um destes ótimos pode não funcionar para escapar de outro, o que torna estas funções significativamente mais difíceis.

5.3 Considerações Parciais

Este capítulo apresentou as funções de *benchmark* utilizadas neste trabalho para a avaliação dos algoritmos experimentados, incluindo as suas principais características.

Dentre as 28 funções deste conjunto, 23 delas são multimodais, e 24 não são separáveis. Além disso, 8 destas funções são composições de outras funções do conjunto. Isto mostra que este conjunto é composto por funções difíceis, que em sua maioria não poderiam ser solucionadas por algoritmos tradicionais, mas que são ideais para avaliar os algoritmos estudados neste trabalho.

6 Experimentos, Resultados e Análises

Este capítulo apresenta os resultados obtidos, assim como as suas análises e os experimentos realizados para obtê-los. A seção 6.1 explica o protocolo de experimentos utilizado, incluindo o detalhamento do ambiente de experimentos, os parâmetros utilizados para os algoritmos, e como os resultados serão apresentados. Na seção 6.2, os resultados obtidos são apresentados e analisados.

6.1 Protocolo de Experimentos

A implementação de todos os algoritmos foi realizada em linguagem de programação C++. Em todas as etapas em que há geração de números aleatórios, foi utilizado para este fim o algoritmo Mersenne Twister (MATSUMOTO; NISHIMURA, 1998). Os testes estatísticos foram realizados utilizando a linguagem R, e os gráficos de convergência e diversidade foram elaborados com o uso da biblioteca matplotlib da linguagem Python. Os códigos e os resultados obtidos estão disponíveis em <https://github.com/ChrisRenka/TCC>.

Para as análises estatísticas, foi realizado o teste de Kruskal-Wallis para identificar se existe diferença significativa entre os diferentes testes, seguido do teste *post-hoc* de Dunn para avaliar a significância das diferenças entre algoritmos, caso o teste de Kruskal-Wallis retorne positivo (DUNN, 1961). Foi utilizada o método de Holm para o ajuste dos P-values obtidos nos testes de Dunn, com grau de confiança de 95%.

Os experimentos foram executados em computadores com um processador Intel Core i7-4770, com 4 núcleos físicos e 8 virtuais, 16 GB de memória RAM e sistema operacional Linux, com distribuição Ubuntu 18.04 LTS.

Para cada função e algoritmo, foram realizados 30 experimentos independentes considerando 100 dimensões, com um total de $10^4 \times D$ avaliações de função para cada experimento. Para os algoritmos com variação de NP , caso o número de avaliações disponíveis não seja suficiente para concluir a última geração, esta geração é descartada. Os domínios das variáveis (genes) de cada indivíduo são limitados ao intervalo [-100, 100].

Foram utilizados os parâmetros recomendados na literatura para cada algo-

ritmo (STORN; PRICE, 1997; TANABE; FUKUNAGA, 2013; TANABE; FUKUNAGA, 2014; MOHAMED; HADI; JAMBI, 2018) sendo que estes foram:

- DE: $NP = 100$, $F = 0.5$, $CR = 0.9$ e mutação DE/rand/1/bin.
- SHADE: $NP = 100$, $p = 0.10$ e $H = 100$.
- L-SHADE: $p = 0.11$, $NP_i = 18 \cdot D$, $NP_f = 4$ e $r^{arc} = 1.4$.
- EBL-SHADE: $p = 0.11$, $NP_i = 18 \cdot D$, $NP_f = 4$, $r^{arc} = 1.4$ e $c = 0.2$.

Já para os algoritmos propostos neste trabalho, os parâmetros foram definidos em parte dos algoritmos originais, e em parte por experimentação. Estes foram:

- A-SHADE: $p = 0.11$, $NP_i = 18 \cdot D$, $NP_f = 10$ e $r^{arc} = 1.4$.
- EBA-SHADE: $p = 0.11$, $NP_i = 18 \cdot D$, $NP_f = 10$, $r^{arc} = 1.4$ e $c = 0.2$.
- EBA-SHADE-M: $p = 0.11$, $NP_i = 18 \cdot D$, $NP_f = 10$, $r^{arc} = 1.4$, $c = 0.2$, $p_M = 0.95$ e $p_G = 0.10$.

6.2 Resultados e Análises

A tabela 6.1 apresenta os erros médios e desvio padrão obtidos para cada algoritmo aplicado a cada uma das funções experimentadas. Define-se como erro a diferença entre o valor da função objetivo obtida pelo algoritmo e o valor do ótimo global conhecido para a função. Os valores apresentados nesta tabela foram calculados excluindo, sempre que aplicável, os *major outliers* do conjunto de resultados de cada função. Define-se como um *major outlier* todo resultado cujo quartil mais próximo, dentro do conjunto de resultados para uma dada função e algoritmo, estiver a uma distância maior que $3Q_R$ deste resultado, sendo Q_R a distância entre os quartis do conjunto.

Na tabela 6.1, estão destacados com negrito, para cada função, os algoritmos que obtiverem os melhores resultados, segundo as análises estatísticas descritas em 4.5.

Ainda na tabela 6.1, a linha “Destaque” mostra o número total de vezes que os resultados de cada algoritmo estiveram entre os melhores. Já a linha B/S/W (*Best/Same/Worst*) apresenta o número de funções em que o algoritmo EB-A-SHADE, proposto

neste trabalho, foi melhor, equivalente ou pior em comparação a cada um dos outros algoritmos, respectivamente.

O apêndice A apresenta os boxplots de todos os algoritmos para cada função, que podem servir de auxílio para a comparação dos algoritmos.

Tabela 6.1: Média e desvio padrão obtidos para os experimentos em 100 dimensões.

Função	DE	SHADE	L-SHADE	EBL-SHADE	A-SHADE	EBA-SHADE	EBA-SHADE-M
f_1	3.18e-13 ± 1.11e-13	4.55e-13 ± 0.00e+0	2.27e-13 ± 0.00e+0	2.27e-13 ± 0.00e+0	2.27e-13 ± 0.00e+0	2.27e-13 ± 0.00e+0	8.34e-13 ± 1.59e-13
	5.56e+6 ± 1.40e+6	1.19e+5 ± 4.28e+4	1.47e+5 ± 4.87e+4	1.43e+5 ± 4.10e+4	1.53e+5 ± 3.13e+4	1.36e+5 ± 3.05e+4	1.36e+5 ± 3.02e+4
f_3	2.15e+7 ± 1.66e+7	3.62e+7 ± 2.32e+7	3.33e+6 ± 2.91e+6	3.32e+6 ± 3.40e+6	1.40e+7 ± 1.27e+7	1.19e+7 ± 9.27e+6	9.34e+6 ± 6.42e+6
	4.80e+4 ± 7.96e+3	3.08e-4 ± 2.51e-4	1.57e-4 ± 1.34e-4	1.46e-5 ± 8.24e-6	8.59e-5 ± 6.08e-5	4.37e-5 ± 2.74e-5	2.08e-3 ± 8.14e-4
f_5	4.21e-13 ± 8.88e-14	5.46e-13 ± 0.00e+0	3.90e-13 ± 8.14e-14	3.41e-13 ± 0.00e+0	6.40e-13 ± 9.04e-14	5.57e-13 ± 8.98e-14	5.61e-9 ± 3.41e-9
	1.93e+2 ± 2.67e+1	1.04e+2 ± 5.46e+1	1.98e+2 ± 2.75e+1	2.05e+2 ± 2.82e+1	1.86e+2 ± 3.77e+1	1.91e+2 ± 2.88e+1	1.92e+2 ± 3.74e+1
f_7	1.89e+1 ± 6.95e+0	5.61e+1 ± 1.17e+1	7.62e+0 ± 2.41e+0	7.07e+0 ± 1.54e+0	1.18e+1 ± 2.67e+0	1.16e+1 ± 4.24e+0	1.16e+1 ± 3.02e+0
	2.13e+1 ± 2.08e-2	2.12e+1 ± 9.66e-2	2.13e+1 ± 4.36e-2	2.13e+1 ± 4.81e-2	2.12e+1 ± 4.67e-2	2.12e+1 ± 5.54e-2	2.12e+1 ± 2.79e-2
f_9	1.58e+2 ± 2.25e+0	1.38e+2 ± 2.68e+0	1.32e+2 ± 2.90e+0	1.33e+2 ± 2.40e+0	1.32e+2 ± 2.79e+0	1.31e+2 ± 3.21e+0	1.32e+2 ± 2.89e+0
	5.58e-02 ± 2.93e-2	2.85e-2 ± 1.68e-2	1.58e-2 ± 1.19e-2	1.29e-2 ± 1.28e-2	1.28e-2 ± 7.84e-3	1.99e-2 ± 1.36e-2	1.59e-2 ± 1.10e-2
f_{11}	7.74e+1 ± 1.83e+1	1.71e-13 ± 0.00e+0	1.21e-3 ± 6.64e-4	1.48e-3 ± 7.96e-4	1.89e-13 ± 0.00e+0	1.71e-13 ± 0.00e+0	6.80e-12 ± 0.00e+0
	8.42e+2 ± 2.29e+1	1.52e+2 ± 1.74e+1	6.54e+1 ± 9.00e+0	5.27e+1 ± 1.01e+1	4.46e+1 ± 4.59e+0	3.78e+1 ± 5.27e+0	3.95e+1 ± 4.27e+0
f_{13}	8.35e+2 ± 2.13e+1	4.01e+2 ± 5.31e+1	1.50e+2 ± 1.84e+1	1.36e+2 ± 1.75e+1	1.30e+2 ± 1.57e+1	1.09e+2 ± 1.94e+1	1.05e+2 ± 1.70e+1
	2.31e+4 ± 2.12e+3	2.73e-2 ± 1.01e-2	7.52e+1 ± 1.05e+1	9.20e+1 ± 1.56e+1	6.39e-2 ± 1.68e-2	6.20e-2 ± 1.40e-2	9.51e-2 ± 2.14e-2
f_{15}	3.04e+4 ± 4.57e+2	1.39e+4 ± 6.08e+2	1.56e+4 ± 6.68e+2	1.56e+4 ± 4.68e+2	1.24e+4 ± 5.50e+2	1.24e+4 ± 5.43e+2	1.22e+4 ± 8.11e+2
	3.93e+0 ± 2.18e-1	1.76e+0 ± 1.78e-1	1.89e+0 ± 1.60e-1	1.89e+0 ± 1.39e-1	1.47e+0 ± 2.11e-1	1.54e+0 ± 1.47e-1	1.44e+0 ± 1.73e-1
f_{17}	6.66e+2 ± 5.57e+1	1.02e+2 ± 0.00e+0	1.03e+2 ± 3.36e-1	1.03e+2 ± 3.11e-1	1.02e+2 ± 0.00e+0	1.02e+2 ± 0.00e+0	1.02e+2 ± 1.13e-9
	9.21e+2 ± 2.54e+1	2.92e+2 ± 1.89e+1	2.80e+2 ± 1.46e+1	2.79e+2 ± 1.44e+1	1.52e+2 ± 4.54e+0	1.50e+2 ± 2.70e+0	1.50e+2 ± 4.70e+0
f_{19}	6.74e+1 ± 3.79e+0	7.26e+0 ± 1.14e+0	7.31e+0 ± 2.77e-1	7.39e+0 ± 2.62e-1	4.38e+0 ± 1.83e-1	4.34e+0 ± 2.14e-1	4.49e+0 ± 2.27e-1
	5.00e+1 ± 0.00e+0	5.00e+1 ± 0.00e+0	4.97e+1 ± 4.11e-1	4.97e+1 ± 3.92e-1	5.00e+1 ± 0.00e+0	4.97e+1 ± 4.76e-1	4.97e+1 ± 4.64e-1
f_{21}	3.73e+2 ± 4.42e+1	4.00e+2 ± 0.00e+0	3.47e+2 ± 4.99e+1	3.47e+2 ± 4.99e+1	3.60e+2 ± 4.90e+1	3.73e+2 ± 4.42e+1	3.57e+2 ± 4.96e+1
	2.16e+4 ± 2.60e+3	1.78e+1 ± 3.20e+0	1.05e+2 ± 1.74e+1	1.15e+2 ± 2.42e+1	1.82e+1 ± 5.00e-1	1.84e+1 ± 7.12e-1	1.78e+1 ± 5.20e-1
f_{23}	3.06e+4 ± 4.89e+2	1.66e+4 ± 1.10e+3	1.48e+4 ± 7.01e+2	1.50e+4 ± 6.40e+2	1.21e+4 ± 7.78e+2	1.23e+4 ± 7.69e+2	1.22e+4 ± 6.51e+2
	2.55e+2 ± 1.31e+1	3.17e+2 ± 1.74e+1	2.35e+2 ± 6.51e+0	2.32e+2 ± 6.19e+0	2.45e+2 ± 5.98e+0	2.42e+2 ± 5.32e+0	2.39e+2 ± 5.98e+0

Tabela 6.1: Média e desvio padrão obtidos para os experimentos em 100 dimensões.

Função	DE	SHADE	L-SHADE	EBL-SHADE	A-SHADE	EBA-SHADE	EBA-SHADE-M
f_{25}	3.87e+2 ± 9.39e+0	4.47e+2 ± 1.85e+1	3.93e+2 ± 1.01e+1	3.89e+2 ± 1.21e+1	4.09e+2 ± 1.32e+1	4.00e+2 ± 9.44e+0	4.01e+2 ± 9.76e+0
	3.69e+2 ± 1.26e+1	4.43e+2 ± 1.69e+1	3.43e+2 ± 6.50e+0	3.38e+2 ± 5.38e+0	3.56e+2 ± 5.63e+0	3.50e+2 ± 7.41e+0	3.50e+2 ± 6.66e+0
f_{27}	1.07e+3 ± 1.94e+2	1.86e+3 ± 2.08e+2	6.91e+2 ± 9.84e+1	6.38e+2 ± 7.71e+1	8.36e+2 ± 9.32e+1	8.02e+2 ± 8.36e+1	7.77e+2 ± 8.32e+1
	3.41e+3 ± 1.04e+3	3.37e+3 ± 1.01e+3	2.51e+3 ± 1.97e+1	2.51e+3 ± 1.61e+1	2.54e+3 ± 2.09e+1	3.09e+3 ± 9.30e+2	2.53e+3 ± 1.89e+1
Destaques	4	7	14	15	16	19	16
B/S/W	19/7/2	17/9/2	11/12/5	11/10/7	0/28/0	-	4/24/0

Fonte: autoria própria.

Como mostra a tabela 6.1, a redução alternativa para o tamanho da população obteve resultados competitivos nos experimentos realizados, obtendo 16, 19 e 16 destaques com o A-SHADE, EB-A-SHADE e EB-A-SHADE-M, respectivamente. Estes valores foram superiores aos obtidos pelo L-SHADE e EB-L-SHADE, com 14 e 15 destaques, respectivamente. Sendo assim, para este conjunto de funções em 100 dimensões, o EB-A-SHADE se mostrou o melhor otimizador, e os três algoritmos com redução alternativa de NP foram superiores aos com redução linear.

Algo interessante de se notar é a relação entre o A-SHADE e o EB-A-SHADE. Embora o segundo tenha obtido 3 destaques a mais que o primeiro, a linha B/S/W mostra que eles foram estatisticamente equivalentes em todas as 28 funções. Isto porque as diferenças entre eles não foram grandes o suficiente para eles serem considerados estatisticamente diferentes quando avaliados de forma isolada, porém estas diferenças fizeram com que o EB-A-SHADE fosse equivalente ao melhor algoritmo em 3 funções (2, 4 e 13), enquanto o A-SHADE não foi equivalente a este melhor. Por exemplo, na função 4, o melhor algoritmo foi o EB-L-SHADE, que foi estatisticamente igual ao EB-A-SHADE, mas não ao A-SHADE.

Quanto ao uso da mutação não uniforme de Michalewicz, observou-se que a sua incorporação ao EB-A-SHADE degradou os seus resultados, piorando-os em 4 funções. Como será observado na tabela 6.3, porém, esta mudança causou uma mudança de comportamento interessante na diversidade genética da população, que será discutido adiante.

A tabela 6.2 apresenta o tempo total de execução de cada algoritmo, considerando uma execução por função. Os tempos são apresentados em segundos, e foram medidos sem a realização dos cálculos de diversidade e sem salvar o *fitness* a cada geração. Estas medidas de tempo foram realizadas tanto em execuções sequenciais quanto em paralelas, sendo que as execuções paralelas foram feitas com 7 *threads*. O *speedup* obtido pelo paralelismo é apresentado juntamente com os tempos na tabela 6.2.

Tabela 6.2: Tempos totais de uma execução de cada algoritmo, em segundos, de forma sequencial e paralela

<i>Threads</i>	DE	SHADE	L-SHADE	EBL-SHADE	A-SHADE	EBA-SHADE	EBA-SHADE-M
1	4204.724	4193.620	4184.085	4204.547	4176.030	4216.192	4233.149
7	894.373	913.284	899.730	902.446	938.828	942.921	992.340
<i>Speedup</i>	4.701	4.592	4.650	4.659	4.448	4.471	4.266

Fonte: autoria própria.

Como se pode ver na tabela 6.2, houve um *speedup* médio de 4.541 vezes com o paralelismo, o que mostra a importância desta prática. O número de *threads* utilizado na versão paralela foi igual a 7 pois é o número máximo de *threads* suportado pelo processador, menos uma reservada para o sistema. Com um número maior de *threads* suportadas, como por exemplo com o uso de GPU, existe o potencial para *speedups* ainda maiores.

Outro ponto a se notar na tabela é que todos os algoritmos apresentaram um tempo de execução aproximadamente igual. Isso mostra que a diferença de complexidade entre os algoritmos é pequena, e que o processo mais pesado, que todos têm em comum, é a avaliação dos indivíduos.

A tabela 6.3 apresenta os gráficos de convergência e diversidade para cada algoritmo em cada função, sendo que os valores mostrados são obtidos ao final de cada geração. São apresentados todos os algoritmos em um mesmo gráfico para cada função. Nestes gráficos, o eixo X representa o número de avaliações de função, e não as gerações, devido ao fato de que os algoritmos com diferentes dinâmicas para *NP* apresentam diferentes valores para o número total de gerações. Já o eixo Y representa o erro médio, para os gráficos de convergência, e a diversidade média, para os gráficos de diversidade.

Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.

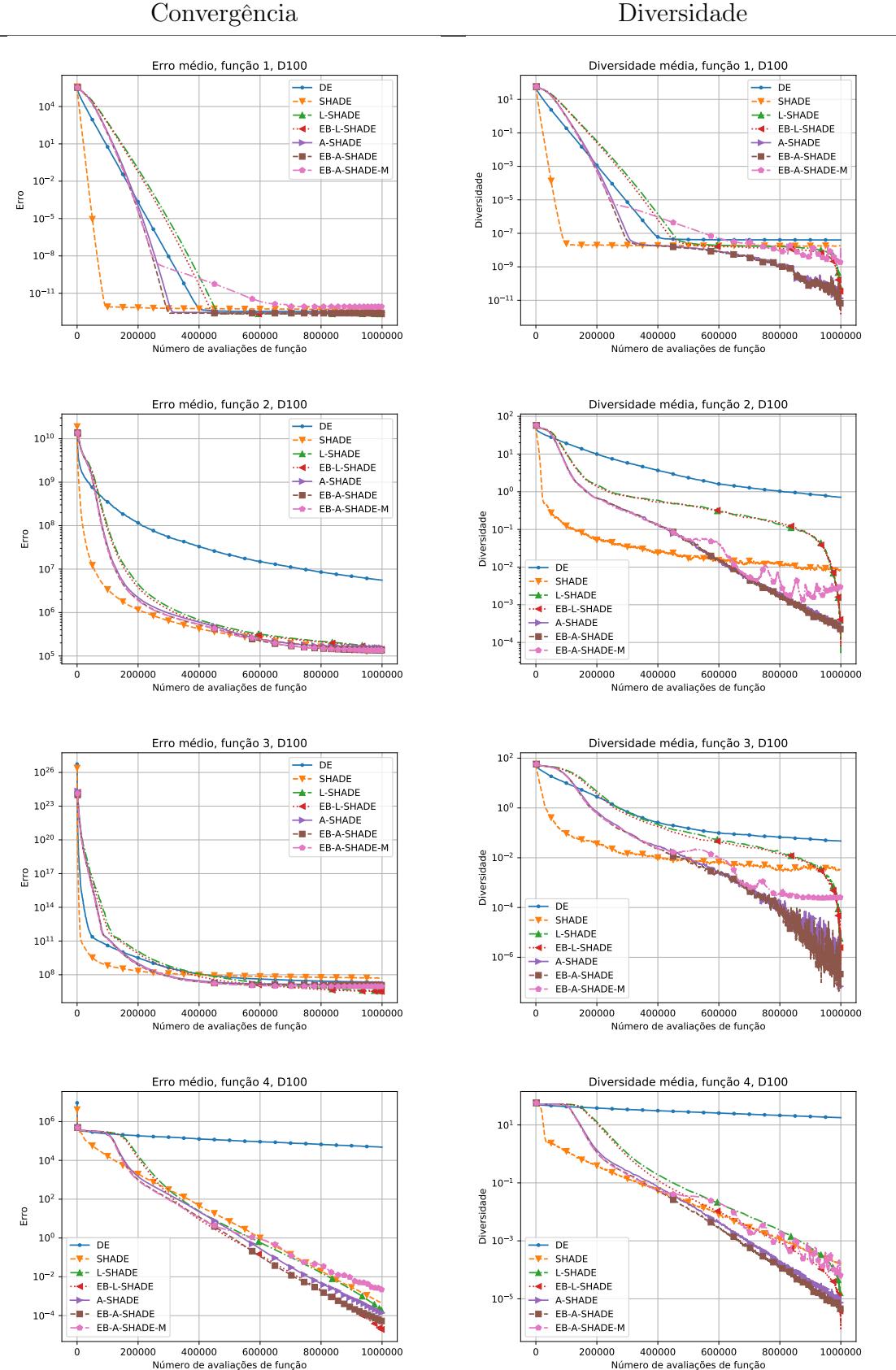


Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.

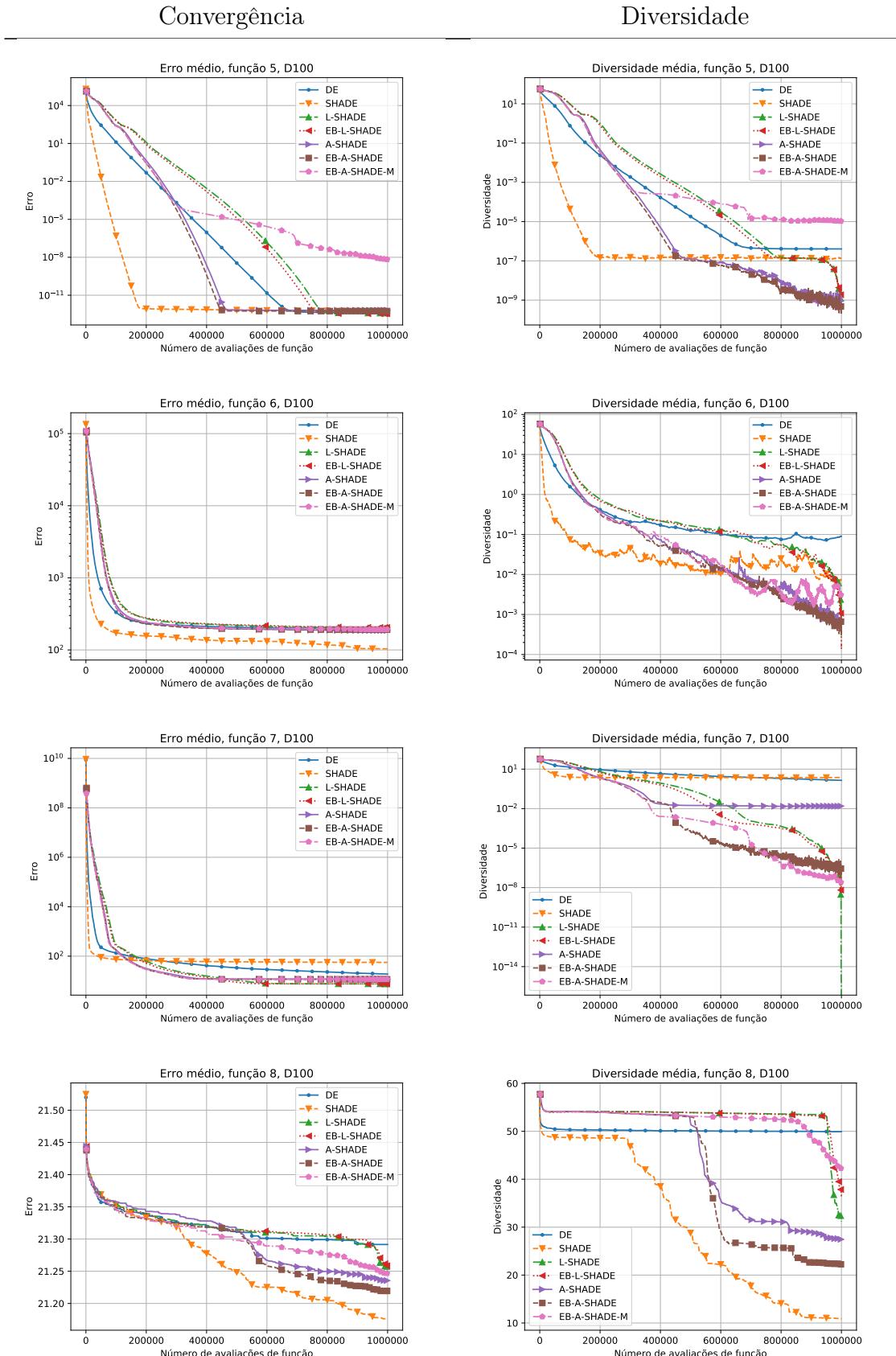


Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.

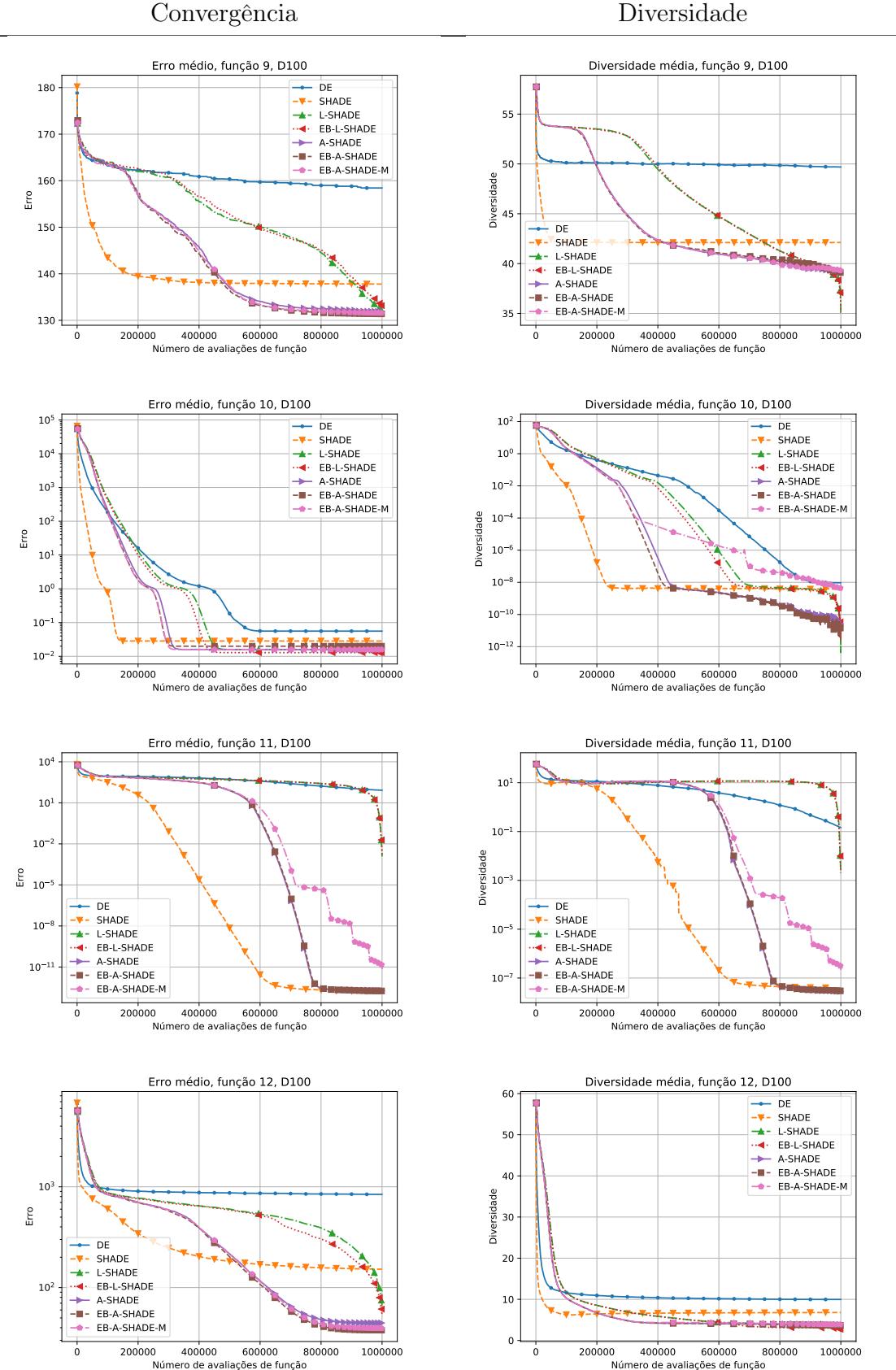


Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.

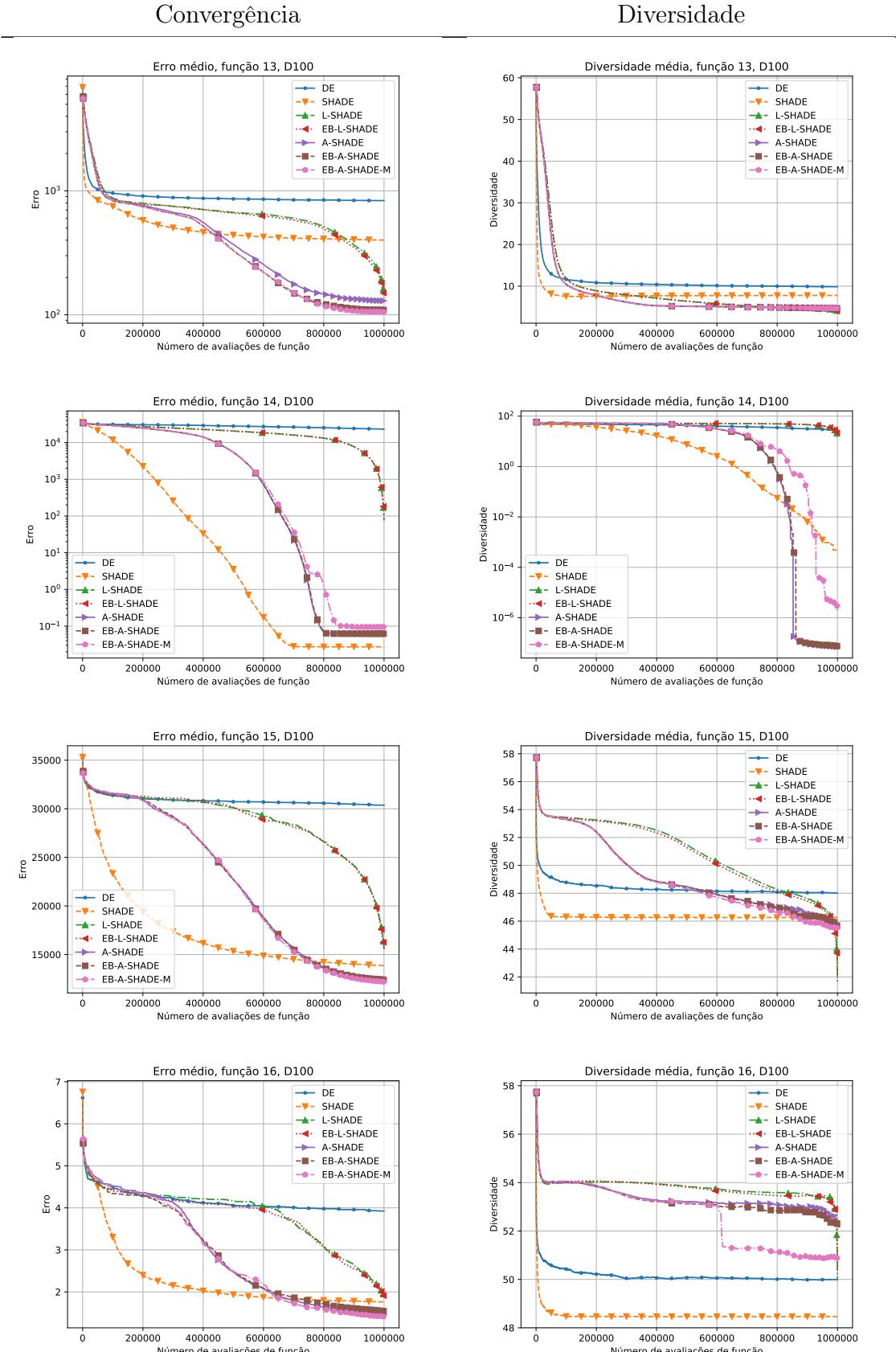


Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.

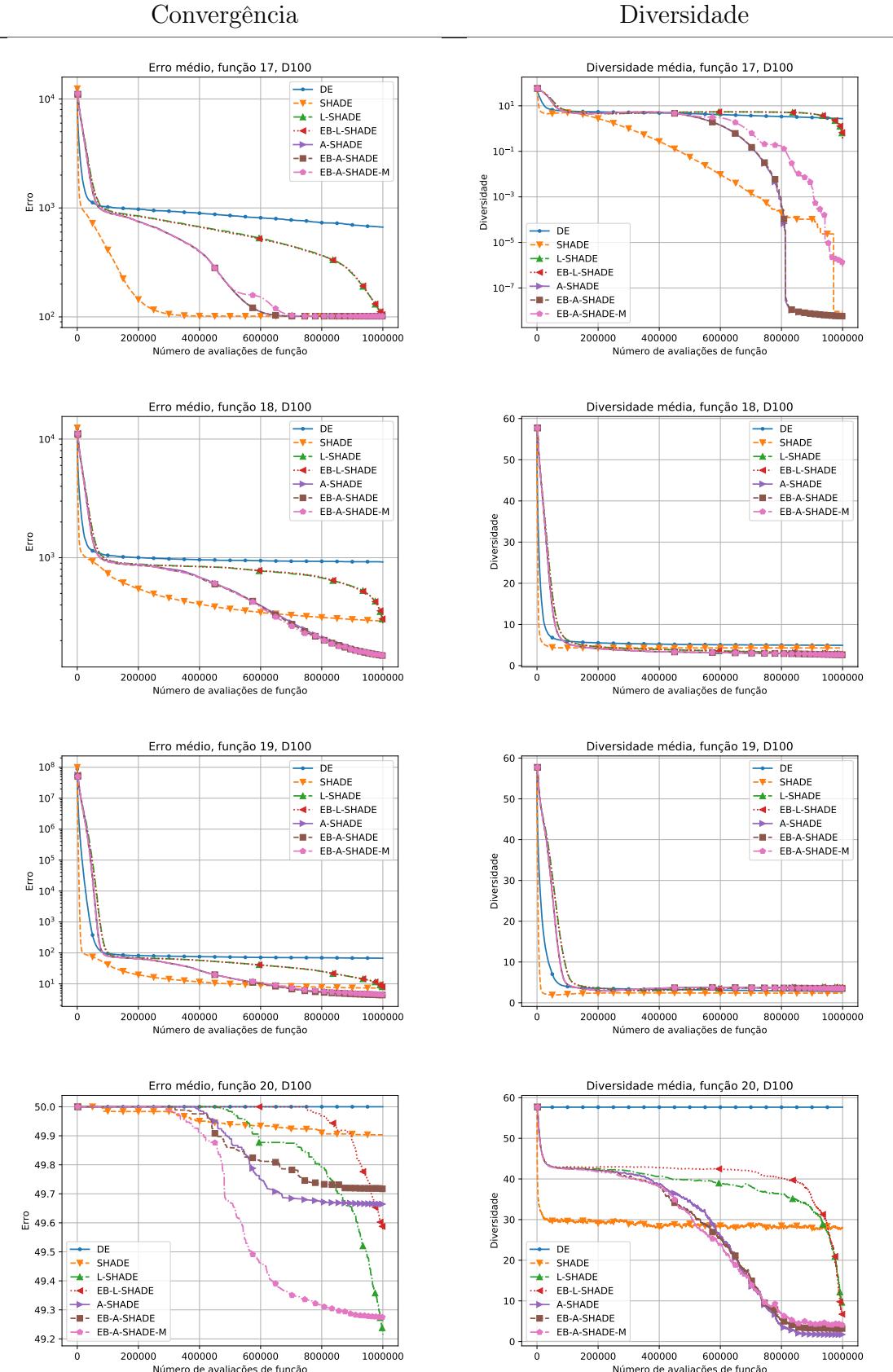


Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.

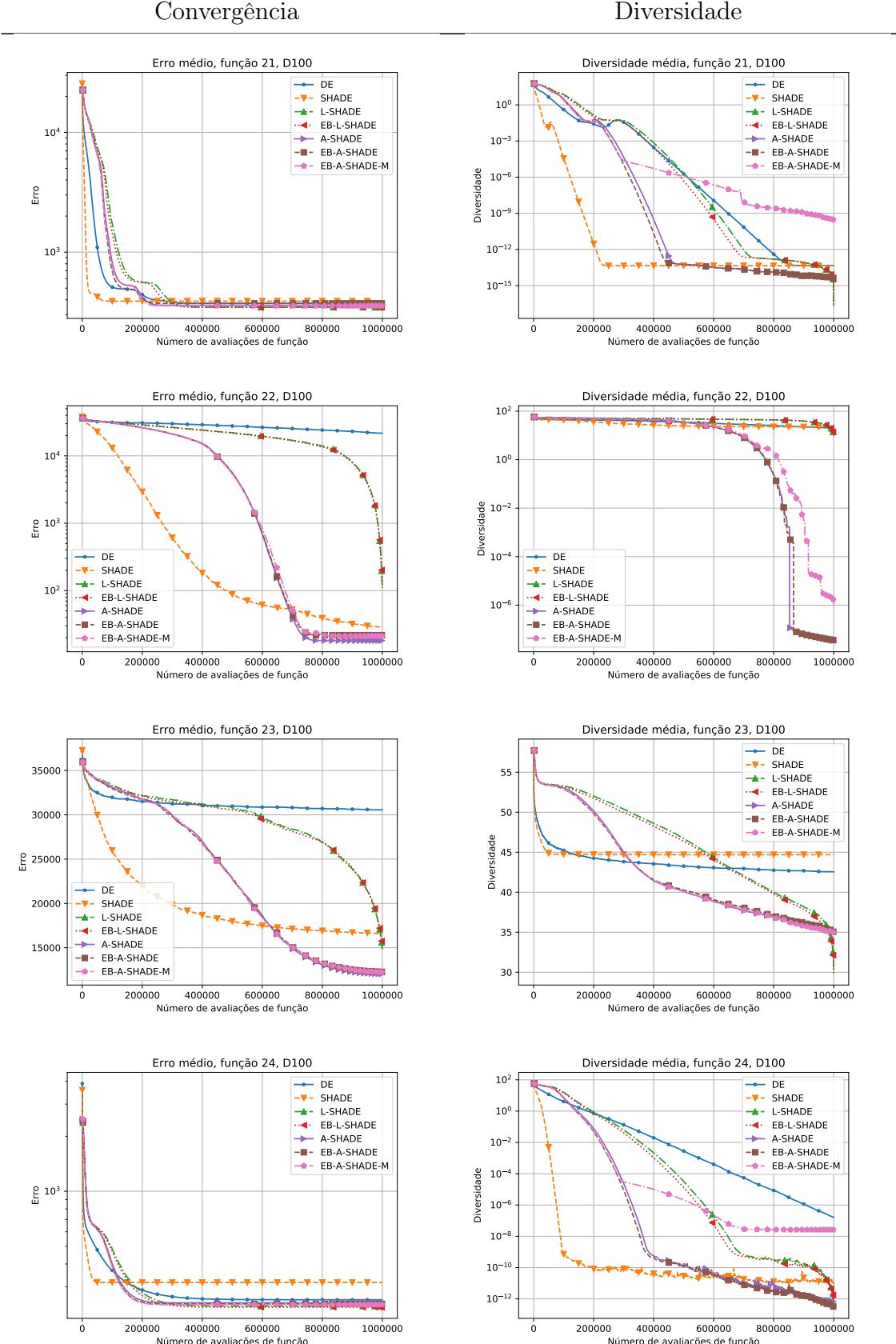
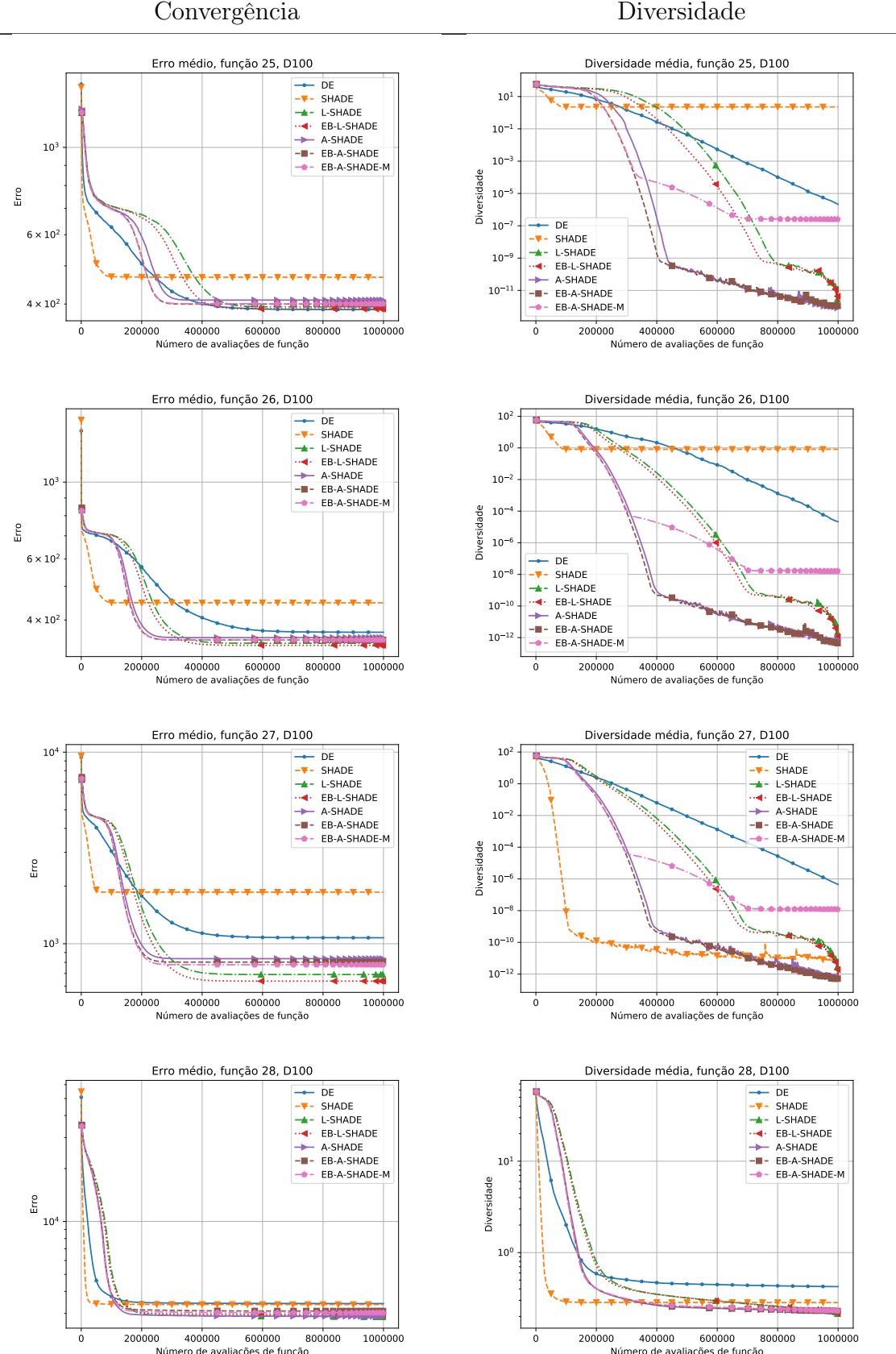


Tabela 6.3: Gráficos e convergência e diversidade para os experimentos executados.



O primeiro ponto que se pode notar nos gráficos é relacionado à velocidade de convergência. Tanto nos gráficos de convergência quanto de diversidade, percebe-se que o SHADE apresenta a convergência mais rápida para todas as funções exceto a número 8, sendo esta justamente uma das funções em que ele apresentou bons resultados. Na maior parte das funções, o SHADE obteve resultados inferiores às suas versões com população de tamanho dinâmico, e a convergência prematura é uma possível causa para esta observação. Porém vale notar que, nas funções 6 e 14, o SHADE apresentou resultados competitivos, mesmo com a convergência rápida, o que indica que há casos particulares em que a convergência rápida não é prejudicial.

Ainda quanto a velocidade de convergência, observa-se que a redução do tamanho da população reduz a velocidade de convergência do SHADE, e estes algoritmos apresentam melhores resultados. Um possível motivo para esta convergência mais lenta é a maior busca global realizada por estes algoritmos, devido ao maior tamanho de sua população no início da busca. Entre a redução linear e a redução alternativa, observa-se que, para todas as funções, os algoritmos com redução alternativa apresentaram uma convergência levemente mais rápida que os algoritmos com redução linear. Uma hipótese para isso é o fato de que eles transacionam de busca global para local mais rapidamente que os algoritmos com redução linear, pois eles passam menos gerações com populações grandes. Esta convergência mais rápida, no entanto, não foi suficiente para ser caracterizada como convergência prematura, o que é evidenciado pelos bons resultados obtidos por esses algoritmos.

De forma oposta ao SHADE, o L-SHADE e o EB-L-SHADE apresentam convergência bastante lenta em 14 das 28 funções. Nestas funções, o erro médio destes algoritmos diminui lentamente até que, após aproximadamente 80% das avaliações de função, ele começa a reduzir de forma abrupta. Este comportamento pode ser observado de forma bastante intensa nos gráficos das funções 11, 12, 13, 14, 15, 18, 20, 22 e 23, e de forma menos intensa, porém ainda presente, nas funções 8, 9, 16, 17 e 19. Dentre estas funções, somente em duas delas, números 9 e 20, esses dois algoritmos apresentaram destaque na tabela 6.1, mostrando que este comportamento é prejudicial à eficácia destes algoritmos. Em contrapartida, os algoritmos com redução alternativa de *NP* apresentaram uma convergência mais usual nestas funções, e obtiveram assim resultados superiores em comparação aos com redução linear.

Quanto ao uso da mutação current-to-ord_pbest, percebe-se que ambos EB-

L-SHADE e EB-A-SHADE não apresentaram degradação dos seus resultados, em relação às suas versões sem esta mutação, e nas funções 4, 5, 12, 13, 25, 26 e 27, apresentaram convergência levemente mais rápida, atingindo valores melhores para a média final. Embora estas diferenças, para a maior parte das funções, não sejam grandes o suficiente para haver diferenças estatísticas quando os algoritmos são analisados de forma isolada, elas se destacam quando múltiplos algoritmos são analisados simultaneamente. Este fenômeno pode ser observado com o EB-A-SHADE, que recebeu mais destaque que o A-SHADE na tabela 6.1, embora eles tenham sido estatisticamente equivalentes em todas as funções.

Analizando a diversidade dos algoritmos, pode-se observar que, para 15 das 28 funções, todos os algoritmos convergem para uma região pequena, atingindo valores de diversidade entre 10^{-4} e 10^{-14} . Isto significa, porém, que em 13 funções ao menos um dos algoritmos teve dificuldades para convergir. Nos parágrafos seguintes, discute-se quais foram estas funções e que análises podem ser feitas a partir destas observações.

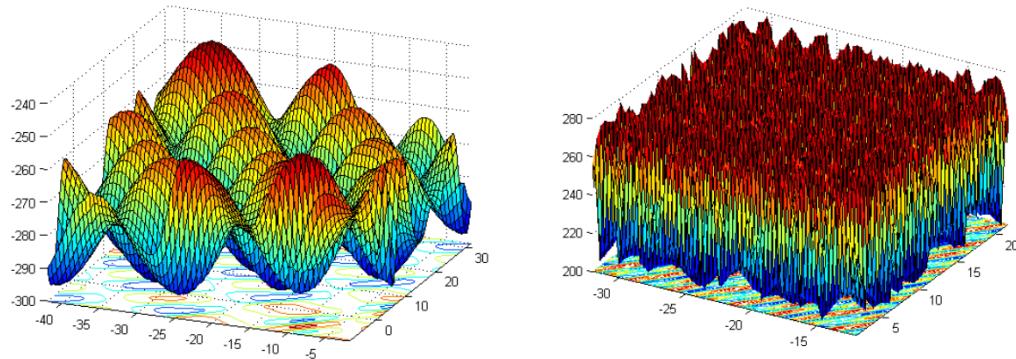
Na função 4, o DE foi o único a não convergir, o que é refletido no seu valor de erro médio obtido. Na função 7, o DE e o SHADE mantiveram valores altos para a sua diversidade, e também apresentaram resultados inferiores em relação aos outros algoritmos. É interessante notar que, nessa função, estes dois algoritmos foram os que apresentaram a convergência mais rápida no início, evidenciando novamente o problema da convergência prematura.

Nas funções 8, 9, 12, 13, 15, 16, 18, 19, 20, 23 e 28, nenhum dos algoritmos foi capaz de reduzir a sua diversidade para valores baixos. Estas funções apresentam duas características em comum, em diferentes proporções, que podem explicar estes resultados: a presença de irregularidades, com um grande número de ótimos locais, e uma variação pequena entre os valores de seus objetivos entre estes diferentes ótimos locais.

Na figura 6.1 são apresentados os gráficos das funções 12 e 16, em duas dimensões (LIANG et al., 2013). Em ambas estas funções, pode-se perceber a presença de um grande número de ótimos locais com valores semelhantes para a função objetivo. Em situações como estas, os algoritmos têm dificuldade para convergir a sua população em um único ótimo. As funções 8, 9, 13, 15, 20 e 23 apresentam também este tipo de comportamento.

Já na figura 6.2, são apresentados os gráficos das funções 19 e 28 em duas dimensões (LIANG et al., 2013). Estas funções apresentam outro fator que interfere na

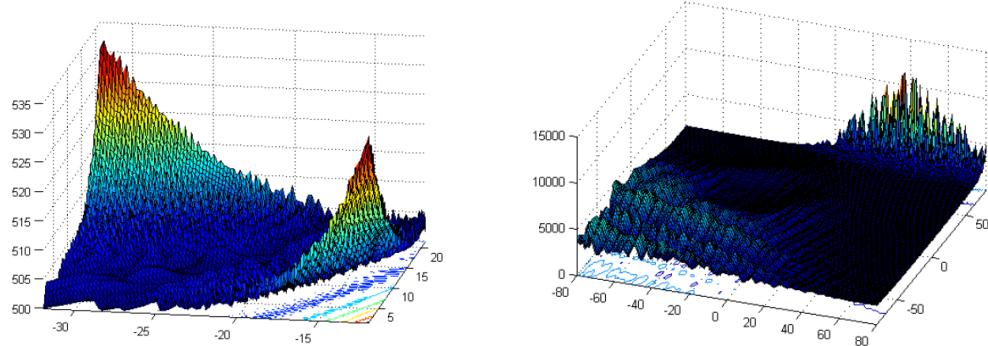
Figura 6.1: Gráficos das funções 12 e 16, respectivamente



Fonte: (LIANG et al., 2013)

diversidade genética dos algoritmos, que é a presença de vales irregulares em seu espaço de busca. As irregularidades destes vales criam um grande número de ótimos locais, porém sem atratores fortes, pois esta região apresenta uma variação pequena no valor da função objetivo. Isto resulta em a população permanecer espalhada dentro deste espaço. A função 18 apresenta também este tipo de comportamento.

Figura 6.2: Gráficos das funções 19 e 28, respectivamente



Fonte: (LIANG et al., 2013)

Com a sua população espalhada pelo espaço de busca, um algoritmo realiza pouca busca local, não intensificando assim os seus resultados obtidos. Devido a isto, os resultados observados nestas funções mostram que, para problemas que apresentem estes tipos de características, os algoritmos estudados podem não ser os mais adequados.

Outro ponto que merece ser citado é a influência que a mutação não uniforme de Michalewicz teve sobre o EB-A-SHADE. Pelos resultados apresentados na tabela 6.1, entende-se que esta rotina deteriorou a eficácia do EB-A-SHADE. Apesar disto, percebe-se também que esta rotina permitiu que o algoritmo mantivesse valores maiores de diversi-

dade ao longo da execução, mas sem impedir a sua convergência. Isto é interessante pois permite um melhor equilíbrio entre busca local e global ao algoritmo. Deste modo, embora esta rotina não tenha sido positiva para o EB-A-SHADE, ela é algo a se considerar para desenvolvimentos futuros, especialmente para algoritmos que sofrem com convergência prematura.

6.3 Considerações Parciais

Neste capítulo, foram apresentados os experimentos realizados, resultados obtidos e análises destes resultados.

Os resultados obtidos mostraram que, para o conjunto de *benchmark* avaliado, os melhores resultados foram obtidos pelos algoritmos com redução alternativa de população, em especial o EB-A-SHADE, que obteve destaque em 19 das 28 funções.

Pôde-se também observar a importância do paralelismo na realização dos experimentos, sendo obtido um *speedup* médio de 4.5 vezes de 1 para 7 *threads*, entre todos os algoritmos.

Foram apresentados os gráficos de convergência e diversidade dos 7 algoritmos em cada uma das 28 funções avaliadas. As análises destes gráficos mostraram que os diferentes algoritmos apresentam diferentes comportamentos em relação a velocidade de convergência. Identificou-se que a convergência prematura foi um problema apresentado pelo SHADE e que afetou negativamente seus resultados, enquanto as suas versões com redução de população foram capazes de convergir mais lentamente e para valores melhores.

7 Considerações Finais

A parametrização é um problema intrínseco a meta-heurísticas de otimização, sendo que identificar um conjunto ótimo de parâmetros para uma dada situação é um problema de otimização dentro de um problema de otimização. Para evitar este problema, múltiplos métodos de parametrização *online* já foram propostos na literatura.

A Evolução Diferencial é uma técnica de otimização contínua, pertencente à família dos algoritmos evolutivos, que apresenta um grande potencial para parametrização *online*, com múltiplos algoritmos já propostos na literatura para este fim. Este trabalho apresentou uma revisão dos principais algoritmos de Evolução Diferencial adaptativa encontrados na literatura, com destaque ao SHADE, L-SHADE e EB-L-SHADE, que foram implementados e analisados, além do DE canônico. Também foram propostas três modificações para o L-SHADE, resultando nos algoritmos A-SHADE, EB-A-SHADE e EB-A-SHADE-M. Estes sete algoritmos foram aplicados ao conjunto de *benchmark* do CEC-2013, em 100 dimensões, e os resultados obtidos foram comparados e analisados.

Os resultados obtidos mostraram que a redução alternativa do tamanho da população foi significativamente positiva, com os três algoritmos que utilizam esta rotina obtendo os maiores números de destaques entre as funções experimentadas e com o número de dimensões utilizado.

Quanto à mutação current-to-ord_pbest, notou-se que o seu uso causou pouco impacto quando o algoritmo que a usa é comparado diretamente à sua versão original. Este pequeno impacto, porém, se destaca quando o contexto é expandido e todos os algoritmos são comparados simultaneamente. Já a aplicação da mutação não uniforme de Michalewicz se mostrou negativa quando aplicada ao EB-A-SHADE. O impacto positivo que esta rotina tem na diversidade da população, porém, faz com que ela tenha potencial para ser estudada em algoritmos que apresentem convergência prematura.

Foi observado que todos os algoritmos adaptativos avaliados apresentaram eficácia superior à do DE canônico nestes experimentos. Isto comprova que, além de remover a necessidade de se definir todos os parâmetros manualmente, a parametrização *online* pode também obter aprimorar a qualidade das soluções obtidas por algoritmos que utilizam

zam parametrização *offline*.

Outro ponto que vale ser mencionado é o uso do paralelismo durante os experimentos. Obteve-se um *speedup* de 4.5 vezes por meio da paralelização das avaliações de função, com o uso de 7 *threads* em um sistema com 4 núcleos físicos e 8 núcleos virtuais. Isto mostra que pode-se ganhar um tempo significativo com os experimentos por meio do paralelismo.

Para trabalhos futuros, algo que deve ser feito é a expansão dos experimentos para além de 100 dimensões, com o objetivo de verificar se a relação observada entre os algoritmos se mantém para outras dimensionalidades. Outro ponto que merece ser avaliado é a implementação destes algoritmos em GPU, o que pode levar a *speedups* ainda maiores. Com experimentos mais rápidos, pode-se inclusive avaliar como os algoritmos desempenham quando o número de avaliações de função disponível é maior que os utilizados neste trabalho.

Por fim, outro possível trabalho futuro é a aplicação dos algoritmos produzidos, em especial o EB-A-SHADE, a um problema real, sendo assim possível avaliar estes algoritmos em um contexto que vá além das funções de *benchmark*, possivelmente auxiliando no melhor entendimento de problemas ainda não totalmente compreendidos.

Referências

- ABBASS, H. A. The self-adaptive pareto differential evolution algorithm. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, Honolulu, HI, USA, v. 1, p. 831–836 vol.1, May 2002.
- ALI, M. M.; TÖRN, A. Population set-based global optimization algorithms: Some modifications and numerical studies. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 31, n. 10, p. 1703–1725, set. 2004. ISSN 0305-0548. Disponível em: <[http://dx.doi.org/10.1016/S0305-0548\(03\)00116-3](http://dx.doi.org/10.1016/S0305-0548(03)00116-3)>.
- ANDRÉ, L. *Modelagem de relações simbióticas em um ecossistema computacional para otimização*. Dissertação (Mestrado) — UDESC - Universidade do Estado de Santa Catarina, 8 2015. Disponível em: <<http://tede.udesc.br/tede/tede/1763>>.
- ANDRÉ, L.; PARPINELLI, R. S. Tutorial sobre o uso de técnicas para controle de parâmetros em algoritmos de inteligência de enxame e computação evolutiva. *Revista de Informática Teórica e Aplicada*, v. 21, n. 2, p. 90–135, 2014. Disponível em: <<https://doi.org/10.22456/2175-2745.48184>>.
- AWAD, N. H.; ALI, M. Z.; SUGANTHAN, P. N. Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems. *2017 IEEE Congress on Evolutionary Computation (CEC)*, p. 372–379, June 2017.
- AWAD, N. H. et al. An ensemble sinusoidal parameter adaptation incorporated with l-shade for solving cec2014 benchmark problems. *2016 IEEE Congress on Evolutionary Computation (CEC)*, p. 2958–2965, July 2016.
- BI, X.-J.; XIAO, J. Classification-based self-adaptive differential evolution with fast and reliable convergence performance. *Soft Computing*, v. 15, n. 8, p. 1581–1599, Aug 2011. ISSN 1433-7479. Disponível em: <<https://doi.org/10.1007/s00500-010-0689-5>>.
- BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information Sciences*, v. 237, p. 82 – 117, 2013. ISSN 0020-0255. Prediction, Control and Diagnosis using Advanced Neural Computations. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0020025513001588>>.
- BRABAZON, A.; O’NEILL, M.; MCGARRAGHY, S. *Natural Computing Algorithms*. 1. ed. Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2015. ISBN 978-3-662-43631-8.
- BREST, J. et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, v. 10, n. 6, p. 646–657, Dec 2006. ISSN 1089-778X.
- BREST, J.; MAUČEC, M. S. Population size reduction for the differential evolution algorithm. *Applied Intelligence*, v. 29, n. 3, p. 228–247, Dec 2008. ISSN 1573-7497. Disponível em: <<https://doi.org/10.1007/s10489-007-0091-x>>.

- BREST, J.; MAUČEC, M. S. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, v. 15, n. 11, p. 2157–2174, Nov 2011. ISSN 1433-7479. Disponível em: <<https://doi.org/10.1007/s00500-010-0644-5>>.
- BREST, J.; MAUČEC, M. S.; BOŠKOVIĆ, B. il-shade: Improved l-shade algorithm for single objective real-parameter optimization. *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, BC, p. 1188–1195, July 2016.
- DEAP Project. *Benchmarks*. 2014. Disponível em: <<http://deap.gel.ulaval.ca/doc/0.9-api/benchmarks.html>>.
- DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. *IEEE Computational Intelligence Magazine*, v. 1, n. 4, p. 28–39, Nov 2006. ISSN 1556-603X.
- DUNN, O. J. Multiple comparisons among means. *Journal of the American Statistical Association*, [American Statistical Association, Taylor & Francis, Ltd.], v. 56, n. 293, p. 52–64, 1961. ISSN 01621459.
- EIBEN, A. E.; HINTERDING, R.; MICHALEWICZ, Z. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 2, p. 124–141, July 1999. ISSN 1089-778X.
- EIBEN, A. E.; RUTTKAT, Z. Constraint-satisfaction problems. In: JONG, K. D.; FOGEL, L.; SCHWEFEL, H.-P. (Ed.). *Handbook of Evolutionary Computation*. United Kingdom: IOP Publishing Ltd and Oxford University Press, 1997. cap. 5.7.
- EPITROPAKIS, M. G.; PLAGIANAKOS, V. P.; VRAHATIS, M. N. Evolutionary adaptation of the differential evolution control parameters. *2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, p. 1359–1366, May 2009. ISSN 1089-778X.
- GONG, W. et al. Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. *Information Sciences*, v. 181, n. 24, p. 5364 – 5386, 2011. ISSN 0020-0255. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0020025511004117>>.
- GOSH, A. et al. A modified differential evolution with distance-based selection for continuous optimization in presence of noise. *IEEE Access*, v. 5, p. 26944–26964, 2017. ISSN 2169-3536.
- GUPTA, D.; GHAFIR, S. An overview of methods maintaining diversity in genetic algorithms. *International Journal of Emerging Technology and Advanced Engineering*, v. 2, n. 5, p. 256–60, 2012. ISSN 2250-2459.
- HOLLAND, J. H. Genetic algorithms and adaptation. In: _____. *Adaptive Control of Ill-Defined Systems*. Boston, MA: Springer US, 1984. p. 317–333. ISBN 978-1-4684-8941-5. Disponível em: <https://doi.org/10.1007/978-1-4684-8941-5_21>.
- HUANG, Z.; CHEN, Y. An improved differential evolution algorithm based on adaptive parameter. *Journal of Control Science and Engineering*, 2013. Disponível em: <<http://dx.doi.org/10.1155/2013/462706>>.

- ISLAM, S. M. et al. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 42, n. 2, p. 482–500, April 2012. ISSN 1083-4419.
- JAMIL, M.; YANG, X. A literature survey of benchmark functions for global optimization problems. *CoRR*, abs/1308.4008, 2013. Disponível em: <<http://arxiv.org/abs/1308-4008>>.
- KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, v. 39, n. 3, p. 459–471, Nov 2007. ISSN 1573-2916.
- KENNEDY, J.; EBERHART, R. Particle swarm optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Perth, Australia, v. 4, p. 1942–1948, 1995.
- KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0-262-11170-5.
- KUTSCHERA, U.; NIKLAS, K. J. The modern theory of biological evolution: an expanded synthesis. *Naturwissenschaften*, v. 91, n. 6, p. 255–276, Jun 2004. ISSN 1432-1904. Disponível em: <<https://doi.org/10.1007/s00114-004-0515-y>>.
- LIANG, J. J. et al. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*. Cancún, Mexico, 2013.
- LIU, J.; LAMPINEN, J. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, v. 9, n. 6, p. 448–462, Jun 2005. ISSN 1433-7479. Disponível em: <<https://doi.org/10.1007/s00500-004-0363-x>>.
- MALLIPEDDI, R. et al. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, v. 11, n. 2, p. 1679 – 1696, 2011. ISSN 1568-4946. The Impact of Soft Computing for the Progress of Artificial Intelligence. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1568494610001043>>.
- MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, ACM, New York, NY, USA, v. 8, n. 1, p. 3–30, jan. 1998. ISSN 1049-3301. Disponível em: <<http://doi.acm.org/10.1145/272991.272995>>.
- MAUČEC, M. S. et al. Improved differential evolution for large-scale black-box optimization. *IEEE Access*, v. 6, p. 29516–29531, 2018. ISSN 2169-3536.
- MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN 3-540-60676-9.
- MOHAMED, A. W.; HADI, A. A.; JAMBI, K. M. Novel mutation strategy for enhancing shade and lshade algorithms for global numerical optimization. *Swarm and Evolutionary Computation*, 2018. ISSN 2210-6502. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2210650218301597>>.
- MOLINA, D.; LATORRE, A.; HERRERA, F. Shade with iterative local search for large-scale global optimization. *IEEE Congress on Evolutionary Computation*, Rio de Janeiro, Brasil, p. 1252–1259, 2018.

- MORRISON, R. W.; JONG, K. A. D. Measurement of population diversity. *Artificial Evolution*, Springer Berlin Heidelberg, Berlin, Heidelberg, p. 31–41, 2002.
- OMRAN, M. G. H.; SALMAN, A.; ENGELBRECHT, A. P. Self-adaptive differential evolution. In: HAO, Y. et al. (Ed.). *Computational Intelligence and Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 192–199. ISBN 978-3-540-31599-5.
- OpenMP Architecture Review Board. *OpenMP Application Program Interface Version 3.0*. 2008. Disponível em: <<http://www.openmp.org/mp-documents/spec30.pdf>>.
- OSTERTAGOVA, E.; OSTERTAG, O.; KOVÁČ, J. Methodology and application of the kruskal-wallis test. *Applied Mechanics and Materials*, v. 611, p. 115–120, 08 2014.
- PARPINELLI, R. S. et al. A review of techniques for on-line control of parameters in swarm intelligence and evolutionary computation algorithms. *International Journal of Bio-Inspired Computation*, 2018.
- PIOTROWSKI, A. P.; NAPIORKOWSKI, J. J. Some metaheuristics should be simplified. *Information Sciences*, v. 427, p. 32 – 62, 2018. ISSN 0020-0255. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0020025517310332>>.
- QIN, A. K.; HUANG, V. L.; SUGANTHAN, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 2, p. 398–417, April 2009. ISSN 1089-778X.
- QIN, A. K.; SUGANTHAN, P. N. Self-adaptive differential evolution algorithm for numerical optimization. *2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, v. 2, p. 1785–1791 Vol. 2, Sept 2005. ISSN 1089-778X.
- REYNOLDS, R. G. New ideas in optimization. In: CORNE, D. et al. (Ed.). Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999. cap. Cultural Algorithms: Theory and Applications, p. 367–378. ISBN 0-07-709506-5. Disponível em: <<http://dl.acm.org/citation.cfm?id=329055.329089>>.
- SENKERIK, R. et al. Differential evolution and chaotic series. *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, Maribor, Slovenia, p. 1–5, June 2018. ISSN 2157-8702.
- SKANDEROVA, L. Influence of control parameters adaptation on spread of positive genomes within populations of selected differential evolution algorithms. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, USA, p. 1–8, Nov 2017.
- STORN, R.; PRICE, K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, v. 11, n. 4, p. 341–359, Dec 1997. ISSN 1573-2916. Disponível em: <<https://doi.org/10.1023/A:1008202821328>>.
- TANABE, R.; FUKUNAGA, A. Success-history based parameter adaptation for differential evolution. *2013 IEEE Congress on Evolutionary Computation*, Cancún, Mexico, p. 71–78, June 2013. ISSN 1089-778X.
- TANABE, R.; FUKUNAGA, A. Tuning differential evolution for cheap, medium, and expensive computational budgets. *2015 IEEE Congress on Evolutionary Computation (CEC)*, Sendai, Japan, p. 2018–2025, May 2015. ISSN 1089-778X.

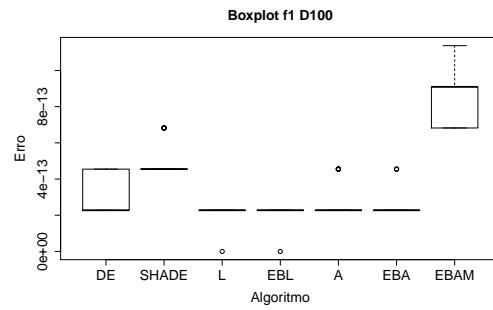
- TANABE, R.; FUKUNAGA, A. How far are we from an optimal, adaptive de? In: HANDL, J. et al. (Ed.). *Parallel Problem Solving from Nature – PPSN XIV*. Edinburgh, UK: Springer International Publishing, 2016. p. 145–155. ISBN 978-3-319-45823-6.
- TANABE, R.; FUKUNAGA, A. S. Improving the search performance of shade using linear population size reduction. *2014 IEEE Congress on Evolutionary Computation (CEC)*, Beijing, China, p. 1658–1665, July 2014. ISSN 1089-778X.
- TEO, J. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, v. 10, n. 8, p. 673–686, Jun 2006. ISSN 1433-7479. Disponível em: <<https://doi.org/10.1007/s00500-005-0537-1>>.
- WANG, Y.; CAI, Z.; ZHANG, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, v. 15, n. 1, p. 55–66, Feb 2011. ISSN 1089-778X.
- ZHANG, J. et al. A survey on algorithm adaptation in evolutionary computation. *Frontiers of Electrical and Electronic Engineering*, v. 7, n. 1, p. 16–31, Mar 2012. ISSN 1673-3584. Disponível em: <<https://doi.org/10.1007/s11460-012-0192-0>>.
- ZHANG, J.; SANDERSON, A. C. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 5, p. 945–958, Oct 2009. ISSN 1089-778X.
- ZHAO, S.-Z.; SUGANTHAN, P. N.; DAS, S. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing*, v. 15, n. 11, p. 2175–2185, Nov 2011. ISSN 1433-7479. Disponível em: <<https://doi.org/10.1007/s00500-010-0645-4>>.

A Apêndice: Boxplots

Este apêndice apresenta os boxplots obtidos pelos experimentos com os sete algoritmos avaliados para cada uma das funções.

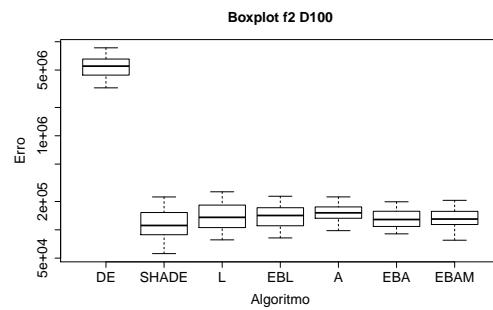
Para os algoritmos L-SHADE, EB-L-SHADE, A-SHADE, EB-A-SHADE e EB-A-SHADE-M, a palavra "SHADE" foi omitida na legenda dos gráficos, devido ao espaço nas legendas. Estes algoritmos são referidos como "L", "EBL", "A", "EBA" e "EBAM", respectivamente.

Figura A.1: Boxplots dos experimentos com a função 1.



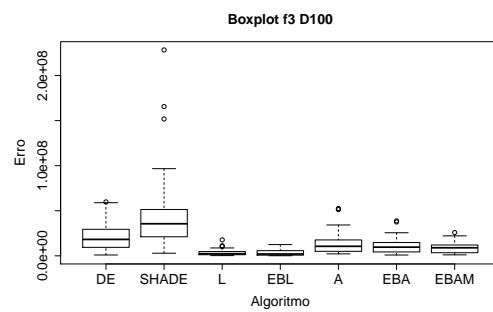
Fonte: autoria própria.

Figura A.2: Boxplots dos experimentos com a função 2.



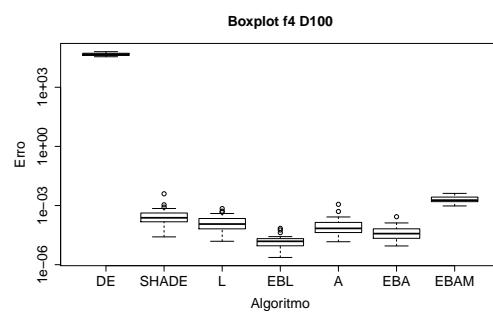
Fonte: autoria própria.

Figura A.3: Boxplots dos experimentos com a função 3.



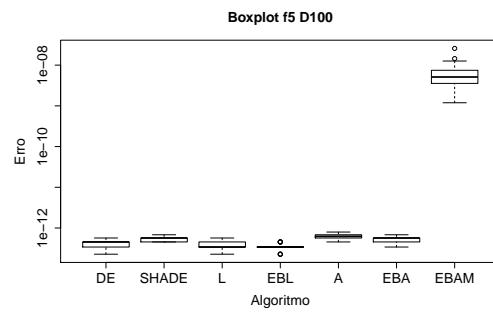
Fonte: autoria própria.

Figura A.4: Boxplots dos experimentos com a função 4.



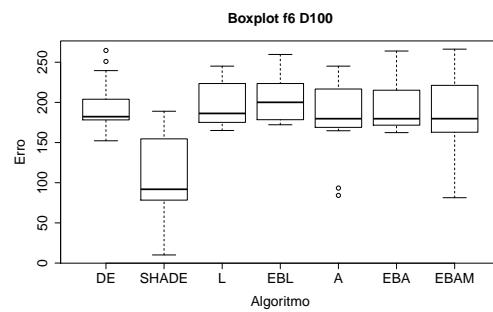
Fonte: autoria própria.

Figura A.5: Boxplots dos experimentos com a função 5.



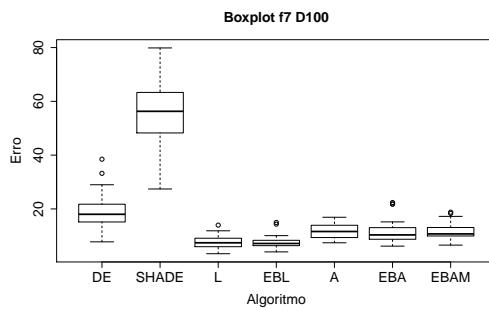
Fonte: autoria própria.

Figura A.6: Boxplots dos experimentos com a função 6.



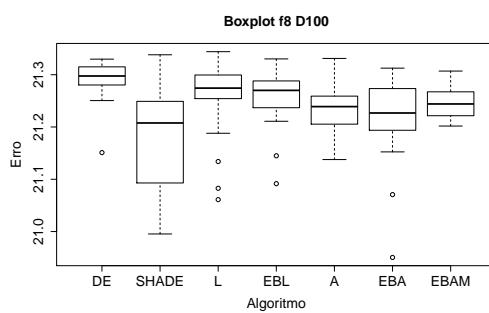
Fonte: autoria própria.

Figura A.7: Boxplots dos experimentos com a função 7.



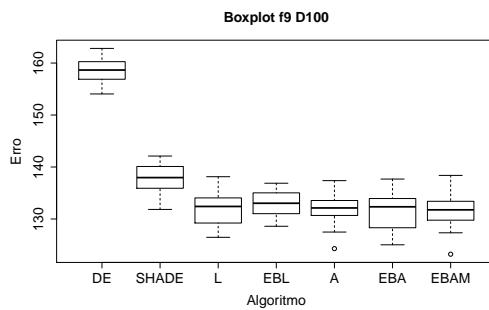
Fonte: autoria própria.

Figura A.8: Boxplots dos experimentos com a função 8.



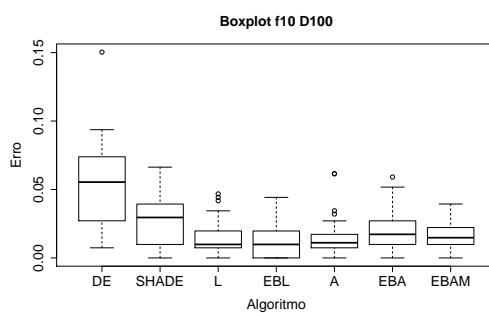
Fonte: autoria própria.

Figura A.9: Boxplots dos experimentos com a função 9.



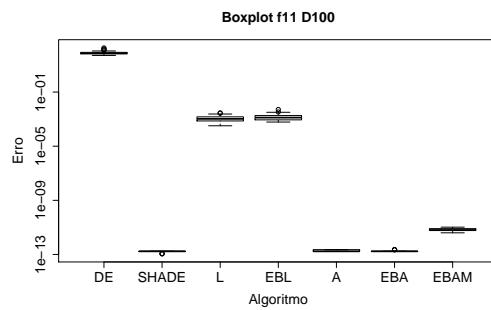
Fonte: autoria própria.

Figura A.10: Boxplots dos experimentos com a função 10.



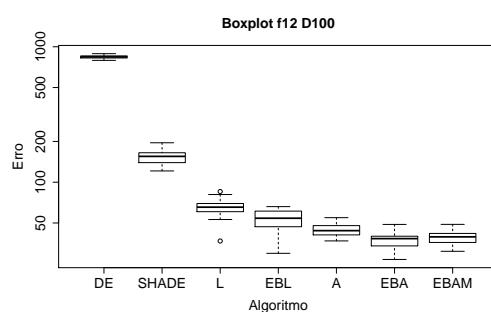
Fonte: autoria própria.

Figura A.11: Boxplots dos experimentos com a função 11.



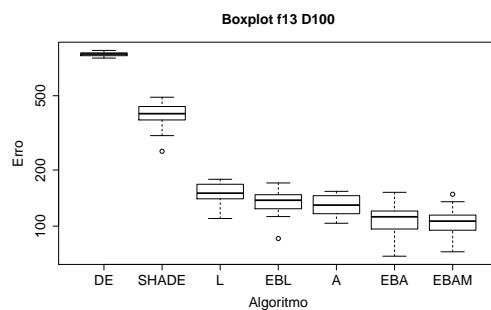
Fonte: autoria própria.

Figura A.12: Boxplots dos experimentos com a função 12.



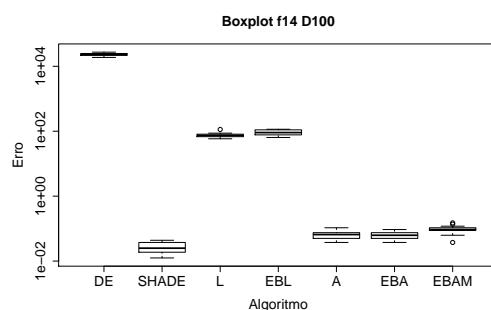
Fonte: autoria própria.

Figura A.13: Boxplots dos experimentos com a função 13.



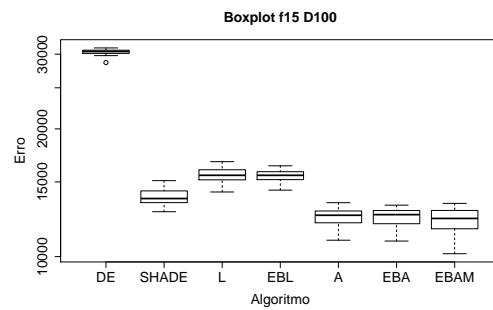
Fonte: autoria própria.

Figura A.14: Boxplots dos experimentos com a função 14.



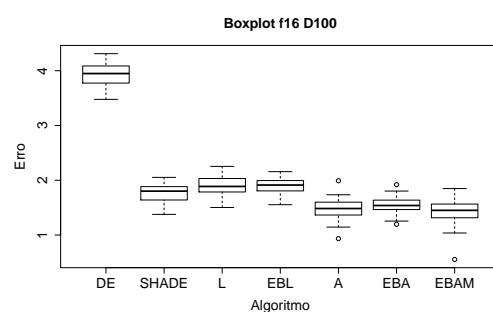
Fonte: autoria própria.

Figura A.15: Boxplots dos experimentos com a função 15.



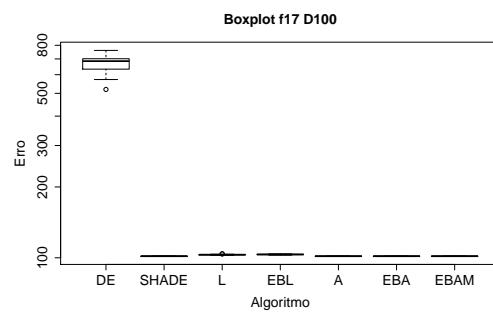
Fonte: autoria própria.

Figura A.16: Boxplots dos experimentos com a função 16.



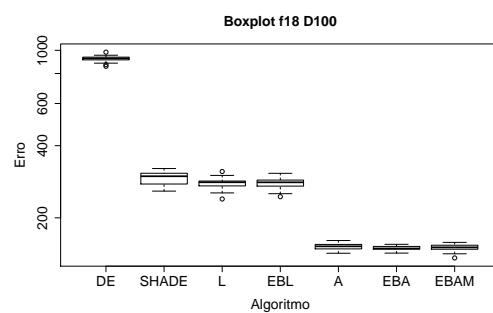
Fonte: autoria própria.

Figura A.17: Boxplots dos experimentos com a função 17.



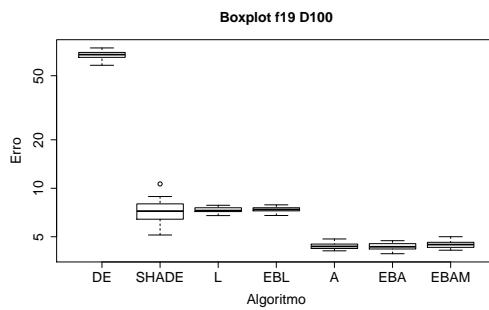
Fonte: autoria própria.

Figura A.18: Boxplots dos experimentos com a função 18.



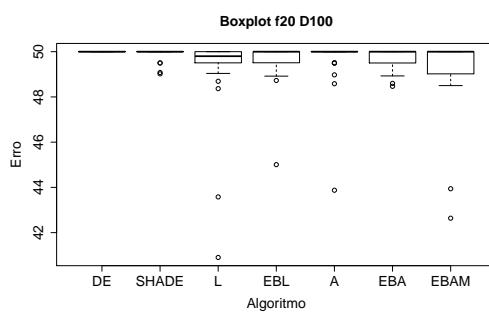
Fonte: autoria própria.

Figura A.19: Boxplots dos experimentos com a função 19.



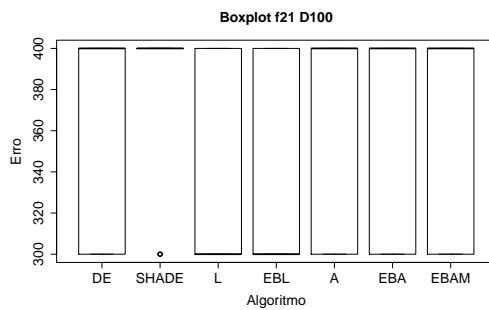
Fonte: autoria própria.

Figura A.20: Boxplots dos experimentos com a função 20.



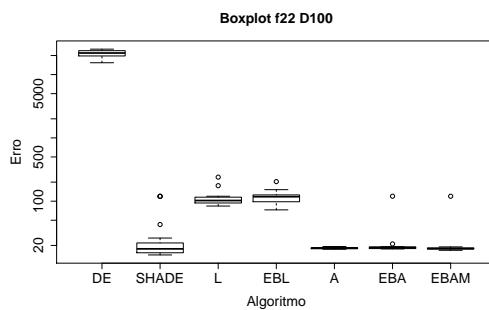
Fonte: autoria própria.

Figura A.21: Boxplots dos experimentos com a função 21.



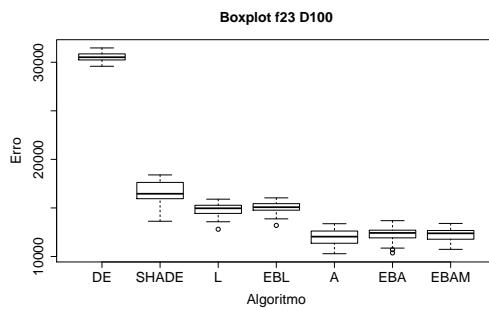
Fonte: autoria própria.

Figura A.22: Boxplots dos experimentos com a função 22.



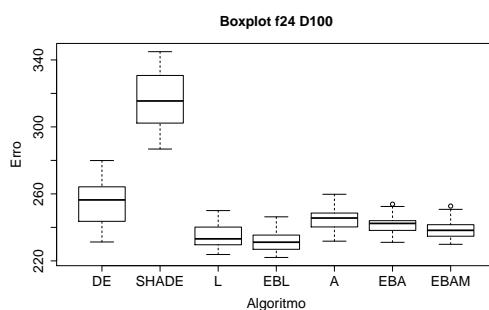
Fonte: autoria própria.

Figura A.23: Boxplots dos experimentos com a função 23.



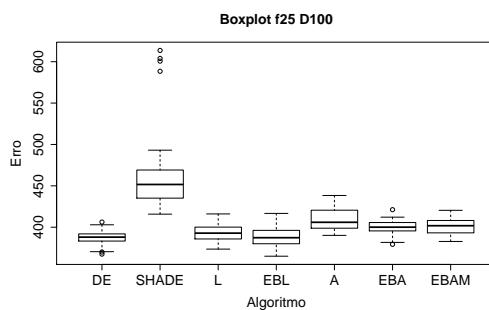
Fonte: autoria própria.

Figura A.24: Boxplots dos experimentos com a função 24.



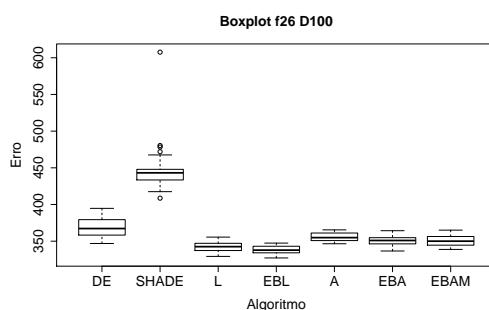
Fonte: autoria própria.

Figura A.25: Boxplots dos experimentos com a função 25.



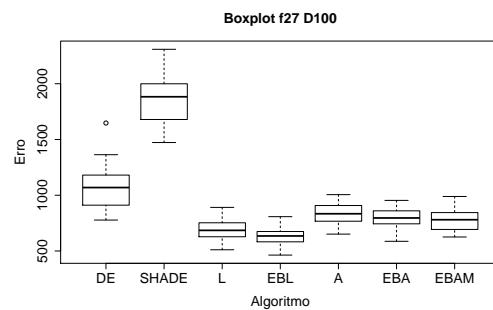
Fonte: autoria própria.

Figura A.26: Boxplots dos experimentos com a função 26.



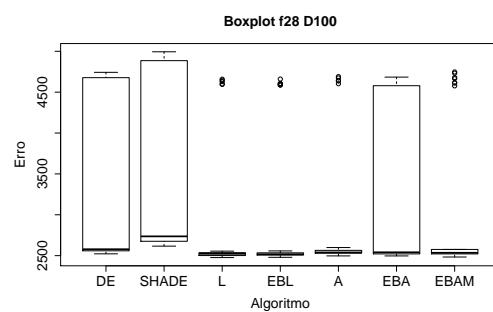
Fonte: autoria própria.

Figura A.27: Boxplots dos experimentos com a função 27.



Fonte: autoria própria.

Figura A.28: Boxplots dos experimentos com a função 28.



Fonte: autoria própria.