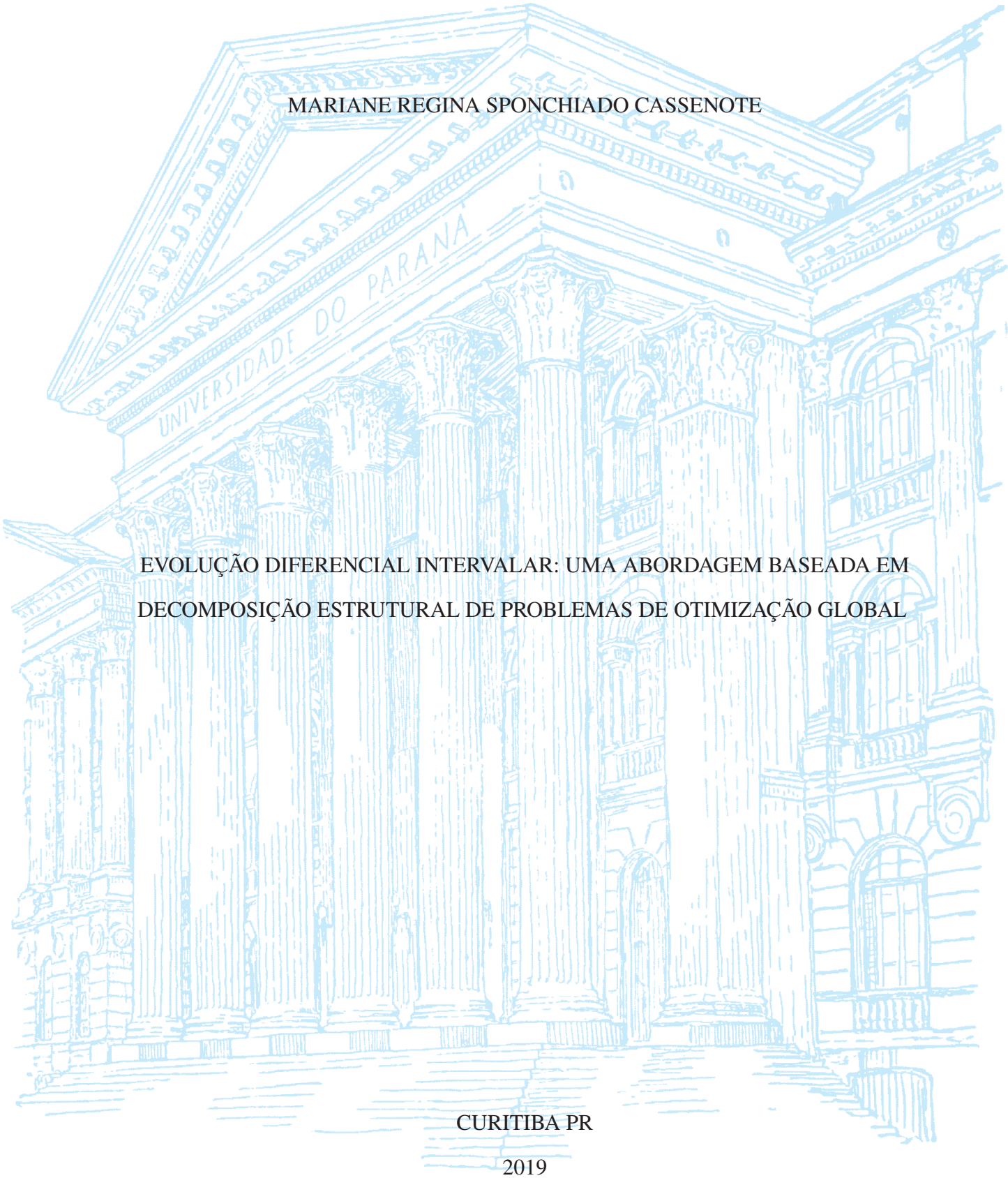
UNIVERSIDADE FEDERAL DO PARANÁ

MARIANE REGINA SPONCHIADO CASSENOTE



EVOLUÇÃO DIFERENCIAL INTERVALAR: UMA ABORDAGEM BASEADA EM DECOMPOSIÇÃO ESTRUTURAL DE PROBLEMAS DE OTIMIZAÇÃO GLOBAL

CURITIBA PR

2019

MARIANE REGINA SPONCHIADO CASSENOTE

EVOLUÇÃO DIFERENCIAL INTERVALAR: UMA ABORDAGEM BASEADA EM DECOMPOSIÇÃO ESTRUTURAL DE PROBLEMAS DE OTIMIZAÇÃO GLOBAL

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Fabiano Silva.

Coorientador: Guilherme Alex Derenievicz.

CURITIBA PR

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

C344e Cassenote, Mariane Regina Sponchiado
Evolução diferencial intervalar: uma abordagem baseada em decomposição estrutural de problemas de otimização global [recurso eletrônico] / Mariane Regina Sponchiado Cassenote – Curitiba, 2019.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática.
Orientador: Fabiano Silva
Coorientador: Guilherme Alex Derenievicz

1. Algoritmos de Computador. 2. Evolução Diferencial (Algoritmos). 3. Otimização Intervalar. I. Universidade Federal do Paraná. II. Silva, Fabiano. III. Derenievicz, Guilherme Alex. IV. Título.

CDD: 001.5

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **MARIANE REGINA SPONCHIADO CASSENOTE** intitulada: **Evolução Diferencial Intervalar: uma abordagem baseada em decomposição estrutural de problemas de otimização global**, após terem inquirido a aluna e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 18 de Junho de 2019.

FABIANO SILVA
Presidente da Banca Examinadora (UFPR)

MARCELO RODRIGUES BESSA
Avaliador Externo (UFPR)

EDUARDO JAQUES SPINOSA
Avaliador Interno (UFPR)

DANIEL WEINGAERTNER
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

Aos meus pais e ao meu irmão, vocês são minha força e inspiração.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me concedido força e coragem para encarar esse desafio e permanecer firme diante das dificuldades.

Aos meus orientadores e amigos, Profs. Fabiano Silva e Guilherme Derenievicz, pela oportunidade, confiança e inesgotável dedicação com a qual conduziram a pesquisa. Obrigada por todo o conhecimento compartilhado nas longas conversas que tornaram este trabalho realidade. Aos Profs. Eduardo Spinosa, Marcelo Bessa, Daniel Weingaertner e Marcos Castilho, pelas excelentes contribuições ao trabalho e à minha formação.

Aos meus pais, Paulo e Ivana, e ao meu irmão, Bruno, pelo amor e apoio incondicionais. Obrigada por terem compreendido a minha ausência, por acreditarem que valia a pena investir na minha formação e por nunca duvidarem da minha capacidade. Cada ligação, cada palavra de incentivo e cada gesto de "força" quando eu deixava a rodoviária me deram coragem e determinação para seguir em frente. Eu nunca alçaria voo se não houvesse um lar amoroso para onde voltar. Essa conquista também é de vocês.

Aos demais familiares, que nunca deixaram de me apoiar e incentivar. Aos meus avós, Mary, Angelino, Glória e Leonel, pelo amor e carinho dedicados. Em especial ao vô Angelino, que deu apoio e suporte à minha formação desde os anos iniciais e infelizmente não pode ver esse sonho se concretizar. Sua memória me deu forças para prosseguir nos momentos de dificuldade. Aos meus tios e primos e aos meus padrinhos, Ieda e Edgar, que sempre estiveram comigo e não passaram um dia sequer sem perguntar sobre mim durante esse período.

Aos meus amigos e colegas, em especial ao Bruno Zanette, à Tamy Beppler, ao Leonam Oliveira, ao Luis Felipe de Lima, ao Marcelino Ulica e ao Jonivan Oliveira, pelos momentos de descontração e companheirismo diário.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“É saber se sentir infinito
Num universo tão vasto e bonito
É saber sonhar
E então fazer valer a pena cada verso
Daquele poema sobre acreditar”
(Ana Vilela – Trem bala)

RESUMO

Algoritmos de Evolução Diferencial (DE) têm se mostrado promissores para abordagem de problemas de otimização numérica global com restrições. Muitas variantes recentes de DE são aplicadas a otimização caixa-preta, em que a estrutura analítica da instância é desconhecida. Neste trabalho é apresentado um algoritmo InDE (do inglês *Interval Differential Evolution*), que explora as informações de uma decomposição estrutural da instância durante o processo de otimização. As restrições e a função objetivo da instância são decompostas em operadores e variáveis originais e auxiliares. Então, a estrutura da instância é codificada em um hipergrafo cujas arestas representam os operadores e vértices representam as variáveis. Uma decomposição Epífita é aplicada sobre o hipergrafo, permitindo que um subconjunto de variáveis críticas seja extraído. Os operadores evolutivos intervalares do InDE são aplicados somente sobre esse subconjunto de variáveis, atribuindo a cada variável um intervalo de valores reais. Os intervalos das demais variáveis da instância são dados por propagação através do hipergrafo de restrições. O otimizador implementado utiliza diversas abordagens adaptativas estado-da-arte em otimização baseada em meta-heurísticas. Os experimentos indicam que a utilização de informação estrutural melhora significativamente o desempenho do algoritmo DE.

Palavras-chave: Evolução Diferencial, Otimização Global, Decomposição Estrutural, Otimização Intervalar.

ABSTRACT

Differential Evolution (DE) algorithms have shown to be promising for tackling numerical constrained global optimization problems (NCOPs). Many recent variants of DE are applied to black-box optimization, in which the analytical structure of the instance is unknown. In this work it is presented an Interval Differential Evolution (InDE) algorithm that explores the information of a structural decomposition of the NCOP instance during the optimization process. The constraints and objective function are decomposed into operators, and original and auxiliary variables. Then, the structure of the instance is encoded in a hypergraph whose edges represent the operators and vertices represent the variables. An Epiphytic decomposition is applied to the hypergraph, allowing a subset of critical variables to be extracted. InDE's evolutionary interval operators are applied only on this subset of variables, assigning to each variable an interval of real values. The intervals of others variables of the instance are given by propagation through the constraint hypergraph. The implemented optmizer uses several state-of-the-art adaptive approaches of metaheuristic-based optimization. The experiments indicate that the use of structural information significantly improves the performance of the DE algorithm.

Keywords: Differential Evolution, Global Optimization, Structural Decomposition, Interval Optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AMPL	*A Mathematical Programming Language*
CAL-SHADE	*Constraint Handling with Success History Adaptive Differential Evolution*
CEC	*IEEE Congress on Evolutionary Computation*
CoDE	*Composite Differential Evolution*
DE	*Differential Evolution*
GAC	*Generalized Arc-Consistency*
IDE	*Individual-Dependent Mechanism*
InDE	*Interval Differential Evolution*
IUDE	*Improved Unified Differential Evolution*
MA-ES	*Matrix Adaptation Evolution Strategy*
NCOP	*numerical constrained global optimization problem*
OGRe	Otimização Global Relaxada
RAC	*Relational Arc-Consistency*
SHADE	*Success-History based Adaptive Differential Evolution*
SOF	*Superiority of Feasible Points*
TLE	Tempo Limite Excedido
UDE	*Unified Differential Evolution*

SUMÁRIO

1 INTRODUÇÃO

A otimização numérica é uma área de pesquisa fundamental para diversos campos da ciência, tais como matemática aplicada e engenharia. Nesses cenários existem inúmeros problemas que podem ser considerados como otimização global, cujo objetivo básico é encontrar a melhor atribuição de valores para as variáveis de uma instância.

Uma instância de um prolema de otimização numérica é dada por um modelo matemático que a represente. A modelagem da instância é crucial, pois se o modelo não expressar corretamente o problema real, a atribuição de valores obtida pelo método de otimização não será adequada. O modelo é geralmente expresso por meio de uma função matemática denominada função objetivo (também denotada no cenário de algoritmos evolutivos como função de custo ou função de *fitness*) associada a um conjunto de variáveis que define o espaço de busca e um conjunto de restrições sobre os valores destas variáveis.

Formalmente, um problema de otimização global com restrições (ou simplesmente NCOP, do inglês *numerical constrained global optimization problem*) consiste na busca por uma atribuição de valores para um conjunto de variáveis $\mathcal{V} = \{x_1, \ldots, x_D\}$ que minimizam uma função objetivo $f : \mathbb{R}^D \mapsto \mathbb{R}$ sujeita a um conjunto de m restrições da forma $g_i(x_{i_1}, \ldots, x_{i_k}) \leq 0$ denominada *rede de restrições*, sendo $\{x_{i_1}, \ldots, x_{i_k}\} \subseteq \mathcal{V}$ o *escopo* da i-ésima restrição, para $1 \leq i \leq m$. D é o número de *dimensões* da instância.

Uma atribuição de valores é dita *factível* quando a valoração das variáveis respeita todas as restrições da instância. Uma atribuição de valores factível é uma *solução*. Um ponto mínimo do espaço de busca de uma instância de minimização pode ser classificado como global ou local. *Ótimo global* é a atribuição de valores que resulta no menor valor da função objetivo em todo o espaço de busca factível, enquanto que o *ótimo local* corresponde ao menor valor da função em um determinado subespaço do espaço de busca. Em instâncias com muitos pontos ótimos locais (instâncias multimodais) o processo de busca, em geral, torna-se mais complexo (Das et al., 2011).

Evolução Diferencial (ou DE, do inglês *Differential Evolution*) (Storn e Price, 1995) é um dos algoritmos evolutivos mais utilizados no contexto de NCOPs devido ao seu desempenho nas edições recentes da *Special Session & Competitions on Real-Parameter Single Objective Optimization*, que ocorre desde 2005 (Suganthan, 2018) e é organizada durante o *IEEE Congress on Evolutionary Computation* (CEC). Nas competições do CEC, algoritmos estado-da-arte são avaliados no contexto de otimização caixa-preta, em que a estrutura analítica na instância NCOP é desconhecida.

Por outro lado, por meio da análise intervalar (Moore, 1966), algoritmos de ramificação e poda têm sido largamente utilizados nas últimas décadas para resolver NCOPs de forma exata (Araya e Reyes, 2016) (Hansen e Walster, 2004) (Kearfott, 1992). Esses algoritmos percorrem o espaço de busca da instância a fim de garantir que a atribuição de valores encontrada representa o ponto ótimo global. Em muitas abordagens, são utilizadas técnicas de consistência local intervalar, como Consistência de Arco Generalizada (Mackworth, 1977) e Consistência de Envoltória (Benhamou e Older, 1992, 1997) que, de forma geral, necessitam de informações estruturais da instância para serem aplicadas. Essas consistências removem valores infactíveis dos intervalos atribuídos às variáveis.

Nesse contexto, Derenievicz (2018b) propôs um método de decomposição estrutural chamado *decomposição Epífita*. Essa decomposição utiliza uma codificação em restrições ternárias da instância NCOP com a função objetivo representada por uma variável adicional a ser

minimizada. A partir disso, o autor caracterizou uma importante classe de problemas em que uma combinação entre Consistência de Arco Generalizada e Consistência de Arco Relacional (Dechter e van Beek, 1997) garante uma atribuição ótima de valores em tempo polinomial.

O otimizador OGRe (Otimização Global Relaxada), proposto em Derenievicz (2018b), é um algoritmo de *Branch and Bound* intervalar que utiliza uma forma relaxada de Consistência de Arco Relacional a fim de encontrar uma aproximação ao mínimo global de instâncias NCOP. Esse algoritmo explora completa e sistematicamente o espaço de busca, pois a cada etapa seleciona uma variável e bissecta seu domínio a fim de executar a busca de forma recursiva. A principal desvantagem do OGRe é que a busca considera uma tolerância $\varepsilon > 0$ para satisfação das restrições da instância. O valor de ε é definido empiricamente, o que pode ser uma tarefa dispendiosa. Outro ponto negativo é o alto custo computacional da abordagem recursiva de *Branch and Bound*.

Neste trabalho é proposto um algoritmo de Evolução Diferencial Intervalar (ou InDE, do inglês *Interval Differential Evolution*) que utiliza a decomposição Epífita como fonte de informação estrutural da instância NCOP. Tal decomposição permite que o InDE aplique seus operadores somente em um subconjunto de componentes da estrutura da instância, enquanto os demais são atribuídos por propagação através do hipergrafo de restrições.

No InDE, as variáveis são representadas por intervalos. Sua população é composta por um conjunto de indivíduos em que cada um deles consiste em uma atribuição de intervalos para as variáveis da instância. Assim, cada indivíduo representa uma região do espaço de busca, em vez de somente um ponto específico, como no DE clássico. A cada avaliação de *fitness*, um procedimento de consistência local permite a poda de valores infactíveis desses intervalos, reduzindo a região do espaço de busca associada ao indivíduo.

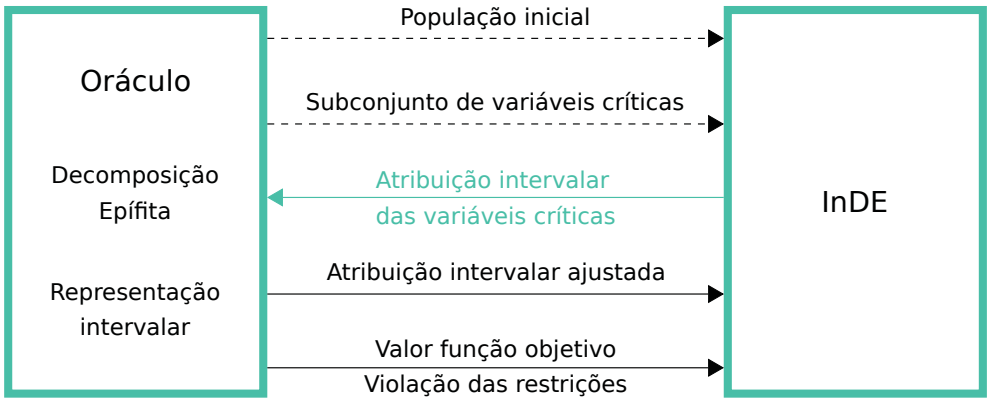


Figura 1.1: Comunicação entre o InDE e o oráculo baseado em decomposição Epífita.

Os recursos relacionados a decomposição Epífita, propagação de valores pelo hipergrafo de restrições e *Branch and Bound* intervalar do OGRe são utilizados pelo InDE como um oráculo. A Figura 1.1 ilustra a comunicação entre o oráculo e o InDE. Inicialmente, o oráculo extrai um subconjunto de variáveis críticas para o processo de busca e gera a população de indivíduos iniciais. A partir disso, o InDE aplica seus operadores somente sobre esse subconjunto de variáveis a fim de gerar um novo indivíduo candidato. O oráculo ajusta os intervalos das variáveis desse indivíduo a fim de evitar atribuições de valores infactíveis, além de computar seu valor da função objetivo e de violação de restrições.

Diferentemente do OGRe, o InDE realiza o processo de busca definindo dinamicamente o valor de tolerância das restrições. Isso significa que, no início da execução do algoritmo, ε é definido como um valor alto com o intuito de explorar amplamente o espaço e busca. Conforme o processo evolutivo avança, o valor ε é decrementado até zero, a fim de priorizar a exploração

de atribuições de valores factíveis. O tempo de processamento é limitado pelo número máximo de avaliações de *fitness*, definido por parâmetro.

O InDE proposto é baseado em diversas estratégias utilizadas em algoritmos estado-da-arte das competições do CEC, especialmente IUDE (Trivedi et al., 2018) e LSHADE44 (Poláková, 2017). A abordagem proposta foi experimentalmente avaliada sobre um conjunto de 80 instâncias selecionadas do *bechmark* COCONUT (Shcherbina et al., 2003). Os resultados apontam que a utilização de informação estrutural associada a técnicas de consistência local melhora significativamente o desempenho do algoritmo DE. Adicionalmente, foram encontradas mais soluções factíveis e com melhor aproximação que aquelas encontradas pelo otimizador OGRe.

1.1 OBJETIVOS

O principal objetivo deste trabalho é avaliar o impacto da utilização de decomposição estrutural associada a uma meta-heurística para otimização global com restrições. Os objetivos específicos do trabalho são:

- Realizar uma revisão de literatura para identificar as estratégias estado-da-arte da área de otimização restrita por meio de meta-heurísticas;
- Estudar os principais conceitos relacionados a otimização intervalar, técnicas de consistência local e decomposição Epífita;
- Implementar uma variante intervalar de DE (InDE) que utiliza estratégias de otimização meta-heurística e explora informações da decomposição estrutural da instância;
- Implementar uma variante de DE que utiliza as mesmas estratégias do InDE no contexto de otimização caixa-preta de parâmetros reais;
- Avaliar o desempenho do InDE sobre um conjunto de instâncias do *benchmark* COCONUT por meio de uma análise comparativa com o OGRe e o DE para otimização caixa-preta de parâmetros reais.

1.2 CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

- Um algoritmo de Evolução Diferencial Intervalar, denominado InDE, que utiliza decomposição estrutural e realiza o processo de otimização a partir de um subconjunto de componentes da estrutura das instâncias NCOP, além de empregar uma técnica de consistência local que ajusta as atribuições dos indivíduos a fim de evitar valores infactíveis;
- Adaptação de estratégias de DE tradicionalmente utilizadas em otimização de parâmetros reais para o contexto de representação intervalar;
- Implementação de uma variante de DE para otimização caixa-preta de parâmetros reais que utiliza uma combinação de estratégias estado-da-arte na área de otimização por meta-heurísticas.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte maneira:

- No Capítulo 2 são apresentadas as definições e conceitos básicos relacionados à otimização intervalar, decomposição Epífita e técnicas de consistência local, especificamente Consistência de Arco Generalizada e Consistência de Arco Relacional. Também são definidas as principais características de meta-heurísticas para otimização global, especificamente DE, e as técnicas de tratamento de restrições mais comuns na literatura recente;
- No Capítulo 3 é realizada uma revisão bibliográfica das principais abordagens adaptativas utilizadas no contexto de DE e dos algoritmos participantes da *Special Session & Competitions on Real-Parameter Single Objective Optimization*, organizada durante o *IEEE Congress on Evolutionary Computation*;
- No Capítulo 4 é descrito o algoritmo InDE proposto neste trabalho. São detalhados aspectos de manipulação da população, as adaptações realizadas nas operações clássicas de DE para o contexto intervalar e o procedimento de avaliação dos indivíduos da população;
- No Capítulo 5 é apresentada a avaliação experimental deste estudo, que inclui uma análise dos efeitos da decomposição estrutural sobre um conjunto de instâncias. Também é realizada uma análise comparativa entre o InDE, um DE para otimização caixa-preta de parâmetros reais e o OGRe, a fim de avaliar o impacto da decomposição Epífita sobre o desempenho do algoritmo e a viabilidade da utilização do InDE no contexto de otimização global com restrições;
- No Capítulo 6 são elencadas as conclusões do trabalho, assim como suas principais contribuições, limitações e alguns possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nas últimas décadas, diversas pesquisas têm sido realizadas no contexto de otimização global. Métodos que exploram características da função objetivo (por meio de vetores gradientes, por exemplo) foram desenvolvidos pela comunidade científica de matemática e pesquisa operacional. Alguns exemplos são os métodos Lagrangeano Aumentado (Hestenes, 1969) e Pontos Interiores (Potra e Wright, 1997), que partem de uma atribuição de valores inicial e percorrem o espaço de busca a fim de obter a atribuição ótima da instância. Ainda que essas abordagens sejam bastante utilizadas, elas não garantem a otimalidade global da atribuição encontrada.

Por outro lado, métodos exatos conseguem garantir a otimalidade da atribuição encontrada, pois percorrem todo o espaço de busca. Ramificação e poda, busca com retrocesso, métodos intervalares (Hansen e Walster, 2004) e busca exaustiva são alguns exemplos de abordagens dessa área. Apesar de garantirem a atribuição ótima, essas abordagens possuem um alto custo computacional, o que muitas vezes inviabiliza a sua utilização.

Outra comunidade que vem se destacando nos últimos anos é a de otimização por meta-heurísticas, que desenvolve métodos estocásticos que não garantem a obtenção da atribuição ótima de uma instância. No entanto, elas possibilitam a obtenção de atribuições que se aproximam do resultado ótimo em um tempo computacional viável, o que geralmente não pode ser alcançado com abordagens exatas que impõem um alto custo computacional (Gonçalves Júnior, 2016).

Neste estudo são reunidas técnicas da área de otimização exata a uma abordagem meta-heurística que tem se destacado no contexto de otimização global com restrições. Dessa forma, na Seção 2.1 são introduzidos conceitos relacionados à otimização global intervalar, consistência local e decomposição Epífita. Na Seção 2.2 são elencadas as principais características de meta-heurísticas, especificamente Evolução Diferencial (DE), e as técnicas de tratamento de restrições mais utilizadas na literatura recente.

2.1 OTIMIZAÇÃO GLOBAL INTERVALAR

Uma rede de restrições $\mathcal{R}$ (ou simplesmente uma rede) é uma tupla $(\mathcal{V}, \mathcal{D}, C)$, em que $\mathcal{V} = \{x_1, \ldots, x_D\}$ é um conjunto de variáveis com domínios $\mathcal{D} = \{X_1, \ldots, X_D\}$ e $C = \{R_{C_1}, \ldots, R_{C_m}\}$ é um conjunto de restrições tal que $C_i = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq \mathcal{V}$ é o *escopo* da relação $R_{C_i} \subseteq X_{i_1} \times \cdots \times X_{i_k}$, que define um conjunto de atribuições legais para as variáveis de C_i. A *aridade* de uma restrição é a cardinalidade de seu escopo. Uma rede de restrições $\mathcal{R}$ é dita *k-ária* se todas as suas restrições têm aridade k ou menor. Se os domínios das variáveis são conjuntos contínuos de números reais, $\mathcal{R}$ é dita uma *rede de restrições numérica*. Em tal rede, as restrições são usualmente da forma $R_{C_i} \equiv g_i(x_{i_1}, \ldots, x_{i_k}) \leq 0$. Nesse contexto, NCOP é definido pela função objetivo e a rede de restrições da instância.

A aritmética intervalar foi proposta nos anos 60 como uma abordagem formal para analisar erros de arredondamento em sistemas computacionais (Moore, 1966). Um *intervalo fechado* $X = [\underline{x}, \overline{x}]$ é o conjunto de números reais $X = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$, em que $\underline{x}, \overline{x} \in \mathbb{R} \cup \{-\infty, \infty\}$ são os *extremos* de X. Usaremos também a notação $\underline{X}$ para o extremo inferior $\underline{x}$ e $\overline{X}$ para o extremo superior $\overline{x}$ do intervalo X. O *comprimento* de um intervalo X é $\omega(X) = \overline{x} - \underline{x}$ e seu *ponto médio* é $\mu(X) = (\underline{x} + \overline{x})/2$. Um intervalo pode ser definido por seu comprimento e ponto médio como: $X = [\mu(X) - \omega(X)/2,\ \mu(X) + \omega(X)/2]$. Uma *caixa* é

uma tupla de intervalos $(X_1, X_2, \ldots, X_n)$. Dados dois intervalos X e Y a *extensão intervalar* de qualquer operador binário $\circ$ bem definido em $\mathbb{R}$ é definido pela Equação 2.1:

$$X \circ Y = \{x \circ y \mid x \in X,\ y \in Y \text{ e } x \circ y \text{ é definido em } \mathbb{R}\}. \tag{2.1}$$

O conjunto $X \circ Y$ pode ser um intervalo, um conjunto vazio ou um conjunto não contínuo de números reais (um *multi-intervalo*). É possível computar $X \circ Y$ para todas as funções algébricas e as principais transcendentais somente pela análise dos extremos de X e Y (Moore, 1966; Hansen e Walster, 2004). Por exemplo, as operação de adição, subtração, multiplicação e divisão intervalares são dadas, respectivamente, por:

$$\begin{aligned}
[\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] &= [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}] \\
[\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] &= [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}] \\
[\underline{X}, \overline{X}] \cdot [\underline{Y}, \overline{Y}] &= [\min\{\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY}\}, \max\{\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY}\}] \\
[\underline{X}, \overline{X}] \,/\, [\underline{Y}, \overline{Y}] &= [\underline{X}, \overline{X}] \cdot [1/\overline{Y}, 1/\underline{Y}], \text{ se } 0 \notin [\underline{Y}, \overline{Y}].
\end{aligned}$$

As operações de potenciação, radiciação e outras funções algébricas e trigonométricas podem ser definidas de maneira análoga, analisando os extremos dos intervalos. No entanto, a divisão por um intervalo que contém zero pode não resultar em um intervalo, pois $X/Y = \{x/y \mid x \in X,\ y \in Y \text{ e } y \neq 0\}$ não é necessariamente um conjunto contínuo de valores reais, ou seja, a operação resulta em um multi-intervalo.

Na próxima seção serão apresentados os princípios básicos de otimização por métodos exatos e técnicas de consistência local, que são utilizadas na proposta deste trabalho.

2.1.1 Métodos Exatos e Consistência Local

Desde o final dos anos 80, a aritmética intervalar tem sido utilizada no contexto de otimização exata de instâncias numéricas restritas pela combinação de técnicas de consistência da comunidade de programação por restrições com algoritmos de *Branch and Bound* (Araya e Reyes, 2016; Hansen e Walster, 2004; Kearfott, 1992). Um trabalho central nesse contexto é a linguagem Numérica (Van Hentenryck, 1997), cujo algoritmo principal forma a base dos otimizadores intervalares mais recentes. Métodos intervalares computam um conjunto de caixas de comprimento mínimo que contêm uma ou mais atribuições ótimas para a instância NCOP. Geralmente, esses métodos são compostos por três etapas principais que são repetidamente aplicadas até que uma atribuição seja encontrada:

1. Etapa de poda: contratores que garantem algum nível de consistência local são aplicados, removendo valores do domínio das variáveis que certamente não constituem uma atribuição factível da instância. No contexto numérico, esses contratores são geralmente implementados utilizando aritmética intervalar;

2. Busca local: métodos de busca computacionalmente baratos são utilizados para detectar se a caixa atual contém uma atribuição factível da instância que possa ser utilizada como um limite superior para a atribuição ótima. O método de Newton é um exemplo de busca local que pode ser aplicado nesta etapa;

3. Etapa de ramificação: uma variável é escolhida e seu domínio é bissectado para continuar a busca de forma recursiva. Boas heurísticas para escolha da variável e do lado de seu domínio que será processado primeiro são essenciais para um bom desempenho do método.

O método de ramificação e poda é, em essência, um método de retrocesso no qual cada ramificação constitui um nó na árvore de busca. Primeiramente, uma das partições do domínio é processada, para então processar-se a outra, sendo que atribuições parcialmente inconsistentes são descartadas do espaço de busca (Derenievicz, 2018b). Na etapa de poda, contratores de Consistência de Arco Generalizada (Mackworth, 1977) (Definição 2.1) e suas formas relaxadas, Consistência de Envoltória[1] (Benhamou e Older, 1992, 1997) e Consistência de Caixa (Benhamou et al., 1994), são geralmente utilizadas.

Definição 2.1 (Consistência de Arco Generalizada (GAC))

- *Uma restrição R_C é arco-consistente generalizada (ou simplesmente GAC[2]) em relação à variável $x_j \in C$ se, e somente se, para todo valor a_j no domínio de x_j existe uma atribuição de valores I das variáveis em $C \setminus \{x_j\}$ tal que $I \cup \{\langle x_j = a_j \rangle\}$ satisfaz R_C;*
- *Uma restrição R_C é GAC se, e somente se, R_C é GAC em relação a toda variável $x_j \in C$;*
- *Uma rede $\mathcal{R} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ é GAC se, e somente se, toda restrição $R_C \in \mathcal{C}$ é GAC.*

GAC garante que qualquer atribuição de uma única variável pode ser estendida às outras variáveis satisfazendo uma restrição. Em restrições numéricas ternárias da forma $x_1 = x_2 \circ_1 x_3$, sendo $\circ_1$ um operador binário qualquer, essa propriedade é assegurada pela Equação 2.2:

$$(X_1 \subseteq X_2 \circ_1 X_3) \wedge (X_2 \subseteq X_1 \circ_2 X_3) \wedge (X_3 \subseteq X_1 \circ_3 X_2), \tag{2.2}$$

em que X_i é o domínio de x_i, e $\circ_2$ e $\circ_3$ são as operações inversas de $\circ_1$ que mantêm a condição $(x_1 = x_2 \circ_1 x_3) \Leftrightarrow (x_2 = x_1 \circ_2 x_3) \Leftrightarrow (x_3 = x_1 \circ_3 x_2)$. Por exemplo: $(x = y + z) \Leftrightarrow (y = x - z) \Leftrightarrow (z = x - y)$, como $\circ_1 \equiv +$, então $\circ_2 \equiv -$ e $\circ_3 \equiv -$.

GAC é facilmente alcançada em restrições ternárias calculando a intersecção de cada domínio relevante com a respectiva *função de projeção*:

$$\texttt{contrator\_GAC}(x_1 = x_2 \circ_1 x_3) := \begin{cases} X_1 \leftarrow X_1 \cap (X_2 \circ_1 X_3) \\ X_2 \leftarrow X_2 \cap (X_1 \circ_2 X_3) \\ X_3 \leftarrow X_3 \cap (X_1 \circ_3 X_2) \end{cases} \tag{2.3}$$

Estritamente, a caixa (X_1, X_2, X_3) gerada pela Equação 2.3 não é completa devido à precisão finita de números em sistemas computacionais. Além disso, os domínios podem ser conjuntos desconectados de números reais (multi-intervalos).

Geralmente, a fim de aplicar o contrator GAC em uma restrição k-ária um procedimento de decomposição que transforma uma restrição em um conjunto equivalente de restrições ternárias é utilizado (Derenievicz, 2018b). Por exemplo, com uma variável auxiliar z_1 a restrição $x_1 \cdot (x_2 - x_1) = 0$ é decomposta em uma rede de duas novas restrições: $x_2 - x_1 = z_1$ e $x_1 \cdot z_1 = 0$. O contrator da Equação 2.3 é então repetidamente aplicado em ambas as restrições até que uma caixa que torne ambas as restrições GAC seja obtida ou um número máximo de passos seja atingido. Caso um conjunto de restrições não seja GAC, o processo resulta em uma contração aproximada que ainda assim elimina atribuições de valores infactíveis dos intervalos.

[1] Do inglês *Hull Consistency*.

[2] Do inglês *Generalized Arc Consistency*.

2.1.2 Decomposição Epífita

Um hipergrafo é um par $\mathcal{H} = (\mathcal{V}, E)$, em que $\mathcal{V}$ é um conjunto finito de vértices e $E \subseteq \{S \subseteq V \mid S \neq \emptyset\}$ é um conjunto finito de arestas. Se toda aresta $A \in E$ tem cardinalidade k, então $\mathcal{H}$ é dito um hipergrafo k-uniforme. Dado um hipergrafo $\mathcal{H} = (V, E)$, um Berge-caminho entre as arestas A_1 e A_n em $\mathcal{H}$ é uma sequência alternada $(A_1, v_1, A_2, v_2, \ldots, A_{n-1}, v_{n-1}, A_n)$ de arestas e vértices distintos (exceto possivelmente por A_1 e A_n) tal que $n \geq 3$ e $v_i \in (A_i \cap A_{i+1})$, para todo $1 \leq i < n$. Um Berge-ciclo é um Berge-caminho $(A_1, v_1, A_2, v_2, \ldots, A_{n-1}, v_{n-1}, A_n)$ tal que $A_1 = A_n$. Um hipergrafo $\mathcal{H}$ é dito Berge-cíclico se possui pelo menos um Berge-ciclo; do contrário, $\mathcal{H}$ é dito Berge-acíclico (Berge, 1985). Um hipergrafo $\mathcal{H}$ é conexo se existe pelo menos um Berge-caminho entre qualquer par de vértices em $\mathcal{H}$. Uma árvore é um hipergrafo conexo e Berge-acíclico.

No contexto de programação por restrições, um *hipergrafo de restrições* é uma representação estrutural da rede $\mathcal{R} = (\mathcal{V}, \mathcal{D}, C)$, em que $\mathcal{V}$ é um conjunto de vértices que representam as variáveis e $E = \{C_i \mid R_{C_i} \in C\}$ é um conjunto de arestas que representam o escopo das restrições.

Em Derenievicz e Silva (2018) essa representação estrutural é estendida a instâncias NCOP pela codificação da função objetivo $f(x_1, \ldots, x_D)$ como uma nova restrição $z = f(x_1, \ldots, x_D)$ na rede, em que z é chamada variável *raiz* da rede e seu domínio Z é igual a imagem da extensão intervalar de f sobre a caixa $(X_1, \ldots, X_D)$. Os autores encontraram uma relação entre o hipergrafo de restrições ternárias e o nível de consistência que garante uma atribuição livre de retrocesso, ou seja, uma busca que resolve uma instância NCOP sem encontrar nenhum conflito. Essa relação é garantida pela existência de uma decomposição particular do hipergrafo de restrições ternárias denominada decomposição Epífita.

Uma *decomposição Epífita* (Derenievicz e Silva, 2018; Derenievicz, 2018b) de um hipergrafo de restrições $\mathcal{H} = (\mathcal{V}, E)$ de acordo com $z \in \mathcal{V}$ é uma tripla $(\mathcal{A}, \Omega, t)$, em que $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ é um conjunto ordenado de hipergrafos disjuntos $\mathcal{A}_i = (V_i, E_i)$ obtidos pela remoção de um conjunto de arestas Ω de $\mathcal{H}$, e $t : \Omega \mapsto V_\Omega$ é uma função que associa cada aresta $C_i \in \Omega$ a um vértice $t(C_i) \in C_i$ de tal modo que:

1. $\mathcal{A}_1$ é uma árvore enraizada em z (ou seja, conectada e Berge-acíclica);
2. $\forall \mathcal{A}_{i>1} \in \mathcal{A}$ existe no máximo um $C_i \in \Omega$ tal que $t(C_i) \in V_i$ e $\mathcal{A}_i$ é uma árvore enraizada em $t(C_i)$ (ou qualquer outro vértice, se tal aresta C_i não existir);
3. se $t(C_i) \in V_i$, então $C_i \setminus \{t(C_i)\} \subseteq \bigcup_{j=1}^{i-1} V_j$.

O conjunto $V_\Omega = \bigcup\{x_j, x_k\}$, tal que $\left(x_i = x_j \circ x_k\right) \in \Omega$, é dado pelas variáveis que aparecem do lado direito das restrições do conjunto Ω.

Derenievicz e Silva (2018) demonstraram que uma instância NCOP representada pela decomposição Epífita do hipergrafo da sua rede de restrições pode ser resolvida de maneira livre de retrocesso se a rede satisfaz uma combinação entre GAC e Consistência de Arco Relacional (Definição 2.2).

Definição 2.2 (Consistência de Arco Relacional (RAC))

- *Dada uma rede $\mathcal{R} = (\mathcal{V}, \mathcal{D}, C)$, uma restrição $R_C \in C$ é arco-consistente relacional (ou simplesmente RAC[3]) em relação à variável $x_j \in C$ se, e somente se, para toda atribuição I das variáveis em $C \setminus \{x_j\}$, consistente com $\mathcal{R}$, existe um valor a_j no domínio de x_j tal que $I \cup \{\langle x_j = a_j \rangle\}$ satisfaz R_C;*

[3]Do inglês *Relational Arc-Consistent*.

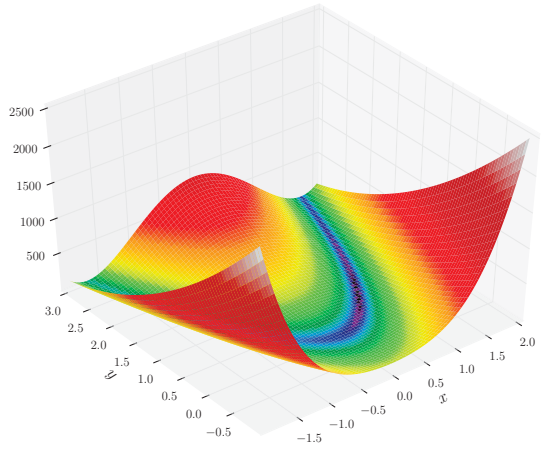


Figura 2.1: Função de Rosenbrock.

- *Uma restrição R_C é RAC se, e somente se, R_C é RAC em relação a toda variável $x_j \in C$;*
- *Uma rede $\mathcal{R} = (\mathcal{V}, \mathcal{D}, C)$ é RAC se, e somente se, toda restrição $R_C \in C$ é RAC.*

RAC é uma consistência local baseada em restrições proposta por Dechter e van Beek (1997) que garante que qualquer atribuição consistente a todas as variáveis, com exceção de uma x_j, pode ser estendida para x_j satisfazendo a restrição. Uma restrição da forma $x_1 = x_2 \circ_1 x_3$ é RAC com relação a x_1 se $X_1 \supseteq X_2 \circ_1 X_3$. Tal consistência permite a obtenção de uma atribuição factível da rede a partir da atribuição inicial $\langle z = \underline{Z} \rangle$ que minimiza a função objetivo, sendo Z o domínio da variável raiz z.

Mais especificamente, Derenievicz e Silva (2018) mostraram que é suficiente que somente as restrições representadas por arestas $C_i \in \Omega$ de uma decomposição Epífita sejam RAC com relação à variável representada por $t(C_i)$, enquanto as outras restrições sejam GAC. Uma decomposição Epífita pode ser encontrada em tempo linear. No entanto, nem todas as instâncias possuem tal decomposição. Os autores introduziram o conceito de decomposição k-Epífita como uma extensão que cobre todas as instâncias NCOP. Além disso, foi demonstrado que a maior parte das instâncias do *benchmark* COCONUT (Shcherbina et al., 2003) possuem decomposições Epífitas.

Um exemplo de decomposição Epífita dado em Derenievicz (2018b) é a função de Rosenbrock, comumente utilizada como teste para algoritmos de otimização. Essa função define graficamente um vale (Figura 2.1, extraída de Ortiz (2012)) no qual está contido o mínimo global da função, com valor $f(1,1) = 0$. Em geral, encontrar o vale é trivial, mas convergir para o ponto mínimo global é difícil.

A Figura 2.2 apresenta uma decomposição Epífita $(\mathcal{A}, \Omega, t)$ segundo v_1 da função Rosenbrock, definida por $f(x,y) = 100(x - y^2)^2 + (1-y)^2$, representada como uma rede ternária, em que $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$, $\mathcal{A}_1 = (\{v_1, \ldots v_{13}\}, \{C_1, \ldots, C_6\})$, $\mathcal{A}_2 = (\{v_{14}\}, \emptyset)$, $\Omega = \{C_7\}$ e $t(C_7) = v_{14}$. Como pode ser observado, o processo de codificação ternária adiciona novas variáveis à instância.

Enquanto o contrator GAC reduz o domínio das variáveis, um contrator RAC deve apertar a restrição em $C_i \setminus \{x_j\}$. Se tal restrição não existir, ela deve ser adicionada à rede, modificando a estrutura do hipergrafo e sua decomposição Epífita. Para evitar isso, Derenievicz e Silva (2018) propuseram um algoritmo de filtragem de domínio que alcança uma forma relaxada de RAC. O algoritmo proposto forma a base de um otimizador denominado OGRe (Otimização

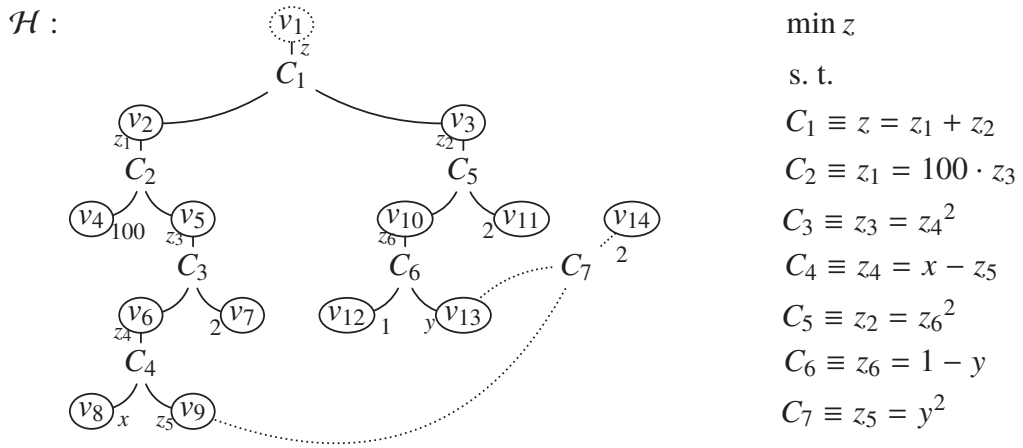


Figura 2.2: Uma decomposição Epífita segundo v_1 da representação ternária da função Rosenbrock. Os rótulos abaixo dos vértices indicam as variáveis que eles representam.

Global Relaxada) (Derenievicz, 2018a), que é uma variação do *Branch and Bound* intervalar tradicionalmente usado para otimização global.

O OGRe considera uma tolerância ε nas restrições representadas pelo conjunto Ω; portanto, é considerada uma versão relaxada da instância original. No exemplo da Figura 2.1, a restrição $C_7 \equiv z_5 = y^2$ passa a ser $C_7 \equiv z_5 = y^2 \pm \varepsilon$. No estágio de poda, o contrator GAC (Equação 2.3) é aplicado para reduzir o domínio das variáveis. Como uma estratégia de busca local, o OGRe tenta a instanciação total da rede de forma livre de retrocesso, começando pela atribuição inicial $\langle z = \underline{Z} \rangle$. Se alguma restrição do conjunto Ω não for satisfeita com a tolerância ε, os domínios de suas variáveis devem ser reduzidos para obter uma melhor aproximação de RAC. Então, na etapa de ramificação, uma variável da restrição não ε-factível do conjunto Ω é escolhida e seu domínio é bissectado para continuar a busca de forma recursiva.

O otimizador OGRe consiste em uma biblioteca multi-intervalar que inclui estruturas de dados e a implementação das operações básicas multi-intervalares $+$, $-$, $*$, $/$, $\sqrt{}$ e $\hat{}$. Em Derenievicz (2018b) o otimizador foi avaliado sobre um conjunto de instâncias propostas no *benchmark* COCONUT (Shcherbina et al., 2003), resolvendo cerca de 60% das instâncias com uma boa aproximação. O OGRe integra a avaliação experimental realizada no Capítulo 5, na qual é realizada uma análise comparativa a fim de avaliar a proposta deste estudo. Na próxima seção serão apresentados os principais conceitos da área de otimização por meta-heurísticas, em especial sobre Evolução Diferencial (DE).

2.2 OTIMIZAÇÃO POR META-HEURÍSTICAS

Na literatura da área de busca e otimização, um termo comumente encontrado é heurística. De acordo com Glover (1986), heurísticas são técnicas utilizadas para resolver dada instância de um problema em tempo computacional considerado aceitável, sem garantia de otimalidade do resultado. Assim, a aplicação de uma heurística para a resolução de uma instância tem por objetivo encontrar uma atribuição de valores que se aproxime do ótimo global.

A partir do desenvolvimento das abordagens heurísticas, surgiu o conceito de meta-heurística. De acordo com Reeves (1993), meta-heurísticas são métodos que promovem interações entre procedimentos de busca local e estratégias de alto nível a fim de criar métodos capazes de escapar de ótimos locais e realizar buscas mais robustas. Dessa forma, uma meta-heurística é um algoritmo projetado para resolver de forma aproximada uma ampla gama de instâncias sem que precise ser adaptado a cada uma delas.

A maior parte das meta-heurísticas compartilham as seguintes características: são inspiradas na natureza (baseadas em alguns princípios da física, biologia ou etologia); utilizam componentes estocásticos (envolvem variáveis aleatórias); e possuem diversos parâmetros que podem ser ajustados de acordo com a instância em questão (Boussaïd et al., 2013). Além disso, meta-heurísticas são comumente subdivididas entre abordagens de atribuição única de valores e abordagens baseadas em populações, em que as atribuições seguem um processo evolutivo inspirado no Darwinismo e competem entre si.

A fim de tratar instâncias com diferentes características, é importante que uma meta-heurística ofereça equilíbrio entre diversidade (exploração global) e intensificação (exploração local). O conceito de diversidade está relacionado à dispersão das atribuições de valores no espaço de busca e é necessário para identificar regiões factíveis. Já a exploração local intensifica a busca em torno de atribuições promissoras a fim de encontrar atribuições melhores na vizinhança. As principais diferenças entre as meta-heurísticas se referem à forma como cada uma delas busca o equilíbrio entre esses dois conceitos (Boussaïd et al., 2013).

Quando as atribuições de valores candidatas estão concentradas em torno de um subespaço específico do espaço de busca, diz-se que houve convergência. É desejável que as atribuições convirjam rapidamente para o ponto ótimo global do espaço de busca. No entanto, a aceleração do processo de convergência pode fazer com que as atribuições candidatas encontrem um ponto ótimo local e tenham dificuldades para se descolar para outras regiões. Com essa convergência prematura, o processo de busca se deteriora.

A seguir será brevemente apresentado o algoritmo de Evolução Diferencial, uma das meta-heurísticas que mais têm se destacado no cenário de otimização numérica com restrições.

2.2.1 Evolução Diferencial

Evolução Diferencial (DE) foi introduzida por Storn e Price (1995) por meio de uma meta-heurística evolutiva de codificação de ponto-flutuante para otimização em espaços de busca contínuos. O algoritmo DE clássico consiste em um laço de G gerações sobre a população $\mathtt{p}$ de indivíduos. Um indivíduo pode ser definido como uma atribuição de valores para todas as D variáveis da instância e é representado por um vetor $\mathtt{x} = (a_1, a_2, \ldots, a_D)$, em que $a_i \in X_i$ é um valor do domínio da variável x_i, $1 \leq i \leq D$. Uma população $\mathtt{p}$ é um conjunto de NP indivíduos e $\mathtt{p}_g$ denota a população na g-ésima geração, $0 \leq g \leq G$. A avaliação de *fitness* de um indivíduo é responsável por determinar sua qualidade ao longo do processo evolutivo.

A estrutura geral do DE é a mesma utilizada em algoritmos evolutivos tradicionais. Durante cada geração, os operadores de mutação, cruzamento e seleção são aplicados à população $\mathtt{p}_g$ até que um dado critério de parada seja satisfeito, como um determinado número de avaliações de *fitness* (*MaxFEs*). A população inicial $\mathtt{p}_0$ é gerada aleatoriamente de acordo com uma distribuição uniforme sobre $X_1 \times \cdots \times X_D$.

Muitas estratégias de mutação e/ou cruzamento têm sido propostas no contexto de DE (Neri e Tirronen, 2010). A fim de distinguir cada uma delas, a notação "DE / a / b / c" é utilizada, sendo que "a" especifica o indivíduo sobre o qual será aplicada a operação de mutação, "b" é o número de indivíduos diferença envolvidos na operação e "c" denota o esquema de cruzamento.

Na fase de mutação de cada geração, um operador é aplicado a fim de gerar um *indivíduo mutante* $\mathtt{v}_i$ para cada indivíduo $\mathtt{x}_i$ (chamado *indivíduo alvo*) da população. O operador de mutação mais conhecido é DE/rand/1, descrito a seguir:

$$\mathtt{v}_i = \mathtt{r}_1 + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{2.4}$$

sendo $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ três indivíduos mutuamente distintos aleatoriamente selecionados da população $\mathtt{p}_g$ e $(\mathtt{r}_2 - \mathtt{r}_3)$ o *vetor diferença* dessa operação, denotado por d. Os indivíduos r_1 e r_2 são chamados de *base* e *terminal*, respectivamente. O fator escalar $F \in [0,2]$ é um parâmetro de controle. Um valor pequeno de F promove a intensificação da busca em torno do indivíduo r_1, enquanto que um valor alto promove a exploração de outras regiões do espaço de busca.

Como será comentado nas próximas seções, ao longo dos últimos anos foram propostas diversas estratégias de mutação. Isso se deve ao fato de que não foi desenvolvida uma estratégia que se adapte bem a todas as particularidades que espaços de busca de diferentes instâncias podem apresentar. Isso recai na dificuldade de encontrar um meio de acelerar a convergência e evitar mínimos locais que não correspondam ao ponto ótimo do espaço de busca.

Após a etapa de mutação, a operação de cruzamento é aplicada a cada par de indivíduo alvo $\mathtt{x}_i$ e seu indivíduo mutante correspondente $\mathtt{v}_i$ a fim de gerar um *indivíduo experimental* $\mathtt{u}_i$. A taxa de cruzamento $CR \in [0,1)$ é um parâmetro de controle responsável por definir a fração do indivíduo mutante que será copiada para o indivíduo experimental. Ainda que existam dois tipos principais de operadores de cruzamento, binomial e exponencial, o cruzamento binomial é o mais utilizado na literatura e é descrito como a seguir:

$$\mathtt{u}_{i,j} = \begin{cases} \mathtt{v}_{i,j} & \text{se } rand_{i,j} \leq CR \text{ ou } j = j_{rand} \\ \mathtt{x}_{i,j} & \text{caso contrário,} \end{cases} \tag{2.5}$$

sendo $\mathtt{u}_{i,j}$, $\mathtt{v}_{i,j}$ e $\mathtt{x}_{i,j}$ os j-ésimos elementos dos indivíduos $\mathtt{u}_i$, $\mathtt{v}_i$ e $\mathtt{x}_i$, respectivamente. O valor $rand_{i,j}$ é um número uniformemente distribuído entre 0 e 1 e $j_{rand} \in \{1,\ldots,D\}$ é um índice aleatoriamente escolhido que garante que o indivíduo experimental terá ao menos um componente herdado do indivíduo mutante.

No operador de cruzamento exponencial, os componentes do indivíduo experimental $\mathtt{u}_i$ são herdados do indivíduo mutante correspondente $\mathtt{v}_i$ a partir de um índice aleatoriamente escolhido $rand_{i,j}[0,1] > CR$. O restante dos componentes do indivíduo experimental $\mathtt{u}_i$ são copiados do indivíduo alvo correspondente $\mathtt{x}_i$.

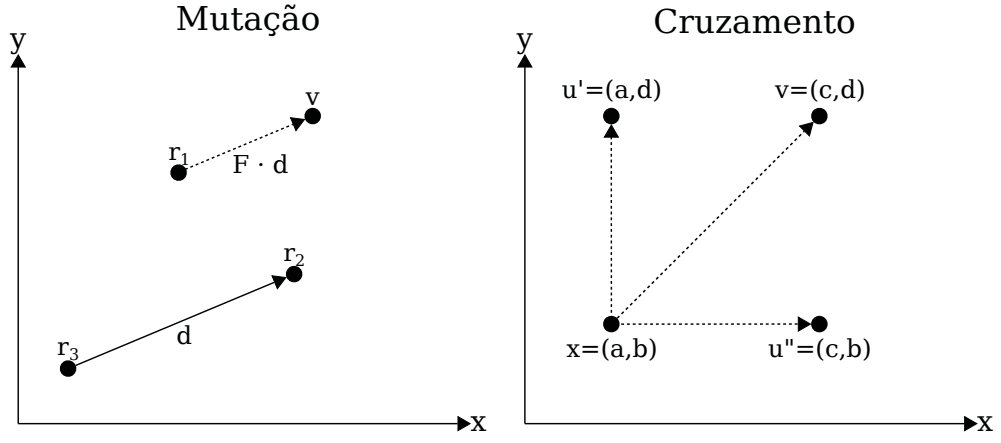


Figura 2.3: Exemplo de operações de mutação e cruzamento da estratégia DE/rand/1/bin sobre uma instância de duas dimensões.

A Figura 2.3 ilustra um exemplo das operações de mutação e cruzamento da estratégia DE/rand/1/bin em uma instância de duas dimensões. Na operação de mutação, $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ são indivíduos mutuamente distintos aleatoriamente selecionados na população. d é o vetor diferença entre os indivíduos $\mathtt{r}_2$ e $\mathtt{r}_3$. Para criação do indivíduo mutante $\mathtt{v}_i$, d é multiplicado pelo fator escalar F e adicionado a $\mathtt{r}_1$.

Na operação de cruzamento é criado o indivíduo experimental $\mathtt{u}_i$ por meio da escolha aleatória de variáveis correspondentes do indivíduo mutante $\mathtt{v}_i$ e do indivíduo alvo $\mathtt{x}_i$. A probabilidade de escolha de uma variável do indivíduo mutante é dada por *CR*. A fim de garantir que o indivíduo experimental não será igual ao indivíduo alvo, o algoritmo seleciona aleatoriamente uma variável para ser herdada do indivíduo mutante. No caso da Figura 2.3, após a operação de cruzamento, o indivíduo experimental pode ser $\mathtt{u}'_i$, $\mathtt{u}''_i$ ou $\mathtt{v}_i$.

O valor da função objetivo de cada indivíduo experimental $f(\mathtt{u}_i)$ é comparado com o valor da função objetivo de seu indivíduo alvo correspondente $f(\mathtt{x}_i)$ na população atual $\mathtt{p}_g$. Se o valor de $f(\mathtt{u}_i)$ for melhor ou igual a $f(\mathtt{x}_i)$, $\mathtt{u}_i$ substitui $\mathtt{x}_i$ na população da próxima geração $\mathtt{p}_{g+1}$. Caso contrário, $\mathtt{x}_i$ permanece na população da próxima geração. Uma operação em que $f(\mathtt{u}_i)$ é selecionado é dita uma atualização de sucesso. Na Seção 2.2.2 são citadas algumas técnicas utilizadas na literatura para a operação de seleção no cenário de otimização de instâncias com restrições.

2.2.2 Técnicas de Tratamento de Restrições

Sabe-se que meta-heurísticas são uma alternativa interessante para abordagem de instâncias que não podem ser eficientemente resolvidas por métodos exatos. No entanto, elas são tradicionalmente projetadas para instâncias de otimização sem restrições. Diversas técnicas de tratamento de restrições têm sido desenvolvidas e adaptadas para integrarem meta-heurísticas (Mezura-Montes e Coello, 2011).

Ainda que não haja um consenso na literatura quanto ao melhor tipo de abordagem, as técnicas existentes ajudam a guiar o processo de busca em direção à regiões factíveis do espaço de busca. Entende-se como região factível um subespaço do espaço de busca em que os indivíduos ali contidos não violam nenhuma restrição da instância em questão.

A seguir serão apresentados os principais tipos de técnicas de tratamento de restrições de acordo com Michalewicz (1995), Coello (2002), Zhang et al. (2004) e Mezura-Montes e Coello (2011).

Funções de Penalização

A aplicação de funções de penalização é uma das técnicas mais utilizadas para tratamento de restrições. Esse tipo de abordagem transforma uma instância de um problema de otimização restrita em irrestrita de maneira que, em instâncias de minimização, é adicionado um determinado valor à função objetivo com base na violação de restrições do indivíduo. Dessa forma, aumenta-se o valor de *fitness* dos indivíduos que violam alguma restrição, que serão desfavorecidos na etapa de seleção do processo evolutivo.

Em otimização clássica são consideradas funções de pontos exteriores e interiores, que remetem à factibilidade das regiões do espaço de busca. Em funções de pontos exteriores, inicia-se com uma atribuição de valores infactível e busca-se movê-la para uma região factível. No caso de funções de pontos interiores, inicia-se com uma atribuição factível e a penalização é definida de forma que seja pequena em pontos distantes dos limites das restrições e tenda ao infinito à medida em que se aproxima desses limites. Os métodos mais utilizados pela comunidade científica de meta-heurísticas evolutivas são baseados em penalizações por pontos exteriores, justamente por não exigirem uma atribuição inicial factível.

Embora sua implementação seja bastante simples, as funções de penalidade exigem um cuidadoso ajuste de seus fatores de penalização, que são altamente dependentes da instância, o que dificulta sua utilização. Se a penalização for muito grande e houver várias regiões factíveis disjuntas no espaço de busca, os indivíduos tendem a se mover para uma delas e dificilmente

chegarão a uma região factível diferente. Por outro lado, se a penalidade for muito pequena, muito tempo de busca é gasto na exploração de regiões infactíveis, pois a penalidade é pouco significativa em relação à função objetivo. A seguir serão apresentados alguns tipos de funções de penalização.

A penalidade de morte consiste na rejeição de indivíduos que violem alguma restrição. Para substituição da atribuição infactível, novos indivíduos são gerados até que uma atribuição factível seja encontrada. Essa abordagem pode resultar na estagnação do processo de busca em instâncias com regiões factíveis muito pequenas, além de descartar informações potencialmente úteis de indivíduos infactíveis.

Na categoria de penalidade estática encontram-se as funções de penalização que não consideram o número da geração atual e permanecem constantes durante o processo evolutivo. Existem diversas estratégias de utilização de penalidades estáticas, tais como a penalização por níveis de violação (Homaifar et al., 1994) e a penalização pela proporção de restrições violadas (Morales e Quezada, 1998). As principais desvantagens desse tipo de abordagem são a quantidade de parâmetros envolvidos e a dificuldade de ajustá-los, visto que são muito dependentes de cada instância (Michalewicz, 1995).

Penalização dinâmica é definida como qualquer função em que o número da geração atual é considerado no cálculo da penalidade aplicada a uma atribuição. Dessa forma, penalidades pequenas são utilizadas no início do processo de otimização a fim de encontrar uma região factível do espaço de busca. Penalidades maiores são empregadas na porção final da busca para preservar as atribuições factíveis encontradas e acelerar a convergência em direção ao ponto ótimo global. Em geral, os parâmetros de ajuste de penalidades dinâmicas são críticos e podem ser tão difíceis de generalizar para mais de uma instância quanto os parâmetros de uma função de penalidade estática.

Funções de penalidade que utilizam informações sobre o processo de busca são denominadas penalizações adaptativas. Na abordagem mais utilizada, o fator de penalização em uma determinada geração é reduzido se as melhores atribuições das últimas k gerações forem factíveis ou aumentado se as melhores atribuições forem infactíveis. Se houver atribuições factíveis e infactíveis dentre as melhores da população, a penalização não é alterada. Vale observar que esse tipo de estratégia ajusta o fator de penalização de forma mais automática que as versões anteriores, mas a definição do intervalo de k gerações pode ser crítica.

O principal desafio com relação às funções de penalização é que o ajuste adequado dos parâmetros geralmente não é previamente conhecido. Caso a penalização adotada seja muito baixa ou muito alta, o desempenho do processo de busca se deteriora. No entanto, pesquisadores da área argumentam que o ajuste de parâmetros das demais estratégias é menos complexo que o ajuste de funções de penalidade estáticas.

Ranking Estocástico

Essa abordagem foi proposta por Runarsson e Yao (2000) a fim de sanar algumas deficiências apresentadas pelas abordagens de funções de penalização. Para tanto, a técnica de *Ranking* Estocástico tenta equilibrar a influência da função objetivo e da função de penalização ao atribuir um valor de *fitness* a um indivíduo. Ainda que essa abordagem não utilize um fator de penalização, é necessário definir o parâmetro P_f, que determina o equilíbrio entre função objetivo e fator de penalização.

A população é ordenada por um processo semelhante ao algoritmo *bubble-sort* (que realiza ordenação baseada em comparação entre pares). De acordo com o valor de P_f, a comparação entre dois indivíduos adjacentes é dada a partir da função objetivo. O restante das comparações ocorrem por meio da soma das violações das restrições. Dessa forma, P_f introduz

o componente estocástico ao processo de *ranking*, visto que algumas atribuições infactíveis podem conseguir uma boa classificação. Apesar da técnica ser de fácil implementação, o ajuste do parâmetro P_f possui grande influência no desempenho desse método.

Estratégias de Reparação

As estratégias de reparação são comumente utilizadas em meta-heurísticas para abordar instâncias de otimização combinatória (por exemplo, o problema do caixeiro-viajante, o problema da mochila, etc.), pois nesses casos a reparação de uma atribuição infactível é mais simples. Por exemplo, quando uma restrição é violada no problema da mochila, basta retirar itens até que as restrições sejam novamente respeitadas.

Não há uma heurística padrão para o projeto de estratégias de reparação. Normalmente, é possível utilizar um algoritmo guloso (ou seja, um algoritmo que prossegue por uma série de alternativas tomando a melhor decisão possível em cada ponto da série), um algoritmo aleatório ou qualquer outra heurística que oriente o processo de reparação. Dessa forma, o sucesso dessa abordagem depende da heurística utilizada.

Quando uma atribuição infactível pode ser facilmente transformada em uma atribuição factível, as estratégias de reparação são boas alternativas. No entanto, isso nem sempre é possível e os operadores de reparo podem introduzir um certo viés na busca, deteriorando o processo evolutivo. Além disso, a estratégia de reparo deve ser projetada individualmente para cada instância abordada, o que pode não ser viável em alguns casos.

Separação entre Objetivos e Restrições

Diferentemente de funções de penalização, que combinam o valor da função objetivo e a violação de restrições de uma atribuição para definir o valor de *fitness*, essas abordagens consideram objetivos e restrições separadamente. Nessa categoria, a função objetivo e a violação de restrições são otimizadas segundo uma ordem lexicográfica em que a violação de restrições precede a função objetivo.

Superioridade de Soluções Factíveis (*Superiority of Feasible Points* - SOF) é um método proposto por Deb (2000) que utiliza um operador de seleção em que duas atribuições são comparadas entre si de acordo com os seguintes critérios:

1. Uma atribuição factível é sempre melhor que uma atribuição infactível;
2. Entre duas atribuições factíveis, a que tiver o melhor valor de função objetivo é escolhida;
3. Entre duas atribuições infactíveis, a que tiver menor violação de restrições é escolhida.

No método SOF nenhum fator de penalização é utilizado, visto que o processo de seleção somente realiza comparações entre pares de atribuições. Dessa forma, atribuições factíveis possuem valor de *fitness* igual ao seu valor de função objetivo e a violação de restrições indiretamente classifica as atribuições infactíveis. A maior desvantagem dessa abordagem é a perda de diversidade entre atribuições.

O método ε-*constrained* (Takahama e Sakai, 2010) segue o mesmo princípio de precedência de violação de restrições sobre o valor da função objetivo, sendo essa precedência ajustada pelo parâmetro ε. Sejam $f(\mathrm{x})$ e $\phi(\mathrm{x})$ os valores de função objetivo e violação de restrições do indivíduo x, respectivamente. Então, para qualquer ε que satisfaça $\varepsilon \geq 0$, as

comparações $<_\varepsilon$ e $\leq_\varepsilon$ entre dois indivíduos x_1 e x_2 são realizadas conforme as Definições 2.6 e 2.7:

$$(f(\mathrm{x}_1), \phi(\mathrm{x}_1)) <_\varepsilon (f(\mathrm{x}_2), \phi(\mathrm{x}_2)) \Leftrightarrow \begin{cases} f(\mathrm{x}_1) < f(\mathrm{x}_2) & \text{se } \phi(\mathrm{x}_1), \phi(\mathrm{x}_2) \leq \varepsilon \\ f(\mathrm{x}_1) < f(\mathrm{x}_2) & \text{se } \phi(\mathrm{x}_1) = \phi(\mathrm{x}_2) \\ \phi(\mathrm{x}_1) < \phi(\mathrm{x}_2) & \text{caso contrário,} \end{cases} \tag{2.6}$$

$$(f(\mathrm{x}_1), \phi(\mathrm{x}_1)) \leq_\varepsilon (f(\mathrm{x}_2), \phi(\mathrm{x}_2)) \Leftrightarrow \begin{cases} f(\mathrm{x}_1) \leq f(\mathrm{x}_2) & \text{se } \phi(\mathrm{x}_1), \phi(\mathrm{x}_2) \leq \varepsilon \\ f(\mathrm{x}_1) \leq f(\mathrm{x}_2) & \text{se } \phi(\mathrm{x}_1) = \phi(\mathrm{x}_2) \\ \phi(\mathrm{x}_1) \leq \phi(\mathrm{x}_2) & \text{caso contrário.} \end{cases} \tag{2.7}$$

Em casos em que $\varepsilon = \infty$, as comparações $<_\infty$ e $\leq_\infty$ pelo valor ε são equivalentes às comparações diretas entre valores da função objetivo. Além disso, em casos em que $\varepsilon = 0$, $<_0$ e $\leq_0$ são equivalentes à ordem lexicográfica utilizada no SOF em que a violação de restrições $\phi(\mathrm{x})$ precede o valor de função objetivo $f(\mathrm{x})$.

Uma instância de otimização resolvida pelo método *ε-constrained* substitui o método clássico de comparação por um processo baseado em ε. Então, uma instância P^ε é definida de forma que as restrições de P dadas por $\phi(\mathrm{x}) = 0$ são relaxadas e substituídas por $\phi(\mathrm{x}) \leq \varepsilon$, como definido a seguir:

$$(P^\varepsilon) \quad \text{minimize} \quad f(\mathrm{x}) \tag{2.8}$$

$$\text{sujeito a} \quad \phi(\mathrm{x}) \leq \varepsilon. \tag{2.9}$$

Normalmente ε não precisa ser controlado, pois muitas instâncias com restrições podem ser resolvidas através de uma ordem lexicográfica com $\varepsilon = 0$. No entanto, para instâncias com restrições de igualdade o ε deve ser devidamente controlado a fim de obter boas atribuições.

Em Takahama e Sakai (2010) foi definido um esquema em que ε é atualizado até o número de iterações atingir uma determinada geração de controle. Após o número de gerações g atingir a geração de controle T, o valor de ε é definido como 0 a fim de obter atribuições com a menor violação de restrições possível. O valor de ε é decrementado conforme o número de gerações aumenta, como descrito na Equação 2.10:

$$\varepsilon = \begin{cases} \varepsilon_0 \cdot (1 - \frac{g}{T})^{cp} & \text{se } 0 < g < T \\ 0 & \text{se } g \geq T, \end{cases} \tag{2.10}$$

em que ε_0 é a violação de restrições do melhor θ-ésimo indivíduo da população inicial, cp é um parâmetro que controla a velocidade de redução do relaxamento das restrições.

Ainda que não haja consenso na literatura quanto à técnica de tratamento de restrições mais robusta, a revisão bibliográfica do Capítulo 3 revelou que as abordagens mais recentes no contexto de otimização de instâncias com restrições por meta-heurísticas utilizam principalmente os métodos SOF e *ε-constrained*.

2.3 CONSIDERAÇÕES

Neste capítulo foram elencados alguns conceitos relacionados a duas comunidades de otimização: métodos exatos e meta-heurísticas. Abordaram-se as noções de otimização intervalar, na qual as variáveis são representadas por intervalos; técnicas de consistência, que são responsáveis por podar regiões infactíveis do espaço de busca; e decomposição Epífita, que é o método de decomposição estrutural utilizado pelo algoritmo InDE proposto neste estudo. Além

disso, foi detalhado o funcionamento do algoritmo de Evolução Diferencial (DE) e as técnicas de tratamento de restrições mais comuns na literatura atual.

No próximo capítulo será realizada uma revisão bibliográfica acerca das abordagens adaptativas mais recentes no contexto de DE, além de os algoritmos participantes das últimas edições da competição *Special Session & Competitions on Real-Parameter Single Objective Optimization* do *IEEE Congress on Evolutionary Computation (CEC)*.

3 TRABALHOS RELACIONADOS

A Evolução Diferencial (DE) é uma técnica de busca estocástica utilizada para abordar instâncias de problemas de otimização em espaços contínuos e tem sido amplamente empregada em diversos estudos e aplicações do mundo real (Zhang e Sanderson, 2009). No entanto, o sucesso do DE na resolução de uma instância depende da escolha adequada da estratégia de geração de indivíduos experimentais, da definição dos parâmetros F e CR e do tamanho da população dado por NP.

Diante das dificuldades enfrentadas ao ajustar manualmente esses parâmetros, diversas abordagens adaptativas foram recentemente propostas. Com isso, as configurações de parâmetros são definidas automaticamente, o que torna o algoritmo DE mais facilmente utilizável e robusto. A associação desses mecanismos com técnicas de tratamento de restrições têm permitido que variantes de DE obtenham excelente desempenho ao abordar instâncias de problemas de otimização numérica com restrições, como será mostrado no decorrer deste capítulo.

Na Seção 3.1 são comentadas algumas das abordagens adaptativas recentes que obtiveram maior destaque na literatura. Na Seção 3.2 são apresentadas as técnicas e estratégias utilizadas nos algoritmos participantes da competição *Special Session & Competitions on Real-Parameter Single Objective Optimization 2018*, organizada durante o *IEEE Congress on Evolutionary Computation (CEC)*. A partir da análise dessas abordagens, espera-se fornecer os subsídios necessários para o entendimento dos detalhes da abordagem proposta neste trabalho.

3.1 ABORDAGENS ADAPTATIVAS

Uma das maiores dificuldades em DE é encontrar uma configuração de parâmetros adequada para F e CR. Em geral, a definição de um valor alto para F pode fazer com que os indivíduos experimentais sejam melhor distribuídos no espaço de busca e se tenha uma boa diversidade populacional; enquanto que valores baixos restringem a busca em torno dos indivíduos atuais e pode acelerar a convergência para um ponto ótimo local. Da mesma forma, valores altos de CR podem tornar o indivíduo experimental muito diferente do indivíduo atual, o que aumenta a diversidade populacional, mas pode desacelerar a convergência (Wang et al., 2011).

Empregar um esquema de tentativa e erro em busca da estratégia e das configurações de parâmetros mais adequadas para cada instância é um processo exaustivo que demanda muitos recursos computacionais. Além disso, essa escolha pode variar de acordo com a região do espaço de busca que está sendo percorrida por cada indivíduo. Dessa forma, diversos esquemas adaptativos têm sido propostos com o intuito de selecionar automaticamente estratégias e configurações de parâmetros a partir da análise do processo evolutivo. Algumas abordagens adaptativas encontradas na literatura recente são brevemente comentadas a seguir.

JADE

O algoritmo JADE foi proposto por Zhang e Sanderson (2009) e introduz uma nova estratégia de mutação denominada DE/current-to-$p$best/1 com arquivo externo opcional. A Equação 3.1 ilustra a versão sem arquivo dessa estratégia:

$$\mathtt{v}_i = \mathtt{x}_i + F_i \cdot (\mathtt{r}_{p\text{best}} - \mathtt{x}_i) + F_i \cdot (\mathtt{r}_1 - \mathtt{r}_2), \tag{3.1}$$

sendo $\mathtt{r}_{p\text{best}}$ aleatoriamente escolhido entre os melhores $(100 \cdot p)\%$ indivíduos da população atual tal que $p \in (0, 1]$, F_i é o fator escalar associado ao indivíduo $\mathtt{x}_i$, e $\mathtt{r}_1$ e $\mathtt{r}_2$ são indivíduos mutuamente distintos e distintos de $\mathtt{r}_{p\text{best}}$ aleatoriamente selecionados da população.

O algoritmo JADE armazena em memória indivíduos recentemente explorados que não obtiveram sucesso na operação de seleção a fim de prover informações adicionais sobre direções promissoras do espaço de busca e aumentar a diversidade populacional. Sendo $\mathtt{a}_g$ e $\mathtt{p}_g$ o conjunto de indivíduos no arquivo externo na geração g e a população atual, respectivamente, DE/current-to-$p$best/1 com arquivo externo difere da Equação 3.1 apenas na seleção de $\mathtt{r}_2$, que é escolhido em $\mathtt{a}_g \cup \mathtt{p}_g$. Se o tamanho do arquivo excede um certo limite, por exemplo NP, indivíduos são aleatoriamente removidos.

Para a adaptação de parâmetros, a cada geração o fator de mutação F_i de cada indivíduo $\mathtt{x}_i$ é gerado por meio de uma distribuição de Cauchy com parâmetro de localização γ_F e fator escalar 0,1; ou seja, $F_i = \text{randc}_i(\gamma_F; 0, 1)$, e então truncado para 1 se $F_i \geq 1$ ou gerado novamente se $F_i \leq 0$. Denota-se por SF o conjunto de todos os fatores de mutação F_i que obtiveram sucesso na geração g. O parâmetro de localização γ_F da distribuição de Cauchy é inicializado como 0,5 e atualizado ao fim de cada geração como na Equação 3.2:

$$\gamma_F = (1 - c) \cdot \gamma_F + c \cdot \text{média}_L(SF), \tag{3.2}$$

sendo c uma constante positiva entre 0 e 1 e $\text{média}_L(\,)$ a média de Lehmer, definida na Equação 3.3:

$$\text{média}_L(SF) = \frac{\sum_{F \in SF} F^2}{\sum_{F \in SF} F}. \tag{3.3}$$

Similarmente, a cada geração a probabilidade de cruzamento CR_i de cada indivíduo $\mathtt{x}_i$ é gerada por uma distribuição normal com média γ_{CR} e desvio padrão 0,1; ou seja, $CR_i = \text{randn}_i(\gamma_{CR}; 0{,}1)$, e então truncada para $[0, 1]$. Denota-se por SCR o conjunto de todos os valores de CR_i que obtiveram sucesso na geração g. A média γ_{CR} é inicializada como 0,5 e então atualizada ao fim de cada geração conforme a Equação 3.4:

$$\gamma_{CR} = (1 - c) \cdot \gamma_{CR} + \text{média}_A(SCR), \tag{3.4}$$

sendo c uma constante positiva entre 0 e 1 e $\text{média}_A(\,)$ a média aritmética simples.

Dessa forma, o algoritmo JADE utiliza uma estratégia de adaptação dos parâmetros F e CR que faz com que configurações bem sucedidas sejam registradas e utilizadas nas gerações seguintes. No entanto, essa abordagem introduz dois novos parâmetros: o controle da taxa de adaptação c e p que determina o quanto a estratégia de mutação irá priorizar os melhores indivíduos da população. De acordo com Zhang e Sanderson (2009), o JADE geralmente apresenta bom desempenho com $1/c \in [5, 20]$ e $p \in [5\%, 20\%]$, ou seja, a vida útil dos valores γ_{CR} e γ_F varia entre 5 e 20 gerações e considera-se 5 a 20% dos melhores indivíduos na operação de mutação.

SaDE

Qin et al. (2009) propuseram uma variante de DE auto-adaptativa denominada SaDE que usa, de forma análoga ao JADE, as estratégias de mutação e as configurações de parâmetros gradualmente adaptadas de acordo com sua taxa de sucesso em gerações anteriores.

O algoritmo SaDE utiliza um conjunto de quatro estratégias para adaptação: DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin e DE/current-to-rand/1. Para cada indivíduo da população atual, uma estratégia de mutação é selecionada desse conjunto de acordo com sua

taxa de sucesso em um determinado número de gerações anteriores. A Equação 3.5 define a estratégia DE/current-to-rand/1, também utilizada na abordagem proposta neste trabalho:

$$\mathtt{v}_i = \mathtt{x}_i + s_i \cdot (\mathtt{r}_1 - \mathtt{x}_i) + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{3.5}$$

em que $\mathtt{x}_i$ é o indivíduo alvo da população atual, $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ são três indivíduos mutuamente distintos aleatoriamente selecionados e s_i é um número aleatório uniformemente distribuído entre 0 e 1.

A escolha da estratégia a ser aplicada a um indivíduo segue um conjunto de probabilidades que são ajustadas durante o processo de busca. Sejam c_k ($k = 1, 2, \ldots, K$) a probabilidade de aplicação da k-ésima estratégia a um indivíduo da população atual e K o número estratégias disponíveis. Inicialmente, as probabilidades de todas as estratégias são inicializadas como $1/K$, ou seja, todas as estratégias possuem a mesma probabilidade de serem selecionadas.

Na geração g, o número de indivíduos gerados pela k-ésima estratégia que obtiveram sucesso e o número de indivíduos que falharam na operação de seleção são armazenados em memórias por um número fixo de gerações, chamado de período de aprendizagem L. Após as primeiras L gerações, a probabilidade de escolha de cada estratégia é atualizada a cada geração com base nessas memórias de sucesso e fracasso, ou seja, a estratégia escolhida é a que obteve maior taxa de sucesso nas últimas L gerações. Quando as memórias ultrapassam o tamanho máximo, os registros mais antigos são removidos para que os registros da geração atual possam ser armazenados.

Além do mecanismo adaptativo de escolha da estratégia da operação de mutação, o SaDE utiliza um esquema de adaptação dos parâmetros de controle F_i e CR_i para cada indivíduo $\mathtt{x}_i$. Nesse caso, o parâmetro F_i é aproximado por uma distribuição normal com valor médio 0,5 e desvio padrão 0,3, denotada por $F_i = \text{randn}_i(0,5; 0,3)$. A amostragem aleatória de um valor de F_i para cada indivíduo $\mathtt{x}_i$ da população permite a busca local em torno dos indivíduos atuais com valores de F_i pequenos e a descoberta de novas regiões do espaço de busca com valores de F_i maiores.

A probabilidade de cruzamento CR_i segue uma distribuição normal com valor médio γ_{CR} e desvio padrão std, denotada por $CR_i = \text{randn}_i(\gamma_{CR}, std)$, com γ_{CR} inicializado com 0,5 e $std = 0,1$ para garantir que os valores de CR_i gerados pela distribuição normal estarão entre $(0, 1]$. Como cada estratégia pode apresentar diferentes desempenhos para cada valor de CR, Qin et al. (2009) propuseram um esquema de adaptação que considera a estratégia de mutação utilizada. Assim, para a k-ésima estratégia, o valor de $\gamma_{CR,k}$ é inicializado como 0,5. Um conjunto de valores de CR é aleatoriamente gerado de acordo com $CR_i = \text{randn}_i(\gamma_{CR,k}; 0, 1)$ e aplicado a cada indivíduo da população atual correspondente à k-ésima estratégia.

São utilizadas memórias $CRMem_k$ para armazenar os valores de CR que obtiveram sucesso na geração de indivíduos experimentais nas últimas L gerações correspondentes ao período de aprendizagem. Durante as primeiras L gerações, os valores de CR_i correspondentes à k-ésima estratégia são gerados por $CR_i = \text{randn}_i(\gamma_{CR,k}; 0, 1)$. A cada geração após o período L, o valor mediano armazenado em $CRMem_k$ deve ser calculado e sobrescrever $\gamma_{CR,k}$. Após a avaliação dos indivíduos experimentais recém gerados, os valores de CR em $CRMem_k$ que correspondem às gerações anteriores são substituídos por valores de CR promissores obtidos na geração atual com relação à k-ésima estratégia.

Nota-se que o esquema adaptativo proposto no SaDE elimina a necessidade de ajuste dos parâmetros F e CR, mas introduz o desvio padrão na distribuição normal. Qin et al. (2009) argumentam que ainda assim essa abordagem é interessante, pois o ajuste dos parâmetros F e CR é muito mais sensível do que aqueles normalmente presentes em estratégias adaptativas.

CoDE

Wang et al. (2011) propuseram o método *Composite Differential Evolution* (CoDE), que utiliza três estratégias de mutação e três conjuntos de configuração de parâmetros para F e CR que são aleatoriamente combinados para a geração de indivíduos experimentais. Com a escolha de diferentes estratégias e configurações de parâmetros, aumenta-se a probabilidade de encontrar boas atribuições de valores para instâncias com diferentes características.

No trabalho que propõe o CoDE, o conjunto de estratégias utilizadas é composto por DE/rand/1/bin, DE/rand/2/bin e DE/current-to-rand/1. Vale salientar que o operador de cruzamento não é aplicado na última estratégia. As três configurações de parâmetros utilizadas são $(F = 1,0; CR = 0,1)$, $(F = 1,0; CR = 0,9)$ e $(F = 0,8; CR = 0,2)$.

A cada geração, cada estratégia do conjunto é utilizada para criar um novo indivíduo experimental com uma configuração de parâmetros de controle aleatoriamente selecionada do conjunto de configurações de parâmetros candidatas. Assim, são gerados três indivíduos experimentais para cada indivíduo da população atual. O melhor dos indivíduos experimentais é selecionado para a próxima geração apenas se for melhor que o indivíduo correspondente na população atual.

Foi realizada uma extensa avaliação experimental que indicou o bom desempenho do CoDE. No entanto, vale observar que a criação de três indivíduos experimentais para cada indivíduo da população atual pode não ser adequada em cenários em que o número de avaliações de *fitness* é restrito.

C²oDE

Wang et al. (2018) propuseram uma extensão do CoDE (Wang et al., 2011) para instâncias de otimização com restrições denominada C²oDE. Esse novo algoritmo também utiliza um conjunto de estratégias de geração de indivíduos experimentais durante o processo evolutivo. Para o tratamento de restrições é empregada uma combinação entre o método de Superioridade de Soluções Factíveis (SOF) (Deb, 2000) e o método *ε-constrained* (Takahama e Sakai, 2010), ambos comentados na Seção 2.2.2.

Especificamente, o método SOF é utilizado para selecionar o melhor dos três indivíduos experimentais gerados para cada indivíduo da população atual. Em seguida, o método *ε-constrained* é utilizado para a operação de seleção entre o indivíduo experimental e o indivíduo correspondente na população atual. Essa abordagem foi empregada a fim de buscar um equilíbrio entre exploração local e global do espaço de busca, pois o método SOF favorece a convergência enquanto o *ε-constrained* promove a diversidade.

O conjunto de estratégias de geração de indivíduos experimentais do C²oDE foi alterado em relação ao CoDE, pois na versão anterior não existia uma estratégia que buscasse promover a exploração local dos indivíduos. Esse conjunto é composto por DE/current-to-rand/1, DE/current-to-best/1 e DE/rand-to-best/1 modificada. A fim de manter um equilíbrio entre restrições e função objetivo, os autores utilizaram uma definição modificada de melhor indivíduo da população nas duas estratégias que utilizam essa informação. Especificamente, o melhor indivíduo da estratégia DE/rand-to-best/1 modificada é o indivíduo da população atual com menor grau de violação de restrições. Por outro lado, o melhor indivíduo na estratégia DE/current-to-best/1 é aquele com melhor valor de função objetivo na população atual.

Por fim, foi adotado um mecanismo de reinicialização que visa auxiliar o algoritmo quando o processo de busca fica estagnado em uma região infactível, o que acaba sendo interessante principalmente para instâncias com espaços de busca complexos. Se o desvio padrão da violação de restrições ou o desvio padrão dos valores de função objetivo forem menores do

que um limite predefinido e se toda a população for infactível, todos os indivíduos da população são aleatoriamente distribuídos no espaço de busca.

No estudo original (Wang et al., 2018), o C²oDE foi avaliado sobre as instâncias do CEC2006 (Liang et al., 2006) e do CEC2010 (Mallipeddi e Suganthan, 2010) com 10 a 30 dimensões, tendo demostrado um desempenho bastante competitivo se comparado a outros algoritmos evolutivos recentes para otimização restrita. Posteriormente, em Trivedi e Srinivasan (2018) o C²oDE foi avaliado sobre as instâncias pertencentes ao CEC2017 (Wu et al., 2017). Foi verificado que o desempenho do algoritmo se deteriora significativamente em instâncias com mais de 50 dimensões. Além disso, identificou-se que o mecanismo de reinicialização não melhora o desempenho do algoritmo nesse conjunto de instâncias.

Operadores de Mutação Baseados em *Ranking*

Conforme comentado na Seção 2.2.1, geralmente a operação de mutação em DE utiliza indivíduos selecionados aleatoriamente na população atual. Gong e Cai (2013) propuseram um esquema em que alguns dos indivíduos nas operações de mutação são selecionados de acordo com seu *ranking* na população atual.

A fim de utilizar as informações de boas soluções da população, é estabelecida uma classificação para cada indivíduo de acordo com seu valor de *fitness*. Inicialmente, a população é ordenada de forma descendente, ou seja, do melhor indivíduo para o pior. Então, para o i-ésimo indivíduo, o valor de classificação é definido por $R_i = NP - i$, em que NP é o tamanho da população. A probabilidade de seleção do i-ésimo indivíduo é estabelecida como $s_i = R_i/NP$.

Gong e Cai (2013) sugerem que a operação de mutação pode ser melhor sucedida se o indivíduo base e o indivíduo terminal forem selecionados com base em sua probabilidade de seleção, enquanto que o terceiro indivíduo é selecionado aleatoriamente, como no algoritmo DE original. Assim, na estratégia DE/rand/1 (definida na Equação 2.4 por $\mathtt{v}_i = \mathtt{r}_1 + F \cdot (\mathtt{r}_2 - \mathtt{r}_3)$), somente $\mathtt{r}_1$ (indivíduo base) e $\mathtt{r}_2$ (indivíduo terminal) são escolhidos a partir da seleção baseada em *ranking*. No entanto, para evitar convergência prematura, cada indivíduo tem apenas uma chance de ser selecionado como base ou terminal em uma mesma geração.

A facilidade de implementação dessa estratégia é um ponto a ser destacado, assim como o fato de não serem adicionados novos parâmetros ao algoritmo. Em Gong e Cai (2013) foram realizados experimentos extensivos que indicam que o operador de mutação baseado em *ranking* é capaz de melhorar a busca local em torno dos indivíduos da população atual e, consequentemente, o desempenho de diferentes variantes de DE.

SHADE

Tanabe e Fukunaga (2013) propuseram uma técnica de adaptação de parâmetros para DE denominada *Success-History based Adaptive Differential Evolution* (SHADE). Essa estratégia, que pode ser considerada um aprimoramento do método JADE (Zhang e Sanderson, 2009), utiliza um histórico de memórias de configurações de parâmetros bem-sucedidas para guiar a escolha dos valores de parâmetros de gerações futuras.

O método SHADE mantém um histórico de memórias de tamanho H para ambos os parâmetros de controle F e CR, definidos como MF e MCR, respectivamente. Ao início da execução do algoritmo, os valores de MF_h e MCR_h ($h = 1, \ldots, H$) são inicializados como $0,5$. A cada geração, os parâmetros utilizados para cada indivíduo são gerados a partir de memórias escolhidas no histórico seguindo índices uniformemente distribuídos e aleatoriamente selecionados em $[1, H]$. Caso os valores gerados estejam fora do intervalo $(0, 1]$, são gerados novamente ou ajustados da mesma forma que no JADE (Zhang e Sanderson, 2009).

Os valores de F_i e CR_i que obtiveram sucesso na geração de um indivíduo experimental melhor que o indivíduo $\mathtt{x}_i$ da população atual são armazenados em SF e SCR. Ao fim de cada geração as memórias são atualizadas conforme as Equações 3.6 e 3.7:

$$MF_{h,g+1} = \begin{cases} \text{média}_{WL}(SF) & \text{se } SF \neq \emptyset \\ MF_{h,g} & \text{caso contrário,} \end{cases} \tag{3.6}$$

$$MCR_{h,g+1} = \begin{cases} \text{média}_{WA}(SCR) & \text{se } SCR \neq \emptyset \\ MCR_{h,g} & \text{caso contrário,} \end{cases} \tag{3.7}$$

sendo que o índice h indica a posição da memória a ser atualizada. No início da execução do algoritmo, h é definido como 1 e incrementado a cada inserção de um novo elemento no histórico. Esse histórico de memória é circular, ou seja, se $h > H$, então h é redefinido como 1. Dessa forma, na geração g o h-ésimo elemento da memória é atualizado. Conforme as Equações 3.6 e 3.7, quando todos os indivíduos da geração g falham ao gerar um indivíduo experimental melhor que o indivíduo correspondente da população atual, ou seja, $SF = SCR = \emptyset$, a memória não é atualizada. As Equações 3.8 a 3.11 são utilizadas para a atualização das memórias:

$$\text{média}_{WL}(SF) = \frac{\sum_{t=1}^{|SF|} w_t \cdot SF_t^2}{\sum_{t=1}^{|SF|} w_t \cdot SF_t}, \tag{3.8}$$

$$\text{média}_{WA}(SCR) = \sum_{t=1}^{|SCR|} w_t \cdot SCR_t, \tag{3.9}$$

$$w_t = \frac{\Delta f_t}{\sum_{t=1}^{|SCR|} \Delta f_t}, \tag{3.10}$$

sendo Δf_t a variação do valor da função objetivo entre o indivíduo atual da população $\mathtt{x}_g$ e o indivíduo experimental $\mathtt{u}_g$, conforme definido na Equação 3.11:

$$\Delta f_t = \left| f(\mathtt{u}_{t,g}) - f(\mathtt{x}_{t,g}) \right|. \tag{3.11}$$

Conforme comentado anteriormente nesta mesma seção, em JADE (Zhang e Sanderson, 2009) foi inserido o parâmetro p, utilizado para ajustar a estratégia de mutação DE/current-to-$p$best/1. Esse parâmetro, que no JADE é estático e definido manualmente, foi adaptado no SHADE para que cada indivíduo $\mathtt{x}_i$ tenha um p_i associado, ajustado a cada geração conforme a Equação 3.12:

$$p_i = \text{rand}[p_{min}; 0,2], \tag{3.12}$$

com p_{min} definido de tal forma que, quando o indivíduo $p$best é selecionado, pelo menos dois indivíduos são selecionados, ou seja, $p_{min} = 2/NP$. O valor máximo $0,2$ na Equação 3.12 é o valor máximo do intervalo de p sugerido em Zhang e Sanderson (2009).

Assim como o JADE, o SHADE utiliza um arquivo externo opcional. A cada geração, os parâmetros F e CR são definidos a partir do esquema de adaptação baseado em histórico, indivíduos experimentais são gerados, a operação de seleção é aplicada e o histórico de memória (MF, MCR) é atualizado. Esse processo se repete até que algum critério de parada seja satisfeito.

L-SHADE

Sabe-se que o tamanho da população desempenha um papel significativo no controle do processo de convergência em algoritmos evolutivos. Populações pequenas tendem a convergir mais rapidamente, mas aumentam o risco de convergência para um ponto ótimo local diferente do ótimo global do espaço de busca. Grandes populações tendem a explorar melhor o espaço de busca, mas a convergência tende a ser mais lenta. Pensando nisso, Tanabe e Fukunaga (2014) propuseram uma extensão do método SHADE (Tanabe e Fukunaga, 2013) denominada L-SHADE, que utiliza redução linear contínua do tamanho da população no decorrer da execução do algoritmo.

No esquema proposto no L-SHADE, a população é reduzida por meio de uma função linear em que o tamanho da população ao início da execução é NP^{init} e o tamanho da população ao fim da execução é NP^{min}. Após cada geração g, o tamanho da população na próxima geração NP_{g+1} é calculado de acordo com a Equação 3.13:

$$NP_{g+1} = \text{round}\left[\left(\frac{NP^{min} - NP^{init}}{MaxFEs}\right) \cdot FE + NP^{init}\right] \tag{3.13}$$

em que NP^{min} é definido como o menor valor possível que garanta que os operadores de mutação empregados possam ser aplicados. Por exemplo, no L-SHADE é definido $NP^{min} = 4$ porque a estratégia de mutação utilizada DE/current-to-$p$best/1 (Equação 3.1) necessita de 4 indivíduos. FE é o número de avaliações de *fitness* atual e $MaxFEs$ é o número máximo de avaliações de *fitness*. Sempre que $NP_{g+1} < NP_g$, os piores indivíduos são removido da população.

Vale salientar a facilidade de implementação desse esquema de redução linear do tamanho da população, assim como o fato de que não são adicionados novos parâmetros ao algoritmo. O método L-SHADE melhorou significativamente os resultados obtidos pelo SHADE e embasou diversas abordagens recentes da literatura da área de otimização de parâmetros reais, como pode ser observado na Seção 3.2.

3.2 ALGORITMOS DA COMPETIÇÃO CEC2018

Organizada durante o *IEEE Congress on Evolutionary Computation (CEC)*, a competição *Special Session & Competitions on Real-Parameter Single Objective Optimization* introduziu ambientes experimentais específicos para avaliação e comparação de algoritmos de busca estocástica de última geração. Realizada em 2006, 2010, 2017 e 2018, a competição tem reunido pesquisadores que utilizam técnicas com desempenhos promissores para resolver instâncias com diferentes características.

Ainda que não seja possível afirmar que os algoritmos participantes da competição possuam os melhores resultados da literatura, por meio de sua análise é possível observar o comportamento de diferentes estratégias nesse cenário de otimização. A fim de gerar o conhecimento necessário para a correta compreensão da proposta deste trabalho, os algoritmos pertencentes à competição do CEC2018 (Suganthan, 2018) na categoria de otimização restrita serão brevemente discutidos nesta seção.

Os algoritmos participantes da competição foram avaliados por meio de um conjunto de 28 funções ($D = \{10, 30, 50, 100\}$) pertencentes ao *benchmark* proposto por Wu et al. (2017). É importante notar que essas funções utilizam conceitos de otimização caixa-preta, ou seja, os detalhes da estrutura analítica das instâncias são desconhecidos. Como critérios de avaliação, são considerados aspectos como a quantidade de soluções factíveis, o valor médio de violação de

restrições e o valor médio de função objetivo encontrado para cada instância. A classificação geral dos algoritmos na competição de 2018 foi:

1º IUDE (2018);

2º εMAg-ES (2018);

3º LSHADE-IEpsilon (2018);

4º LSHADE44 (2017);

5º UDE (2017);

6º LSHADE44+IDE (2017);

7º CAL-SHADE (2017).

Os algoritmos participantes da mesma competição em 2017 tornaram a competir em 2018, por esse motivo o ano em que eles foram propostos é indicado. A fim de facilitar a compreensão das particularidades de cada algoritmo, optou-se por apresentá-los de forma a agrupar aqueles que utilizam estratégias semelhantes.

CAL-SHADE

O algoritmo *Constraint Handling with Success History Adaptive Differential Evolution* (CAL-SHADE) foi proposto por Zamuda (2017) e consiste em uma adaptação do método L-SHADE (Tanabe e Fukunaga, 2014), apresentado na Seção 3.1, para instâncias restritas de parâmetros reais.

A operação de seleção do DE entre o indivíduo experimental $\mathtt{u}_i$ e o indivíduo correspondente na população atual $\mathtt{x}_i$ é realizada com base no método ε*-constrained* (Takahama e Sakai, 2005), comentado na Seção 2.2.2. O valor ε é atualizado até um dado número limite de gerações e, quando esse limite é excedido, o valor ε é definido como zero a fim de obter indivíduos com o mínimo de violação de restrições.

Dessa forma, o valor médio de violação de restrições é utilizado para comparar os dois indivíduos. O indivíduo experimental $\mathtt{u}_i$ é selecionado para a próxima geração se sua violação média for menor que a violação média do indivíduo correspondente na população atual $\mathtt{x}_i$ ou se $\mathtt{x}_i$ possui violação de restrições igual a zero e valor de função objetivo maior que o valor de função objetivo do indivíduo experimental, conforme definido na Equação 2.6.

No trabalho em que foi proposto, o CAL-SHADE foi avaliado sobre as 28 instâncias do *benchmark* CEC2017 (Wu et al., 2017) com 10, 30, 50 e 100 dimensões e apresentou bons resultados. Os autores apontam como trabalhos futuros a possibilidade de testar essa abordagem com outras técnicas de manipulação de restrições, além de aplicações em outros domínios.

LSHADE44

O algoritmo LSHADE44 também baseia-se no L-SHADE (Tanabe e Fukunaga, 2014), apresentado na Seção 3.1, e foi originalmente desenvolvido por Poláková et al. (2016a) para otimização restrita limitada[1] e adaptado por Poláková (2017) para instâncias de problemas de otimização restrita.

[1] *Bound-constrained optimization*

A principal melhoria do LSHADE44 em relação ao L-SHADE é a aplicação de um conjunto de quatro tipos de mutações diferentes que competem entre si a fim de melhorar o desempenho do processo de busca. São utilizadas as operações de mutação DE/current-to-*p*best/1 e a mutação de localização aleatória DE/randrl/1, ambas combinadas com os operadores de cruzamento binomial e exponencial. Dessa forma, as quatro estratégias empregadas no algoritmo LSHADE44 são: DE/current-to-*p*best/1/bin, DE/current-to-*p*best/1/exp, DE/randrl/1/bin e DE/randrl/1/exp.

O mecanismo de competição entre estratégias utilizado foi proposto por Tvrdık (2006). Tem-se K diferentes estratégias para criação de um indivíduo experimental, cada uma com probabilidade q_l de ser escolhida, tal que no início da busca $q_l = 1/K$ para todo $l = 1, 2, \ldots, K$. Então, as probabilidades são adaptadas de acordo com seu sucesso ao gerar um indivíduo experimental melhor que o atual, conforme mostra a Equação 3.14.

$$q_l = \frac{n_l + n_0}{\sum_{k=1}^{K}(n_k + n_0)}, \tag{3.14}$$

sendo que n_l é a contagem atual de sucessos da l-ésima estratégia e $n_0 > 0$ é uma constante que atenua a influência de um sucesso aleatório da l-ésima estratégia em q_l. Para evitar a degeneração do processo de busca, se alguma probabilidade q_l diminui além de um dado limite durante o processo evolutivo, os valores atuais de q_l são redefinidos para seus valores iniciais $q_l = 1/K$.

No algoritmo L-SHADE original (Seção 3.1) é utilizado um par de memórias MF e MCR para adaptação dos parâmetros F e CR. No LSHADE44 esse esquema de adaptação é realizado da mesma forma, mas são empregados quatro pares de memórias, um para cada estratégia. O sufixo "44" no nome do algoritmo vem da junção entre esses quatro pares de memórias e as quatro estratégias utilizadas. Como o LSHADE44 é utilizado para resolver instâncias de otimização com restrições, quando é necessário ordenar os indivíduos da população é realizado um procedimento em duas etapas. Primeiramente, os indivíduos são ordenados pela função objetivo e então pela violação média de restrições.

O algoritmo LSHADE44 foi avaliado com o auxílio do *benchmark* CEC2017 (Wu et al., 2017) e apresentou bons resultados, o que indica que a adaptação do algoritmo para instâncias de otimização restritos associada à competição entre estratégias é uma boa alternativa para atacar esse tipo de instância.

LSHADE-IEpsilon

Fan et al. (2018) propuseram uma versão melhorada do método de manipulação de restrições ε-*constrained* associada ao algoritmo LSHADE44. O método IEpsilon ajusta adaptativamente o valor de ε de acordo com a proporção de indivíduos factíveis na população atual, o que acaba por balancear a busca entre regiões factíveis e infactíveis durante o processo de otimização. Adicionalmente, um novo operador de mutação DE/rand1*/1 é proposto. Um esquema de definição do valor ε melhorado foi sugerido de acordo com a Equação 3.15:

$$\varepsilon(g) = \begin{cases} \phi_\theta & \text{se } g = 0 \\ \varepsilon(g-1)(1 - \frac{FEs}{T})^{cp} & \text{se } r_g < \alpha \text{ e } FEs < T \\ (1+\tau)\phi_{max} & \text{se } r_g \geq \alpha \text{ e } FEs < T \\ 0 & \text{se } FEs \geq T, \end{cases} \tag{3.15}$$

sendo $\varepsilon(g)$ o valor de ε na geração g, r_g é a proporção de soluções factíveis e ϕ_θ é o valor da violação geral das restrições do θ-ésimo melhor indivíduo da população inicial. T define um

limite de avaliações de *fitness FEs* até onde o valor de ε é ajustado. A partir desse ponto, ε é redefinido como 0 a fim de buscar indivíduos com violação mínima de restrições. O parâmetro cp controla a velocidade de redução do relaxamento das restrições, τ é definido como $0,1$ pelos autores e α é um limiar para controle de mudanças em ε definido como $0,5$.

Quando a proporção de indivíduos factíveis na geração atual é maior que α, a população é considerada rica em indivíduos factíveis e o ε aumenta gradualmente com o incremento das gerações, o que faz a população buscar regiões infactíveis do espaço de busca. Por outro lado, quando a proporção é menor que α a população é considerada pobre em indivíduos factíveis e o ε decai com o incremento das gerações, o que faz com que a população tenda a buscar mais indivíduos factíveis. Dessa forma, a principal diferença entre o IEpsilon e o *ε-constrained* é essa capacidade de crescimento do valor ε, pois na versão original esse valor somente decresce.

As estratégias utilizadas no LSHADE44-IEpsilon são DE/current-to-*p*best/1/bin, DE/current-to-*p*best/1/exp, DE/randr1*/1/bin e DE/randr1*/1/exp. A estratégia DE/randr1*/1 (Equação 3.16) foi proposta nesse trabalho e utiliza o indivíduo atual para substituir um indivíduo pai aleatório, o que pode reduzir a diferença entre o indivíduo experimental e o indivíduo atual.

$$\mathrm{v}_i = \mathrm{r}_1^* + F \cdot (\mathrm{r}_2^* - \mathrm{r}_3^*), \tag{3.16}$$

sendo v_i indivíduo mutante do indivíduo atual x_i na geração g. Sejam r_1 e r_2 dois indivíduos distintos uniformemente escolhidos do conjunto $\mathrm{p}_g \setminus \{\mathrm{x}_i\}$. $\mathrm{r}_1^* \in \{\mathrm{r}_1, \mathrm{r}_2, \mathrm{x}_i\}$ define r_1^* como o melhor indivíduo dentre os três, e r_2^* e r_3^* são os dois indivíduos remanescentes $\{\mathrm{r}_1, \mathrm{r}_2, \mathrm{x}_i\} \setminus \mathrm{r}_1^*$.

Em sua apresentação original (Fan et al., 2018), o algoritmo LSHADE44-IEpsilon foi avaliado sobre as 28 instâncias propostas no *benchmark* CEC2017 (Wu et al., 2017) com quatro dimensões diferentes (D = 10, 30, 50, 100). Os resultados obtidos foram comparados a outras quatro variações recentes de DE (CAL-SHADE, LSHADE44+IDE, LSHADE44 e UDE) em 25 execuções independentes. Os resultados experimentais mostraram que o LSHADE44-IEpsilon tem o melhor desempenho nessas 28 funções para todos os números de dimensões analisados.

LSHADE44+IDE

Tvrdík e Poláková (2017) propuseram o algoritmo LSHADE44+IDE, um *framework* simples para cooperação entre duas variantes adaptativas de DE para resolução de instâncias de problemas de otimização restritos. No LSHADE44+IDE o processo de busca é dividido em duas etapas. Para implementação da primeira fase do algoritmo foi utilizado o LSHADE44 (Poláková et al., 2016b) e no segundo estágio foi aplicada a variação de DE adaptativa *individual-dependent mechanism* (IDE) (Tang et al., 2015).

Na primeira etapa do processo de otimização, a busca por indivíduos factíveis é realizada pela minimização da violação média de restrições. Esse estágio é interrompido se for encontrado um número previamente definido de indivíduos factíveis ou se o número de avaliações de *fitness* designado for consumido. O processo de busca continua na segunda etapa, em que o valor da função objetivo é minimizado até que uma condição de parada seja atendida. Se uma quantidade suficiente de indivíduos factíveis for encontrada no primeiro estágio, esses indivíduos são utilizados como população inicial para o segundo estágio. Caso contrário, os indivíduos com menor violação média são empregados como população inicial para a segunda etapa.

Os autores explicam que qualquer variante de DE pode ser utilizada na primeira etapa, incluindo aquelas que adaptam o tamanho da população. No entanto, como pode haver um número limitado de indivíduos factíveis na primeira fase, variantes de DE com tamanho fixo de população são uma escolha mais conveniente para a segunda etapa.

A operação de seleção tradicionalmente utilizada foi modificada a fim de considerar os valores da função objetivo e as violações médias de restrições. Na primeira etapa, um indivíduo experimental $\mathtt{u}_i$ é selecionado se sua violação média é menor que a violação média do indivíduo atual correspondente $\mathtt{x}_i$ ou se as violações são iguais e o indivíduo experimental possui melhor valor de função objetivo. No segundo estágio as violações de restrições não são consideradas. Um indivíduo experimental $\mathtt{u}_i$ é aceito somente se o valor de sua função objetivo é menor que o valor da função objetivo do indivíduo atual correspondente $\mathtt{x}_i$ e sua violação média não é pior que um limite definido. Esse limite pode ser estabelecido, por exemplo, como a menor violação encontrada na população inicial.

O processo de busca IDE, que compõe a segunda fase do algoritmo, é dividido em duas etapas. A primeira delas é direcionada à exploração de novos indivíduos e a segunda realiza uma intensificação da busca em torno de indivíduos promissores. Dessa forma, os autores propuseram a definição dinâmica de F e CR baseada na qualidade dos indivíduos encontrados. Indivíduos com menor (maior) valor de função objetivo têm valores de F e CR menores (maiores). Todos os indivíduos da população são ordenados de forma crescente segundo o valor da função objetivo. Os valores de F e CR são calculados pelas Equações 3.17 e 3.18:

$$F_o = \frac{o}{NP}, \tag{3.17}$$

$$CR_i = \frac{i}{NP}, \tag{3.18}$$

sendo o o índice do indivíduo base utilizado na operação de mutação, i o índice do indivíduo atual $\mathtt{x}_i$ e NP o tamanho da população. Após o cálculo dos parâmetros, F e CR são modificados utilizando distribuição normal até que estejam no intervalo $(0, 1)$.

O IDE utiliza duas estratégias de mutação diferentes. Na fase inicial é selecionada uma estratégia que utiliza o indivíduo base a fim de aumentar a diversidade da população, ou seja, $o = i$. Na segunda fase é estabelecida uma estratégia de mutação em que o indivíduo base é selecionado aleatoriamente a fim de aumentar a velocidade de convergência. Os indivíduos ordenados de forma crescente pelo valor da função objetivo são divididos em dois conjuntos, superior S e inferior I. O esquema de mutação descrito é definido como na Equação 3.19:

$$\mathtt{v}_i = \begin{cases} \mathtt{x}_o + F_o \cdot (\mathtt{r}_1 - \mathtt{x}_o) + F_o \cdot (\mathtt{r}_2 - \mathtt{r}_3) & \text{se } \mathtt{x}_o \in S \\ \mathtt{x}_o + F_o \cdot (\mathtt{r}_{best} - \mathtt{x}_o) + F_o \cdot (\mathtt{r}_2 - \mathtt{r}_3) & \text{se } \mathtt{x}_o \in I, \end{cases} \tag{3.19}$$

em que $\mathtt{x}_o$, $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ são indivíduos distintos selecionados aleatoriamente de toda a população atual e $\mathtt{r}_{best}$ é um indivíduo aleatoriamente escolhido na parte superior da população atual.

O indivíduo $\mathtt{r}_3$ é alterado com uma pequena probabilidade pb a fim de auxiliar o indivíduo base a escapar de mínimos locais no espaço de busca. Essa alteração é executada conforme a Equação 3.20:

$$\mathtt{r}_{3,j} = \begin{cases} a_j + \text{rand}_j[0,1] \cdot (b_j - a_j) & \text{se rand}_j[0,1] < pb \\ \mathtt{r}_{3,j} & \text{caso contrário,} \end{cases} \tag{3.20}$$

em que $pb = (0,1 \cdot ps)$ e a_j, b_j são, respectivamente, os limites inferiores e superiores do domínio da j-ésima dimensão do indivíduo. O valor de ps é a proporção de indivíduos superiores na população atualizados de acordo com a Equação 3.21:

$$ps = 0,1 + 0,9 \cdot 10^{5(g/G-1)}, \tag{3.21}$$

sendo g a geração atual e G o número máximo de gerações.

No artigo original (Tvrdík e Poláková, 2017), o LSHADE44+IDE foi avaliado sobre o conjunto de instâncias do CEC2017 (Wu et al., 2017). O algoritmo encontrou soluções factíveis para 75% das instâncias em todas as dimensões testadas. Os autores acreditam que melhorias no primeiro estágio do algoritmo poderiam ajudar a encontrar uma quantidade maior de soluções factíveis e ainda sugerem que a combinação adaptativa de mais estratégias de busca ou reinícios controlados possivelmente melhorariam o desempenho do algoritmo.

UDE

O algoritmo *Unified Differential Evolution* (UDE), proposto por Trivedi et al. (2017), tem como principal característica a unificação de ideias de algumas variantes adaptativas de DE apresentadas na Seção 3.1, tais como CoDE (Wang et al., 2011), JADE (Zhang e Sanderson, 2009), SaDE (Qin et al., 2009) e operador de mutação baseado em *ranking* (Gong e Cai, 2013), além de um operador de busca local. O UDE utiliza um conjunto de três estratégias de mutação e dois conjuntos de configurações dos parâmetros F e CR.

O conjunto de estratégias empregadas no CoDE consiste em DE/rand/1/bin, DE/-rand/2/bin e DE/current-to-rand/1 sem o operador de cruzamento. No entanto, os autores observam que nesse algoritmo não existe uma estratégia de mutação que promova a busca local em torno dos indivíduos atuais. Por esse motivo, no lugar de DE/rand/2/bin, no UDE foi utilizada a estratégia DE/current-to-$p$best/1 sem o operador de cruzamento. Não é relatado no trabalho se o arquivo externo opcional sugerido em Zhang e Sanderson (2009) é utilizado.

Foi incorporada uma versão modificada de seleção de indivíduos pais baseada em *ranking* em que os indivíduos base e terminal utilizados na operação de mutação das três estratégias são selecionados entre os S melhores membros da população. Por exemplo, no caso da operação DE/rand/1 definida por $\mathtt{v}_i = \mathtt{r}_1 + F \cdot (\mathtt{r}_2 - \mathtt{r}_3)$, os indivíduos $\mathtt{r}_1$ (base) e $\mathtt{r}_2$ (terminal) são selecionados entre os S melhores indivíduos.

O conjunto de parâmetros utilizados no CoDE é composto pelas seguintes configurações: $(F = 1,0; CR = 0,1)$, $(F = 1,0; CR = 0,9)$ e $(F = 0,8; CR = 0,2)$. Já no UDE existem somente duas configurações possíveis: $(F = 0,9; CR = 0,9)$ e $(F = 0,5; CR = 0,5)$.

A cada geração, o UDE divide a população atual em duas subpopulações. Na subpopulação superior, que corresponde aos S melhores indivíduos, cada estratégia de mutação é aplicada ao indivíduo atual, resultando em três indivíduos experimentais que são comparados entre si. A estratégia de mutação do melhor dos três indivíduos contabiliza uma vitória.

A cada L gerações, a taxa de sucesso de cada estratégia de mutação no período é avaliada (sendo L o período de aprendizagem). Durante esse período uma estratégia de mutação é probabilisticamente empregada na parte inferior da população, que consiste em $NP - S$ indivíduos. A probabilidade de uma estratégia ser utilizada é igual a sua taxa de sucesso recente, ou seja, a taxa de sucesso obtida durante as últimas L gerações.

No UDE foi utilizado o método de penalidade estática para o tratamento de restrições. Dessa forma, um indivíduo $\mathtt{x}_i$ tem seu valor de *fitness* atribuído de acordo com a Equação 3.22:

$$F(\mathtt{x}_i) = f(\mathtt{x}_i) + penalidade \cdot \phi(\mathtt{x}_i), \tag{3.22}$$

em que $f(\mathrm{x}_i)$, $\phi(\mathrm{x}_i)$, $F(\mathrm{x}_i)$ são, respectivamente, o valor da função objetivo, o valor total de violação de restrições e o valor da função de *fitness* do indivíduo x_i na geração g e *penalidade* é o fator estático de penalidade.

No lugar da operação tradicional de seleção do DE, os autores utilizaram a estratégia de substituição geracional. Assim, a população de indivíduos experimentais u_i e a população atual composta pelos indivíduos x_i são combinadas e os *NP* melhores indivíduos são selecionados para a próxima geração.

Por fim, os autores associaram o DE a um operador de busca local. A cada f_{bl} gerações, uma estratégia de mutação sem o operador de cruzamento é executada sobre os NP_{bl} melhores indivíduos da população atual, exceto no melhor, conforme a Equação 3.23:

$$\mathrm{v}_{bl,i} = \mathrm{x}_i + F_{bl} \cdot (\mathrm{x}_{best} - \mathrm{r}_1) + F_{bl} \cdot (\mathrm{r}_2 - \mathrm{r}_3), \tag{3.23}$$

sendo $\mathrm{v}_{bl,i}$ o indivíduo gerado pela busca local sobre o i-ésimo indivíduo da população atual (x_i). x_{best} é o melhor indivíduo da população atual, F_{bl} é o fator de mutação usado na operação de busca local, r_1 e r_3 são indivíduos selecionados aleatoriamente dentre todos os membros da população atual e r_2 é selecionado aleatoriamente na subpopulação S.

Dentre os N_{bl} indivíduos gerados por meio da busca local, o indivíduo v_{bl} com melhor valor de *fitness* é comparado com o melhor indivíduo x_{best} da população atual. Se v_{bl} for melhor que x_{best}, então o pior indivíduo x_w na população atual é substituído por v_{bl}, senão x_w é substituído por x_{best}. Vale observar que nesse último caso uma cópia duplicada do melhor indivíduo x_{best} é adicionada à população atual. Os autores afirmam que essa duplicação melhora o desempenho do UDE, que se mostrou bastante satisfatório quando avaliado sobre as funções do CEC2017 (Wu et al., 2017).

IUDE

O algoritmo *Improved Unified Differential Evolution* (IUDE), proposto por Trivedi et al. (2018), é uma versão melhorada do UDE (Trivedi et al., 2017). O IUDE foi desenvolvido com base nas variantes de DE: CoDE, JADE, SaDE, operador de mutação baseado em *ranking*, SHADE e C²oDE, discutidos na Seção 3.1.

Assim como no UDE, o conjunto de estratégias empregadas no IUDE é composto por DE/rand/1/bin, DE/current-to-rand/1 e DE/current-to-*p*best/1. Na última estratégia foi utilizado arquivo externo, proposto por Zhang e Sanderson (2009), em instâncias com mais de 50 dimensões para armazenar indivíduos recentemente explorados a fim de fornecer informações adicionais sobre direções promissoras do espaço de busca.

Trivedi et al. (2018) afirmam que o operador de cruzamento exponencial apresenta melhores resultados que o cruzamento binomial em instâncias de otimização contínua com muitas dimensões. Assim, IUDE emprega cruzamento binomial para instâncias com menos de 100 dimensões e cruzamento exponencial em instâncias com maior dimensionalidade.

O algoritmo IUDE utiliza o mesmo esquema de subpopulações empregado no UDE. A subpopulação superior utiliza a abordagem de tratamento de restrições do C²oDE (Wang et al., 2018), comentada na Seção 3.1. Assim, o método de Superioridade de Soluções Factíveis (SOF) (Deb, 2000) é utilizado para seleção entre os três indivíduos experimentais u_i, enquanto que o método *ε-constrained* (Takahama e Sakai, 2010) é empregado para seleção entre o melhor indivíduo experimental e o indivíduo correspondente na população x_i. Como na subpopulação inferior é gerado apenas um indivíduo experimental u_i correspondente a cada indivíduo atual x_i, é aplicado somente o método *ε-constrained*.

O esquema de adaptação de parâmetros no estilo proposto no LSHADE44 (Poláková, 2017) é utilizado. Considerando que o IUDE possui um conjunto de três estratégias, é empregado um par de memórias *MF* e *MCR* para adaptação dos parâmetros *F* e *CR* correspondentes a DE/rand/1 e DE/current-to-*p*best/1, e *MF* para adaptação do valor de *F* correspondente a DE/current-to-rand/1. Vale observar que somente a subpopulação superior é utilizada para a adaptação de parâmetros, pois somente ela utiliza todas as três estratégias de mutação. É empregada a mesma estratégia de seleção de indivíduos baseada em *ranking* para a operação de mutação utilizada no UDE.

Ao início de cada geração, os membros da população (compreendendo os membros de ambas as subpopulações) são ordenados de acordo com o método SOF (Deb, 2000). Essa etapa permite a migração dos melhores indivíduos da subpopulação *B* para a subpopulação *A*, e a migração dos piores indivíduos da subpopulação *A* para a subpopulação *B*. Com isso, a estratégia de mutação no estilo CoDE sempre será implementada na metade superior da população. Essa característica tende a promover uma busca local em torno dos melhores indivíduos e levar a uma convergência mais rápida.

A seleção de indivíduos candidatos para a próxima geração é realizada por meio do método *ε-constrained*. Segundo os autores, já que o IUDE utiliza uma combinação entre o método *ε-constrained* e SOF, poderia haver perda de boas atribuições de valores ao longo do processo de evolução. Dessa forma, foi adicionado um arquivo externo de população que é atualizado com o melhor indivíduo em relação a SOF ao fim de cada geração.

Os operadores de busca local e duplicação foram removidos no IUDE, uma vez que resultavam em convergência prematura do processo de busca em algumas instâncias. A substituição geracional da população utilizada no UDE deu lugar à estratégia tradicional de seleção entre dois indivíduos, visto que esta apresentou melhores resultados.

εMAg-ES

Hellwig e Beyer (2018) propuseram um método que, diferentemente do restante dos algoritmos de competições passadas do CEC, não baseia-se em DE. Trata-se do εMAg-ES, uma combinação de técnicas de manipulação de restrições que obtiveram sucesso em variações da estratégia evolutiva *Matrix Adaptation Evolution Strategy* (MA-ES) (Beyer e Sendhoff, 2017).

Em uma Estratégia Evolutiva (ES), novos indivíduos candidatos são amostrados de acordo com uma distribuição normal multivariada em $\mathbb{R}^D$, com D sendo o número de dimensões da instância. Assim como outros algoritmos evolutivos, o processo de busca ocorre em ciclos a partir de recombinações e mutações em uma população que evolui em direção ao ponto ótimo global da instância. A cada ciclo uma nova população é amostrada seguindo uma determinada distribuição. A recombinação de indivíduos equivale a selecionar um novo valor médio para a distribuição. A operação de mutação equivale a adicionar um vetor com uma perturbação aleatória com média zero aos indivíduos amostrados. As dependências entre pares de variáveis da instância na distribuição são representadas por uma matriz de covariância. A atualização da matriz de covariância dessa distribuição é feita pelo método *Covariance Matrix Adaptation* (CMA) (Hansen, 2006).

O εMAg-ES trata restrições de borda (limite superior e inferior das variáveis) por meio de estratégias de correção, conforme explicado na Seção 2.2.2. De maneira análoga ao DE, os indivíduos criados a cada geração devem ser classificados. Indivíduos factíveis são considerados superiores a indivíduos infactíveis. É utilizado o método *ε-constrained* para classificação dos indivíduos candidatos. Além disso, o εMAg-ES utiliza uma abordagem de vizinhanças baseada em gradientes. A aplicação dessas duas técnicas foi inspirada no εDEga (Takahama e Sakai,

2010), vencedor da competição CEC2010 em otimização restrita de parâmetros reais (Mallipeddi e Suganthan, 2010).

O algoritmo εMAg-ES foi avaliado com base no *benchmark* CEC2017 (Wu et al., 2017). Independentemente do número de dimensões, a estratégia foi capaz de encontrar soluções factíveis para 20 das 28 instâncias abordadas. Se comparado ao LSHADE44, vencedor da competição do CEC2017, o εMAg-ES apresentou resultados superiores, especialmente nas instâncias mais difíceis.

3.3 CONSIDERAÇÕES

Neste capítulo foram elencadas algumas das variantes de DE que utilizam abordagens adaptativas mais recentes. Por meio de sua utilização, parâmetros como a estratégia de mutação, F e CR são ajustados automaticamente, o que torna o algoritmo mais facilmente utilizável e robusto.

Além disso, foram apresentados os algoritmos participantes da *Special Session & Competitions on Real-Parameter Single Objective Optimization* do CEC2018. A partir da observação de suas principais características e particularidades, foi possível identificar algumas técnicas promissoras para abordagem de instâncias de problemas de otimização restrita.

Conforme pode ser observado, alguns participantes da competição CEC2017 concorreram novamente no CEC2018. No entanto, essas estratégias foram superadas pelos novos algoritmos. Vale ressaltar que cinco participantes empregam de alguma forma as técnicas propostas no L-SHADE (Tanabe e Fukunaga, 2014). Além disso, identificou-se uma tendência à utilização do método ε-*constrained* (Takahama e Sakai, 2005) para manipulação de restrições. No próximo capítulo será descrita a proposta deste trabalho e os detalhes de sua implementação.

4 EVOLUÇÃO DIFERENCIAL INTERVALAR

Neste capítulo é descrito o algoritmo proposto, denominado InDE (do inglês: *Interval Differential Evolution*). O objetivo deste trabalho é avaliar a utilização de informação estrutural de uma instância NCOP no decorrer do processo de busca. Para isto, o InDE utiliza uma nova representação para os indivíduos: durante o ciclo de operações, somente as variáveis de V_Ω de uma decomposição Epífita são consideradas (Seção 2.1.2). Além disso, diferentemente do DE clássico, a cada uma dessas variáveis é atribuído um intervalo de valores reais, ou seja, um indivíduo é uma atribuição de intervalos para as variáveis de V_Ω (uma caixa).

A população é um conjunto de caixas que cobre parte do espaço de busca, em vez de somente pontos específicos como no DE clássico. Isso permite a utilização do contrator GAC (Seção 2.1.1) para podar atribuições de valores não factíveis. O mecanismo de consistência local utilizado no OGRe (Derenievicz, 2018b) é empregado para computar o valor de *fitness* de cada indivíduo.

A abordagem proposta no InDE utiliza adaptações intervalares das principais características das variantes de DE participantes das últimas edições das competições do CEC em otimização restrita de parâmetros reais. Foram utilizadas especialmente técnicas do IUDE e do LSHADE44 (apresentados na Seção 3.2), primeiros lugares no CEC2018 e CEC2017, respectivamente.

Este capítulo se divide em quatro seções:

4.1. **Aspectos Gerais**: Traz as principais características da implementação do algoritmo InDE e as justificativas para a utilização de cada componente;

4.2. **População Intervalar**: Caracteriza a população composta por indivíduos intervalares;

4.3. **Operações Intervalares**: Descreve as versões intervalares das operações tradicionais de DE e o esquema de adaptação de parâmetros utilizado;

4.4. **Avaliações de *Fitness***: Detalha o processo de atribuição do valor de função objetivo e de violação de restrições dos indivíduos da população, assim como a operação de seleção entre indivíduos.

4.1 ASPECTOS GERAIS

A partir da revisão bibliográfica realizada no Capítulo 3, identificou-se a tendência de utilização de Evolução Diferencial (DE) nesse contexto. Da mesma forma, o método de tratamento de restrições *ε-constrained* tem sido predominante nas abordagens mais recentes da literatura, especialmente nas competições do CEC2017 e CEC2018 em otimização de parâmetros reais.

A implementação do InDE foi feita na linguagem C, visto que esta foi utilizada na implementação da decomposição estrutural e consistência local do OGRe (Derenievicz, 2018a). Optou-se pela utilização de um código-base de DE disponibilizado nessa mesma linguagem por um dos autores do algoritmo original (Price e Storn, 1995). Tendo como base esse código-fonte, foram implementadas tecnologias de adaptação de parâmetros, manipulação da população e tratamento de restrições inspiradas nas abordagens do CEC.

O InDE visa manter as características de meta-heurística do DE, ou seja, fornecer uma implementação genérica capaz de abordar diferentes tipos de instâncias sem necessidade de

ajustes específicos no código. Além disso, buscou-se reduzir a quantidade de parâmetros do algoritmo por meio de esquemas adaptativos a fim de tornar sua utilização mais simples.

A implementação do OGRe considera apenas os operadores $+, -, \cdot, /, \hat{}$ e $\sqrt{}$, sendo a exponenciação e a radiciação limitados, respectivamente, a expoentes e índices inteiros. Essa limitação é estendida ao InDE, pois ele utiliza recursos do OGRe para decomposição estrutural e propagação dos valores dos intervalos pelo hipergrafo de restrições.

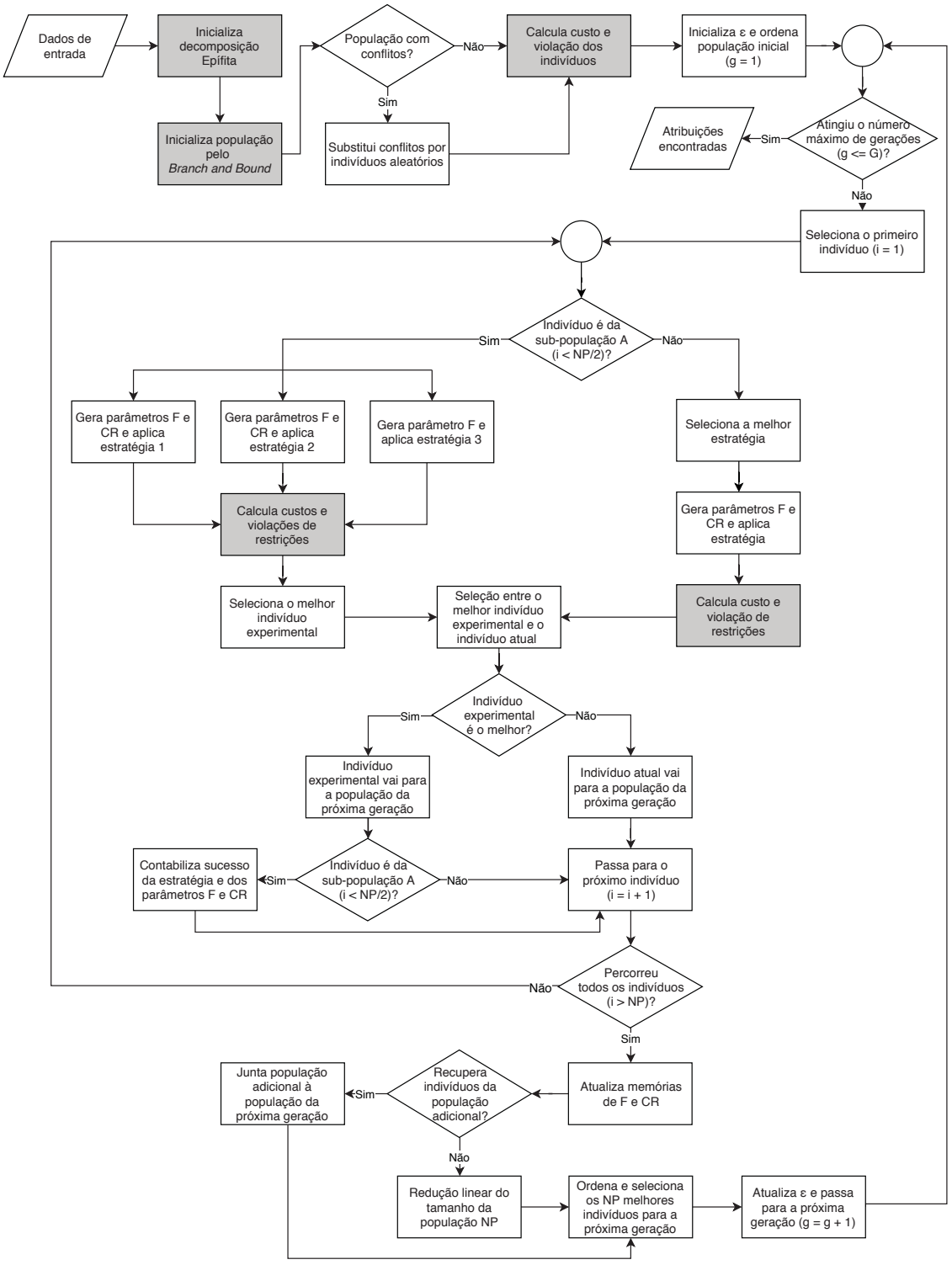


Figura 4.1: Fluxograma simplificado de execução do InDE. As etapas destacadas em cinza correspondem aos pontos em que são utilizados recursos de implementação do OGRe.

Um fluxograma simplificado da execução do InDE é ilustrado na Figura 4.1. Os itens destacados em cinza correspondem aos pontos em que os recursos de decomposição estrutural e de consistência local do OGRe são acessados pelo InDE. O processo de execução ocorre de forma semelhante ao DE clássico. Após a inicialização da decomposição Epífita da instância, é gerada a população inicial. Na sequência, é calculado o valor de custo e violação de restrições de cada indivíduo.

O InDE possui dois ciclos principais em seu fluxo de execução, sendo o primeiro relacionado às gerações do processo evolutivo e o segundo aos indivíduos sobre os quais são realizadas as operações de mutação e cruzamento. Após cada indivíduo da população ser operado, os valores dos parâmetros F e CR, o esquema de seleção de estratégia e a população atual são atualizados para a próxima geração. Cada um dos passos da execução do algoritmo será comentado em mais detalhes nas próximas seções.

4.2 POPULAÇÃO INTERVALAR

Um indivíduo intervalar é um vetor de intervalos $\mathtt{X} = (A_1, A_2, \ldots, A_D)$, sendo $A_i \subseteq X_i$ um intervalo de valores reais do domínio X_i da variável x_i, para $1 \leq i \leq D$. Uma população $\mathtt{p}$ é um conjunto de NP indivíduos intervalares e $\mathtt{p}_g$ denota a população intervalar na g-ésima geração.

O OGRe utiliza uma representação multi-intervalar do domínio das variáveis para um melhor uso do contrator GAC. Um multi-intervalo pode ser representado pela união de intervalos e uma caixa de multi-intervalos é uma maneira simples de descrever uma combinatória de intervalos. Por exemplo, a caixa de multi-intervalos $([-2,-1] \cup [1,2], [0,2])$ representa ambas as caixas $([-2,-1],[0,2])$ e $([1,2],[0,2])$. No InDE, multi-intervalos são separados em uma representação combinatória.

A população inicial é gerada pelo nível superior de ramificação da árvore do *Branch & Bound* do OGRe. Cada nodo da árvore de busca contém uma caixa de multi-intervalos consistente. Cada caixa de multi-intervalos é separada em um conjunto de caixas de intervalos (indivíduos) que são adicionados à população inicial. Os nodos são iterativamente processados até que o número de indivíduos na população seja NP. Essa estratégia garante que a população inicial cubra todo o espaço de busca.

A seguir, o valor da função objetivo e da violação de restrições de cada indivíduo são avaliados por meio da busca local sem retrocesso do OGRe (Seção 4.4). Como esse processo está sujeito a erros numéricos de arredondamento, é possível que alguns indivíduos sejam considerados inconsistentes. Nesse caso, indivíduos substitutos são gerados aleatoriamente a partir da caixa consistente inicial por meio da variação dos limites de algum intervalo. Esse processo se repete iterativamente até que a população seja composta somente por indivíduos consistentes.

Assim como no IUDE (Seção 3.2), a população atual é dividida em duas subpopulações, A e B, de tamanho $NP/2$. A população é ordenada a cada geração por meio dos valores da função objetivo e da violação de restrições de cada indivíduo. Isso permite que a subpopulação A seja composta pelos melhores $NP/2$ indivíduos de acordo com o método *ε-constrained*, descrito na Seção 2.2.2.

Esse esquema de subpopulações foi utilizado para controlar a aplicação dos operadores de mutação. Na subpopulação A é aplicado um conjunto de três estratégias de mutação. Dessa forma, são gerados três indivíduos experimentais que competem entre si e o melhor deles, $\mathtt{u}_i$, compõe a operação de seleção com o indivíduo $\mathtt{x}_i$ da subpopulação atual. Essa competição entre estratégias é computacionalmente cara, pois demanda três avaliações de *fitness* para cada

indivíduo $\mathtt{x}_i$. No entanto, intensifica-se a exploração dos melhores indivíduos da população atual, o que pode acelerar o processo de convergência. Na subpopulação *B* foi utilizado o esquema de seleção adaptativa proposto no IUDE, que escolhe uma única estratégia de mutação a ser empregada a cada indivíduo. Com isso, é gerado somente um indivíduo experimental $\mathtt{u}_i$ para cada indivíduo $\mathtt{x}_i$ da subpopulação atual.

O algoritmo DE tradicionalmente utiliza uma comparação um-para-um entre indivíduos experimentais e indivíduos atuais a fim de selecionar o melhor deles para a próxima geração. A fim de explorar direções promissoras do espaço de busca, é mantida uma população adicional com os indivíduos experimentais que não foram selecionados para a próxima geração. Da mesma forma, quando uma caixa de multi-intervalos é dividida em indivíduos intervalares, somente o melhor indivíduo experimental resultante $\mathtt{u}_i$ é comparado com o indivíduo $\mathtt{x}_i$ da população atual. Os indivíduos experimentais restantes são incluídos na população adicional. Periodicamente, a população atual (compreendendo as subpopulações *A* e *B*) e a população adicional são unidas e ordenadas. Dessa forma, a nova população é composta pelos *NP* melhores indivíduos segundo o método *ε-constrained* e a população adicional é esvaziada.

O esquema de redução linear da população proposto em LSHADE (Seção 3.1) é utilizado para reduzir o tamanho da população durante a execução do algoritmo. Tendo em vista que os indivíduos excluídos da população são sempre aqueles com a pior avaliação, esse esquema acaba por intensificar a exploração local dos melhores indivíduos da população, o que pode acelerar o processo de convergência.

Na próxima seção serão comentadas as características específicas das adaptações realizadas nas operações de mutação para o contexto intervalar.

4.3 OPERAÇÕES INTERVALARES

Assim como no IUDE, um conjunto de três operadores de mutação constituído por DE/rand/1, DE/rand-to-*p*best/1 e DE/current-to-rand/1 é utilizado no InDE. Na subpopulação superior *A*, cada uma das três estratégias é utilizada para gerar novos indivíduos experimentais, empregando três avaliações de *fitness* para cada indivíduo da subpopulação atual. Como a subpopulação *A* contém os $NP/2$ melhores membros da população atual, foi utilizado o mesmo esquema modificado de seleção baseada em *ranking* utilizado no IUDE. Então, na operação de mutação, o indivíduo base e o indivíduo terminal são selecionados dentre os indivíduos da subpopulação *A*.

Tendo em vista que os operadores de mutação utilizados na literatura foram desenvolvidos para otimização de parâmetros reais, foi necessário adaptá-los para o contexto de otimização intervalar. Seja um intervalo X definido por seu comprimento $\omega(X) = \overline{x} - \underline{x}$ e seu ponto médio $\mu(X) = (\underline{x} + \overline{x})/2$, tal que $X = [\mu(X) - \omega(X)/2,\ \mu(X) + \omega(X)/2]$, operadores de mutação podem ser aplicados sobre pontos médios, assim como nas representações de valores reais. No entanto, esses operadores devem ser estendidos para o comprimento dos intervalos. As Equações 4.1, 4.2 e 4.3 apresentam as versões intervalares dos operadores utilizados no InDE.

A versão intervalar do operador DE/rand/1 (Equação 2.4) combina os indivíduos aleatórios $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ para gerar um indivíduo mutante $\mathtt{v}_i$. O j-ésimo elemento de $\mathtt{v}_i$, denotado por $\mathtt{v}_{ij}$, é definido por:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{r}_{1j}) + F_i \cdot \left(\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j})\right), \tag{4.1}$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{r}_{1j}) + F_i \cdot \left(\omega(\mathtt{r}_{1j}) \cdot \frac{\omega(\mathtt{r}_{2j})}{\omega(\mathtt{r}_{3j})} - \omega(\mathtt{r}_{1j})\right).$$

Os outros dois operadores de mutação são definidos da mesma maneira. A versão intervalar de DE/current-to-*p*best/1 (Equação 3.1) combina o indivíduo $\mathtt{x}_i$ da população atual com dois indivíduos aleatoriamente selecionados, $\mathtt{r}_1$ e $\mathtt{r}_2$, e com $\mathtt{r}_{p\text{best}}$, um dos p melhores indivíduos da população. O indivíduo mutante resultante é definido por:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + F_i \cdot \left(\mu(\mathtt{r}_{\text{pbest}j}) - \mu(\mathtt{x}_{ij})\right) + F_i \cdot \left(\mu(\mathtt{r}_{1j}) - \mu(\mathtt{r}_{2j})\right), \tag{4.2}$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{x}_{ij}) + F_i \cdot \left(\omega(\mathtt{r}_{\text{pbest}j}) - \omega(\mathtt{x}_{ij})\right) + F_i \cdot \left(\omega(\mathtt{x}_{ij}) \cdot \frac{\omega(\mathtt{r}_{1j})}{\omega(\mathtt{r}_{2j})} - \omega(\mathtt{x}_{ij})\right).$$

A versão intervalar de DE/current-to-rand/1 (Equação 3.5) combina o indivíduo $\mathtt{x}_i$ da população atual com três indivíduos aleatoriamente escolhidos $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$. O indivíduo mutante resultante é definido por:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + s_i \cdot \left(\mu(\mathtt{r}_{1j}) - \mu(\mathtt{x}_{ij})\right) + F_i \cdot \left(\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j})\right), \tag{4.3}$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{x}_{ij}) + s_i \cdot \left(\omega(\mathtt{r}_{1j}) - \omega(\mathtt{x}_{ij})\right) + F_i \cdot \left(\omega(\mathtt{x}_{ij}) \cdot \frac{\omega(\mathtt{r}_{2j})}{\omega(\mathtt{r}_{3j})} - \omega(\mathtt{x}_{ij})\right),$$

sendo s_i um número aleatório entre 0 e 1. Esse operador de mutação não é associado a um operador de cruzamento, ou seja, o indivíduo experimental $\mathtt{u}_i$ é uma cópia do indivíduo mutante $\mathtt{v}_i$.

O Algoritmo 1 descreve a estratégia DE/rand/1/bin (Equação 4.1) utilizada no InDE. As demais estratégias possuem implementações semelhantes. Inicialmente, são selecionados três indivíduos aleatórios mutuamente distintos da população atual $\mathtt{p}_g$. Vale observar que os indivíduos base e terminal, correspondentes a $\mathtt{r}_1$ e $\mathtt{r}_2$, utilizam seleção baseada em *ranking* sobre a metade superior da população, como pode ser observado nas linhas 3 e 6 do Algoritmo 2. Nas linhas 4 e 5 do Algoritmo 1 é realizada uma cópia do indivíduo $\mathtt{x}_i$ da população atual e escolhido um intervalo aleatório a partir do qual as operações serão realizadas.

O ciclo de operações sobre os intervalos ocorre entre as linhas 7 e 34 do Algoritmo 1. Para cada um dos intervalos, é analisado se ele representa uma das variáveis V_Ω do conjunto Ω e se o operador de cruzamento se aplica a ele. Caso algum desses casos seja negativo, o ciclo de execução ignora esse intervalo e considera o próximo. A seguir, é calculado o comprimento $\omega(\mathtt{u}_{ij})$ e o ponto médio $\mu(\mathtt{u}_{ij})$ do intervalo que está sendo operado, conforme ilustrado nos Algoritmos 3 e 4. Caso o comprimento $\omega(\mathtt{u}_{ij})$ seja maior que o comprimento do indivíduo base $\omega(\mathtt{r}_{1j})$, esse valor é corrigido a fim de impedir que o intervalo aumente.

Nas linhas 16 e 17 do Algoritmo 1, são definidas as bordas do intervalo a partir de seu ponto médio e comprimento. No início da execução do InDE é definido um indivíduo original composto pelas atribuições de valores iniciais dos intervalos após a primeira aplicação do contrator GAC sobre a instância recém decomposta, que representa a caixa que contém todas as atribuições factíveis possíveis para as variáveis da instância. Dessa forma, entre as linhas 18 e 29 verifica-se se o intervalo gerado encontra-se dentro dos limites do intervalo do indivíduo original. Em caso negativo, o intervalo operado é deslocado para dentro dos limites do intervalo original. Por fim, na linha 33 atualiza-se o índice j do intervalo e as operações são aplicadas ao restante do indivíduo.

Seguindo a sequência de execução do InDE, o indivíduo experimental $\mathtt{u}_i$ passa pelo contrator GAC que poda valores dos intervalos que certamente não constituem a solução ótima da instância. Após a contração, os valores de função objetivo e violação de restrições são calculados para $\mathtt{u}_i$. Vale observar que, com a aplicação do contrator GAC, os intervalos que não foram

Algoritmo 1 Estratégia DE/rand/1/bin

Entrada: Indivíduo a ser operado $\mathtt{x}_i$, intervalo consistente original $\mathtt{y}_i$, população atual $\mathtt{p}_g$, tamanho da população NP, F, CR, número de dimensões D, conjunto de variáveis V_Ω
Saída: Indivíduo experimental $\mathtt{u}_i$

// Seleciona membros aleatórios da população
$(\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3) \leftarrow$ `seleciona_individuos`$(i, \mathtt{p}_g, NP)$
// Copia o indivíduo que será operado e escolhe uma posição aleatória para começar a operação
$\mathtt{u}_i \leftarrow \mathtt{x}_i$
$j \leftarrow rand[1 \ldots D]$
// Percorre cada variável do indivíduo
para $L \leftarrow 1$ **até** D **faça**
    // Se a variável pertence ao conjunto V_Ω
    se $\mathtt{u}_{ij} \in V_\Omega$ **então**
        // Se o operador de cruzamento deve ser aplicado ao intervalo $\mathtt{u}_{ij}$
        se $rand[0 \ldots 1] < CR$ **ou** $L = D$ **então**
            // Calcula comprimento e ponto médio do intervalo
            $\omega(\mathtt{u}_{ij}) \leftarrow$ `comprimento`$(\mathtt{u}_i, \mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3, F, j)$
            $\mu(\mathtt{u}_{ij}) \leftarrow$ `pontomedio`$(\mathtt{u}_i, \mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3, F, j)$
            // Define as bordas do intervalo
            $\underline{\mathtt{u}}_{ij} \leftarrow \mu(\mathtt{u}_{ij}) - \omega(\mathtt{u}_{ij})/2$
            $\overline{\mathtt{u}}_{ij} \leftarrow \mu(\mathtt{u}_{ij}) + \omega(\mathtt{u}_{ij})/2$
            // Garante que o novo intervalo está dentro dos limites do intervalo consistente original
            se $\omega(\mathtt{u}_{ij}) > \omega(\mathtt{y}_{ij})$ **então**
                $\omega(\mathtt{u}_{ij}) \leftarrow \omega(\mathtt{y}_{ij})$
            fim se
            se $\underline{\mathtt{u}}_{ij} < \underline{\mathtt{y}}_{ij}$ **então**
                $\underline{\mathtt{u}}_{ij} \leftarrow \underline{\mathtt{y}}_{ij}$
                $\overline{\mathtt{u}}_{ij} \leftarrow \underline{\mathtt{u}}_{ij} + \omega(\mathtt{u}_{ij})$
            fim se
            se $\overline{\mathtt{u}}_{ij} > \overline{\mathtt{y}}_{ij}$ **então**
                $\underline{\mathtt{u}}_{ij} \leftarrow \overline{\mathtt{u}}_{ij} - \omega(\mathtt{u}_{ij})$
                $\overline{\mathtt{u}}_{ij} \leftarrow \overline{\mathtt{y}}_{ij}$
            fim se
        fim se
    fim se
    // Passa para o próximo intervalo
    $j \leftarrow (j + 1)\ mod\ D$
fim para

operados têm seus valores atribuídos por meio de propagação pelo hipergrafo de restrições a partir do conjunto Ω.

Assim como no IUDE, na subpopulação A os três indivíduos experimentais gerados para cada indivíduo da população atual são comparados entre si e a estratégia que gerou o melhor deles contabiliza uma vitória. A cada geração, a taxa de sucesso de cada estratégia é avaliada sobre o período das L gerações anteriores, sendo L o período de aprendizagem. Por exemplo, a taxa de sucesso SR da estratégia 1 no período de aprendizagem $L = 25$ é dada por $SR_1 = NW_1/(NW_1 + NW_2 + NW_3)$, em que NW_i é o número de vitórias da estratégia i nas últimas 25 gerações. Na subpopulação B, a probabilidade de empregar uma estratégia é igual à sua taxa de sucesso recente na subpopulação A nas últimas L gerações.

Algoritmo 2 Seleciona indivíduos aleatórios: `seleciona_individuos`$(i, \mathtt{p}_g, NP)$

Entrada: Índice do indivíduo de referência i, população atual $\mathtt{p}_g$, tamanho da população atual NP
Saída: Indivíduos aleatórios $\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3$.

// Sorteia índices mutuamente distintos. i_1 e i_2 utilizam seleção baseada em *ranking*
repete
    $i_1 \leftarrow rand[1 \ldots NP/2]$
até que $i_1 \neq i$
repete
    $i_2 \leftarrow rand[1 \ldots NP/2]$
até que $i_2 \neq i$ **e** $i_2 \neq i_1$
repete
    $i_3 \leftarrow rand[1 \ldots NP]$
até que $i_3 \neq i$ **e** $i_3 \neq i_1$ **e** $i_3 \neq i_2$
// Define os indivíduos aleatórios
$\mathtt{r}_1 \leftarrow \mathtt{p}_{gi_1}$
$\mathtt{r}_2 \leftarrow \mathtt{p}_{gi_2}$
$\mathtt{r}_3 \leftarrow \mathtt{p}_{gi_3}$

Algoritmo 3 Calcula o comprimento do intervalo: `comprimento`$(\mathtt{u}_i, \mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3, F, j)$

Entrada: Indivíduo mutante $\mathtt{u}_i$, indivíduos aleatórios $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$, fator escalar de mutação F, intervalo a ser alterado j
Saída: Comprimento do intervalo $\omega(\mathtt{u}_{ij})$

// Calcula o comprimento do intervalo selecionado de cada indivíduo
$\omega(\mathtt{r}_{1j}) \leftarrow \overline{r}_{1j} - \underline{r}_{1j}$
$\omega(\mathtt{r}_{2j}) \leftarrow \overline{r}_{2j} - \underline{r}_{2j}$
$\omega(\mathtt{r}_{3j}) \leftarrow \overline{r}_{3j} - \underline{r}_{3j}$
$\omega(\mathtt{u}_{ij}) \leftarrow \omega(\mathtt{r}_{1j}) + F \cdot (\omega(\mathtt{r}_{1j}) \cdot (\omega(\mathtt{r}_{2j})/\omega(\mathtt{r}_{3j})) - \omega(\mathtt{r}_{1j}))$
// Corrige comprimento se for maior que o comprimento de $\omega(\mathtt{r}_{1j})$
se $\omega(\mathtt{u}_{ij}) > \omega(\mathtt{r}_{1j})$ **então**
    $\omega(\mathtt{u}_{ij}) \leftarrow \omega(\mathtt{r}_{1j})$
fim se

Algoritmo 4 Calcula ponto médio do intervalo: `pontomedio`$(\mathtt{u}_i, \mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3, F, j)$

Entrada: Indivíduo mutante $\mathtt{u}_i$, indivíduos aleatórios $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$, fator escalar de mutação F, intervalo a ser alterado j
Saída: Ponto médio do intervalo $\mu(\mathtt{u}_{ij})$

// Calcula o ponto médio do intervalo selecionado de cada indivíduo
$\mu(\mathtt{r}_{1j}) \leftarrow (\underline{r}_{1j} + \overline{r}_{1j})/2$
$\mu(\mathtt{r}_{2j}) \leftarrow (\underline{r}_{2j} + \overline{r}_{2j})/2$
$\mu(\mathtt{r}_{3j}) \leftarrow (\underline{r}_{3j} + \overline{r}_{3j})/2$
$\mu(\mathtt{u}_{ij}) \leftarrow \mu(\mathtt{r}_{1j}) + F \cdot (\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j}))$

Conforme discutido no Capítulo 3, a configuração de valores para os parâmetros F e CR é dependente da instância e pode mudar de acordo com a região do espaço de busca que está sendo percorrida. Por esse motivo, foi utilizado o esquema de adaptação de parâmetros aplicado no LSHADE44 (Seção 3.2). Assim, é empregado um par de memórias MF e MCR para as estratégias DE/rand/1 e DE/current-to-$p$best/1 e MF para DE/current-to-rand/1. É importante notar que somente indivíduos experimentais que obtiveram sucesso na subpopulação A são

utilizados na adaptação de parâmetros, visto que somente nessa subpopulação os três operadores de mutação são utilizados para cada indivíduo.

Dessa forma, ao fim de cada geração os valores médios das configurações *F* e *CR* que obtiveram sucesso são calculados e armazenados nas memórias. Então, os parâmetros utilizados por cada indivíduo a cada geração são gerados de acordo com uma distribuição uniforme a partir dessas memórias.

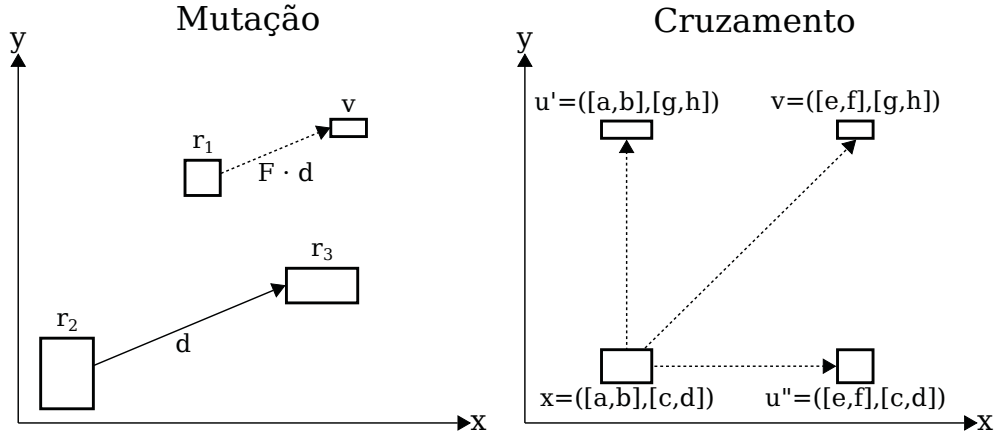


Figura 4.2: Exemplo de operações intervalares de mutação e cruzamento da estratégia DE/rand/1/bin em duas dimensões.

A Figura 4.2 ilustra um exemplo das operações de mutação e cruzamento intervalares da estratégia DE/rand/1/bin (Equação 4.1) em duas dimensões. Nesse caso, os indivíduos são representados por caixas que se deslocam pelo espaço de busca por meio das operações. Na mutação, $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ são indivíduos mutuamente distintos aleatoriamente selecionados na população. *d* é o vetor diferença entre os indivíduos $\mathtt{r}_2$ e $\mathtt{r}_3$. Para criação do indivíduo mutante $\mathtt{v}_i$, *d* é multiplicado pelo fator escalar *F* e adicionado a $\mathtt{r}_1$.

Na operação de cruzamento é criado o indivíduo experimental $\mathtt{u}_i$ por meio da escolha aleatória de intervalos correspondentes do indivíduo mutante $\mathtt{v}_i$ e do indivíduo alvo $\mathtt{x}_i$. A probabilidade de escolha de um intervalo do indivíduo mutante é dada por *CR*. De forma análoga ao DE clássico, um intervalo do indivíduo mutante é aleatoriamente selecionado a fim de garantir que o indivíduo experimental não será igual ao indivíduo alvo. No caso da Figura 4.2, após a operação de cruzamento, o indivíduo experimental pode ser $\mathtt{u}'_i$, $\mathtt{u}''_i$ ou $\mathtt{v}_i$.

Na próxima seção é descrito o processo de avaliação de *fitness* dos indivíduos pertencentes à população intervalar do InDE.

4.4 AVALIAÇÃO DE *FITNESS*

O método ε*-constrained* (Seção 2.2.2) foi utilizado na operação de seleção do InDE de forma similar ao IUDE e LSHADE44. As regras de comparação entre indivíduos e o declínio de ε são definidos pelas Equações 2.6 e 2.7.

A fim de computar o valor de *fitness* de um indivíduo $\mathtt{x}$, é utilizado o mecanismo de consistência local do OGRe. Primeiramente, o contrator GAC reduz os intervalos para remover valores inconsistentes. Em seguida, tenta-se encontrar uma atribuição de valores para a rede de restrições, começando pela atribuição inicial $\langle z = \underline{Z} \rangle$, a borda inferior do intervalo associado à função objetivo.

Ao contrário do OGRe, o InDE permite que as restrições do conjunto Ω tenham seus valores atribuídos independentemente da tolerância necessária para satisfazê-las. A soma de

todas essas tolerâncias forma o valor de violação de restrições $\phi(\mathrm{x})$ do indivíduo. Se o contrator implica em algum intervalo vazio, a violação é definida como $\phi(\mathrm{x}) = \infty$. O valor da função objetivo $f(\mathrm{x})$ é simplesmente definido por $\underline{Z}$.

O indivíduo consistente obtido a partir do contrator GAC pode conter multi-intervalos. Neste caso, a caixa de multi-intervalos é separada em um conjunto de indivíduos intervalares, adicionando à população aquele que possui a melhor avaliação pelo método ε-*constrained* (se for melhor que o indivíduo correspondente na população atual) e alocando na população adicional todos os outros indivíduos intervalares gerados. Vale notar que, ao contrário dos métodos usuais de DE, um novo indivíduo não é definido apenas pelas operações de mutação e cruzamento, mas também pelo contrator GAC que reduz seus intervalos.

4.5 CONSIDERAÇÕES

Neste capítulo foram apresentadas as principais características do método proposto neste trabalho. O objetivo do InDE é a integração entre tecnologias estado-da-arte em otimização de parâmetros reais por meta-heurísticas e o recurso de decomposição Epífita. A utilização dessa decomposição permite que o DE execute suas operações somente em um subconjunto de variáveis da representação ternária, enquanto as demais são atribuídas por propagação pelo hipergrafo de restrições da instância. A cada avaliação de *fitness*, a técnica de consistência local dada pelo contrator GAC poda valores infactíveis dos intervalos, o que pode acelerar a convergência do processo de busca.

Dessa forma, procurou-se demonstrar de que forma os conceitos e algoritmos comentados na revisão bibliográfica do Capítulo 3 foram utilizados para possibilitar a avaliação da proposta deste estudo. Foi descrito o processo de execução do algoritmo e detalhadas cada uma das etapas, salientando os pontos em que são utilizados os recursos de decomposição estrutural e consistência local. Adicionalmente, foram caracterizadas as adaptações realizadas nas operações tradicionais de DE para o contexto intervalar.

No próximo capítulo é apresentada a avaliação experimental deste estudo. São analisadas características da decomposição estrutural das instâncias e seu impacto sobre o processo de otimização. É realizada, também, uma análise comparativa entre o InDE, uma variante de DE para otimização caixa-preta de parâmetros reais e o OGRe a fim de avaliar o impacto da junção das estratégias elencadas neste capítulo.

5 AVALIAÇÃO EXPERIMENTAL

O algoritmo InDE tem por objetivo agregar informações estruturais de instâncias NCOP e um procedimento de consistência local ao DE. Dessa forma, o InDE executa suas operações apenas sobre as variáveis V_Ω pertencentes às restrições de uma decomposição Epífita da instância ternária e a atribuição de valores das demais é inferida automaticamente por meio de propagação pelo hipergrafo de restrições. Vale salientar que os indivíduos são constituídos por uma atribuição de intervalos para as variáveis da forma ternária da instância NCOP original.

Neste capítulo é descrita a avaliação experimental realizada sobre o algoritmo InDE, apresentado no Capítulo 4. Assim como o OGRe, o InDE foi implementado na linguagem de programação C e compilado com *gcc* versão 5.4.0. Toda a avaliação descrita neste capítulo foi executada em um computador com 4 *sockets* Intel Xeon E5-4627 v2 @ 3.30GHz com 8 núcleos por *socket*, 256GB de memória RAM e sistema operacional CentOS Linux 7, pertencente ao Centro de Computação Científica e Software Livre (C3SL)[1].

Para a avaliação experimental deste trabalho foram selecionadas 80 instâncias do *benchmark* COCONUT (Shcherbina et al., 2003), comumente utilizado no contexto de otimização global e satisfação de restrições. Essa escolha se deve ao fato de que o otimizador OGRe foi avaliado sobre esse mesmo conjunto de instâncias, o que possibilita a análise comparativa dos resultados obtidos em termos de qualidade da solução encontrada e tempo de processamento. Além disso, os módulos de decomposição estrutural e consistência local do OGRe que foram empregados no InDE utilizam a representação das instâncias no formato AMPL (*A Mathematical Programming Language*), fornecido pelo COCONUT. Assim como no OGRe, as instâncias escolhidas para os experimentos limitaram-se àquelas com as seguintes características:

- instâncias de problemas de otimização sem a ocorrência de variáveis discretas;
- instâncias que utilizam apenas os operadores $+, -, \cdot, /, \hat{}$ e $\sqrt{}$, sendo estes últimos limitados, respectivamente, a expoentes e índices inteiros.

Além disso, a escolha das instâncias restringiu-se àquelas que foram convertidas com sucesso do formato AMPL para a linguagem C, conforme será comentado com mais detalhes na Seção 5.2. Vale observar que a implementação do InDE não se limita à abordagem de instâncias do *benchmark* COCONUT, podendo atacar funções com diferentes características, desde que sejam modeladas no formato AMPL e respeitem as restrições operacionais citadas acima.

Este capítulo se divide em duas seções:

5.1. **Análise da Decomposição Estrutural**: Traz informações relacionadas às características das instâncias em sua modelagem original, representadas na forma de rede ternária de restrições e como conjunto Ω de uma decomposição Epífita;

5.2. **Análise Comparativa de Desempenho**: Relata os critérios para a seleção dos parâmetros de execução dos algoritmos e analisa a qualidade das soluções obtidas e o tempo de execução consumidos pelos algoritmos InDE, DE para otimização caixa-preta de parâmetros reais e OGRe.

[1] https://www.c3sl.ufpr.br/, acessado em 24/06/2019.

5.1 ANÁLISE DA DECOMPOSIÇÃO ESTRUTURAL

O primeiro aspecto analisado refere-se ao impacto da decomposição estrutural da instância NCOP. Conforme comentado na Seção 2.1, inicialmente o OGRe codifica a instância original em um hipergrafo de restrições ternárias. Como essa transformação normalmente implica na adição de variáveis auxiliares à representação original, tem-se um crescimento no número total de variáveis e restrições da modelagem da instância.

A partir do hipergrafo de restrições ternárias, o OGRe extrai as restrições pertencentes ao conjunto Ω da decomposição Epífita, que é representado por um subconjunto de variáveis V_Ω da instância que possuem atribuição de valores crítica. Conforme comentado no Capítulo 4, o InDE realiza seu ciclo de operações somente sobre essas variáveis, sendo as demais valorações atribuídas por meio de propagação pelo hipergrafo de restrições da instância. Um exemplo disso é a função de Rosenbrock, dada na Figura 2.2.

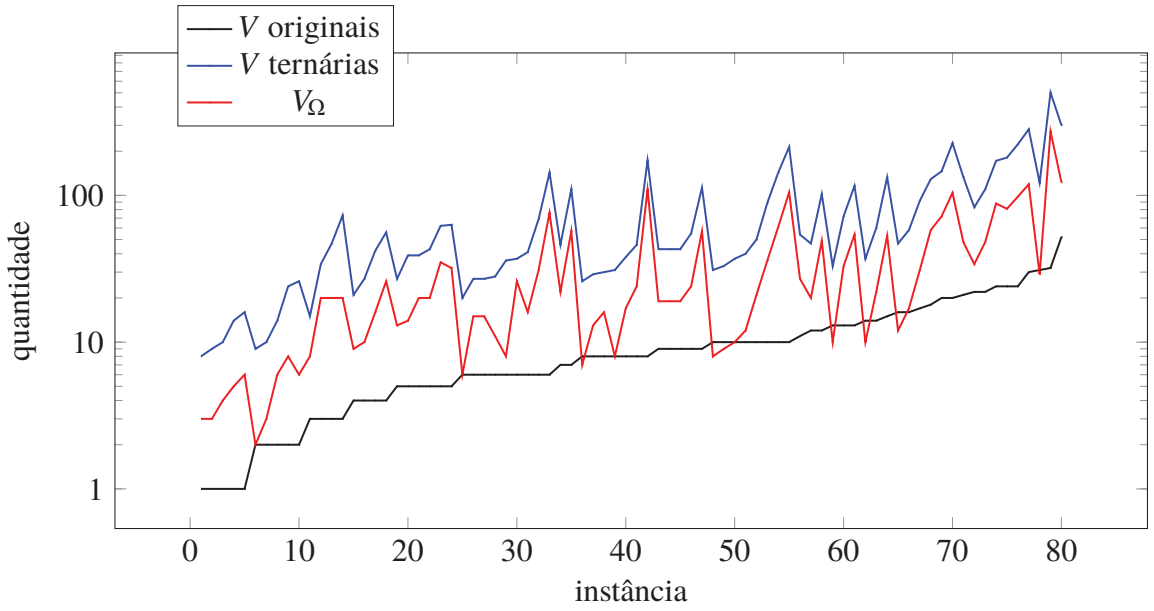


Figura 5.1: Quantidade de variáveis originais, ternárias e V_Ω por instância.

As Tabelas 5.1 e 5.2 apresentam a quantidade de variáveis e restrições das três diferentes representações das instâncias selecionadas do *benchmark* COCONUT. A Figura 5.1 ilustra esses mesmos dados ordenados pela quantidade de variáveis de representação original. Vale notar que a quantidade de variáveis e restrições pertencentes ao conjunto Ω varia de acordo com a decomposição que está sendo considerada, visto que uma instância pode possuir mais de uma decomposição Epífita. Os dados apresentados nas tabelas e na figura referem-se à decomposição Epífita utilizada em Derenievicz (2018b).

As variáveis auxiliares adicionadas para a representação ternária ocasionam um aumento no número total de variáveis e restrições da instância. Quanto ao conjunto Ω, a redução do número de variáveis e restrições em relação à rede ternária é evidente. No entanto, quando comparamos a representação original da instância ao conjunto Ω, observa-se que na maior parte dos casos a quantidade de variáveis e restrições da representação original é consideravelmente menor. Algumas das instâncias que demonstraram comportamento distinto foram ex9_1_4, ex9_1_5, ex9_1_8, ex9_2_2, ex9_2_4, ex9_2_5 e hydro.

Tabela 5.1: Relação entre quantidade de variáveis e restrições da modelagem original, da rede de restrições ternárias e do conjunto Ω das instâncias selecionadas - parte 1.

	original		rede ternária		conjunto Ω	
instância	variáveis	restrições	variáveis	restrições	variáveis	restrições
abel	30	14	282	267	119	89
alkyl	14	7	60	54	22	15
chance	4	3	42	42	16	13
circle	3	10	73	81	20	18
dispatch	4	2	56	55	26	23
ex14_1_1	3	4	47	49	20	18
ex14_1_2	6	9	143	147	76	72
ex14_1_5	6	6	37	38	26	21
ex14_1_6	9	15	113	120	57	49
ex2_1_1	5	1	39	36	14	9
ex2_1_2	6	2	41	38	16	11
ex2_1_3	13	9	72	69	33	22
ex2_1_4	6	5	69	69	31	26
ex2_1_5	10	11	214	216	105	96
ex2_1_6	10	5	143	139	61	51
ex2_1_8	24	10	181	168	81	62
ex3_1_1	8	6	38	37	17	11
ex3_1_2	5	6	63	65	32	27
ex3_1_3	6	6	36	37	8	6
ex3_1_4	3	3	34	35	20	17
ex4_1_1	1	0	16	16	6	5
ex4_1_3	1	0	14	14	5	4
ex4_1_4	1	0	8	8	3	2
ex4_1_5	2	0	14	13	6	4
ex4_1_6	1	0	9	9	3	2
ex4_1_7	1	0	10	10	4	3
ex4_1_8	2	1	10	10	3	2
ex4_1_9	2	2	24	25	8	7
ex5_2_2_case1	9	6	43	41	19	15
ex5_2_2_case2	9	6	43	41	19	15
ex5_2_2_case3	9	6	43	41	19	15
ex5_2_4	7	6	46	46	22	18
ex5_2_5	32	19	502	490	273	247
ex5_3_2	22	16	83	78	34	25
ex5_4_2	8	6	30	29	16	11
ex7_2_10	11	9	54	53	27	18
ex7_2_1	7	14	110	118	57	50
ex7_2_2	6	5	27	27	15	9
ex7_2_3	8	6	46	45	24	17
ex7_2_5	5	6	62	64	35	29

Tabela 5.2: Relação entre quantidade de variáveis e restrições da modelagem original, da rede de restrições ternárias e do conjunto Ω das instâncias selecionadas - parte 2.

	original		rede ternária		conjunto Ω	
instância	variáveis	restrições	variáveis	restrições	variáveis	restrições
ex7_2_6	3	1	15	14	8	5
ex7_3_2	4	7	27	31	10	7
ex7_3_3	5	8	43	47	20	15
ex7_3_4	12	17	102	108	49	40
ex7_3_5	13	15	116	119	54	43
ex8_1_6	2	0	26	25	6	4
ex8_1_7	5	5	39	40	20	15
ex8_1_8	6	5	27	27	15	9
ex8_4_1	22	10	110	99	48	28
ex8_4_2	24	10	172	159	88	66
ex8_4_3	52	25	301	275	123	73
ex8_4_5	15	11	133	130	53	40
ex9_1_4	10	9	33	33	9	6
ex9_1_5	13	12	33	33	10	7
ex9_1_8	14	12	37	34	10	7
ex9_2_1	10	9	40	40	12	10
ex9_2_2	10	11	31	31	8	7
ex9_2_3	16	15	58	58	17	13
ex9_2_4	8	7	26	26	7	5
ex9_2_5	8	7	31	31	8	6
ex9_2_6	16	12	47	40	12	8
ex9_2_7	10	9	37	37	10	8
ex9_2_8	6	5	20	20	6	4
harker	20	7	146	134	72	53
haverly	12	9	47	45	20	14
himmel11	9	4	55	51	24	18
himmel16	18	21	129	133	58	57
house	8	8	29	30	13	9
hydro	31	24	122	116	29	17
immun	21	7	133	120	48	35
linear	24	20	223	220	98	76
meanvar	8	2	173	168	110	103
mhw4d	5	3	27	26	13	9
process	10	7	50	48	21	13
prolog	20	22	227	230	104	88
rbrock	2	0	9	8	2	1
sambal	17	10	92	86	31	20
sample	4	2	21	20	9	6
ship	10	16	89	96	36	29
wall	6	6	28	29	11	9

É importante notar que o crescimento do número de variáveis e restrições é linear em relação ao tamanho original da instância, sendo esse tamanho definido pela quantidade de símbolos (variáveis, operadores, restrições, etc.) da instância. Isso significa que o crescimento

da quantidade de variáveis e restrições está diretamente relacionado à modelagem da instância original, pois quanto mais operadores houverem na função objetivo e nas restrições originais, maior será a rede de restrições ternárias.

Na próxima seção serão apresentados os resultados relacionados ao desempenho dos algoritmos de otimização participantes desta avaliação experimental.

5.2 ANÁLISE COMPARATIVA DE DESEMPENHO

A avaliação de desempenho deste trabalho é baseada em uma análise comparativa entre o algoritmo InDE proposto, o otimizador OGRe e uma versão de DE que utiliza as mesmas tecnologias de meta-heurísticas do InDE para otimização de parâmetros reais. A partir disso, espera-se avaliar o impacto da utilização de decomposição Epífita aplicada ao contexto de otimização restrita por meta-heurísticas.

O algoritmo DE utilizado nesta análise comparativa emprega as mesmas estratégias e tecnologias de meta-heurísticas do InDE, mas no contexto de otimização caixa-preta, em que as informações sobre a estrutura da instância são desconhecidas. Para integração entre o DE caixa-preta e o *benchmark* COCONUT, foi utilizado um conversor[2] para transformação das instâncias AMPL em funções na linguagem C. Essas funções fornecem os limites originais das variáveis da instância e são responsáveis por calcular o valor da função objetivo e da violação de restrições.

Tendo em vista que no DE as variáveis não são representadas por intervalos, não é possível utilizar o recurso de *Branch & Bound* do OGRe para a criação da população inicial. Dessa forma, a população inicial do DE é gerada por indivíduos aleatoriamente distribuídos no espaço de busca da instância. Além disso, as operações são executadas sobre variáveis de parâmetros reais da mesma forma que no DE clássico. É importante notar que essa variante de DE não utiliza decomposição estrutural, então todas as variáveis da instância, em sua codificação original, participam do ciclo de operações do algoritmo.

Os esquemas de subpopulações A e B, a adaptação dos parâmetros F e CR, o método de tratamento de restrições ε-*constrained*, a competição entre estratégias, a população adicional e o mecanismo de redução linear do tamanho da população são adaptações da implementação intervalar do InDE.

Conforme comentado no Capítulo 4, no InDE buscou-se integrar estratégias adaptativas para reduzir a quantidade de parâmetros a serem ajustados a fim de tornar a execução do algoritmo mais simples e robusta. No entanto, nem todos os parâmetros puderam ser removidos e alguns foram adicionados em função da adoção de estratégias adaptativas. As configurações utilizadas no InDE foram definidas a partir de avaliações preliminares do algoritmo e são listadas a seguir:

- o número máximo de avaliações de *fitness*: $MaxFEs = 20000 \times |V_\Omega|$;
- os tamanhos inicial e final da população foram definidos por $NP^{max} = 12 \times |V_\Omega|$ e $NP^{min} = 6$, respectivamente. A população adicional e a população atual $\mathtt{p}_g$ são unidas e ordenadas a cada 50 gerações;
- o operador de mutação DE/current-to-$p$best/1 utilizou $p = 20\%$ da população;
- o tamanho do histórico de memória utilizado para adaptação dos parâmetros F e CR foi $H = 5$ e o período de aprendizagem utilizado na seleção adaptativa de estratégias de mutação foi definido como $L = 25$ gerações;

[2]Disponível em `https://ampl.com/`, acessado em 09/05/2019.

- os parâmetros empregados no método de tratamento de restrições ε-*constrained* foram $\theta = 0,5$ e $T = 0,75 \times G$, sendo G o número máximo de gerações;
- o número máximo de aplicações do contrator GAC foi de 1000 iterações, o mesmo valor utilizado no OGRe.

Na variante de DE buscou-se manter a mesma configuração de parâmetros do InDE. As únicas diferenças consistem no número máximo de avaliações de *fitness* e no tamanho inicial da população, que utilizam o número total de variáveis da instância no lugar do número de variáveis V_Ω.

A partir disso, tornou-se necessária a definição do tipo de cruzamento utilizado nas operações. Conforme citado no Capítulo 4, o InDE utiliza um conjunto de três estratégias, mas somente duas delas utilizam o operador de cruzamento: DE/rand/1 e DE/current-to-*p*best/1. Tendo em vista que os principais operadores de cruzamento presentes na literatura são binomial e exponencial, com as duas estratégias tem-se quatro combinações possíveis de cruzamento.

Para avaliação do melhor tipo de cruzamento tanto para o InDE quanto para o DE, realizamos um experimento com 10 execuções das quatro diferentes configurações de cruzamento sobre as 80 instâncias selecionadas do *bechmark* COCONUT. Na Tabela 5.3, são apresentados os resultados obtidos para as estratégias de mutação DE/rand/1 (a) e DE/current-to-*p*best/1 (b), sendo "bin" e "exp" os tipos de cruzamento binomial e exponencial, respectivamente.

Tabela 5.3: Análise do InDE e do DE em relação à factibilidade das atribuições de valores encontradas.

	InDE			DE		
cruzamento a x b	instâncias factíveis	execuções factíveis	total de vitórias	instâncias factíveis	execuções factíveis	total de vitórias
bin x bin	62	476	26	35	328	28
bin x exp	61	489	28	37	330	26
exp x bin	60	461	31	36	330	28
exp x exp	62	471	31	36	333	28

Inicialmente consideramos a quantidade de instâncias para as quais foram encontradas soluções factíveis, ou seja, atribuições de valores que não violam nenhuma restrição da instância. No InDE, as configurações "bin x bin" e "exp x exp" demonstraram desempenho superior, obtendo 62 instâncias factíveis cada. No caso do DE, a configuração "bin x exp" se sobressaiu com um total de 37 instâncias factíveis. No entanto, observou-se que as diferentes combinações de estratégias de cruzamento não apresentaram uma diferença muito significativa entre si.

Na sequência avaliamos quantas das 10 execuções das 80 instâncias encontraram soluções factíveis, totalizando 800 execuções. No InDE, a configuração "bin x exp" se sobressaiu com 489 execuções factíveis. Para o DE, a discrepância entre os resultados foi menor e a configuração "exp x exp" mostrou-se superior às demais configurações.

A qualidade dos resultados obtidos foi avaliada com base na solução com menor erro absoluto dentre as 10 execuções de cada instância para cada configuração de parâmetros. O erro absoluto é a distância absoluta entre a solução obtida e o ótimo global da instância. Dessa forma, realizou-se uma competição entre as quatro configurações de cruzamento em que a configuração com menor erro absoluto para determinada instância contabiliza uma vitória. Em caso de empate, todas as melhores configurações pontuam. Para o InDE, "exp x bin" e "exp x exp" obtiveram 31 vitórias. No caso do DE, "bin x bin", "exp x bin" e "exp x exp" obtiveram 28 vitórias. A partir dessa análise, optou-se por utilizar o operador de cruzamento exponencial associado às estratégias de mutação DE/rand/1 e DE/current-to-*p*best/1 tanto no InDE quanto no DE.

Com relação ao OGRe, os resultados apresentados neste capítulo foram extraídos de Derenievicz (2018b). O autor executou uma série de experimentos com diferentes conjuntos de configurações de parâmetros e selecionou a melhor execução de cada instância. As configurações de parâmetros empregadas para a obtenção de cada uma das atribuições de valores são detalhadas no trabalho original.

A partir da seleção do tipo de cruzamento a ser utilizado nas operações do InDE e do DE, foi executado um experimento com 25 execuções de cada uma das instâncias selecionadas do *benchmark* COCONUT. Tendo em vista que os resultados disponibilizados do OGRe retratam a melhor execução do algoritmo dentre diversas configurações de parâmetros, tanto para o InDE quando para o DE caixa-preta, foi considerada a melhor das 25 execuções de cada instância.

Nesta avaliação experimental, tanto para o InDE quanto para o DE consideramos apenas as atribuições de valores que respeitam todas as restrições da instância, ou seja, somente as soluções factíveis. Como o OGRe utiliza uma estratégia de busca essencialmente exaustiva, admitimos atribuições de valores que possuam um valor máximo de tolerância $\varepsilon = 10^{-1}$, aqui denominadas ε-factíveis.

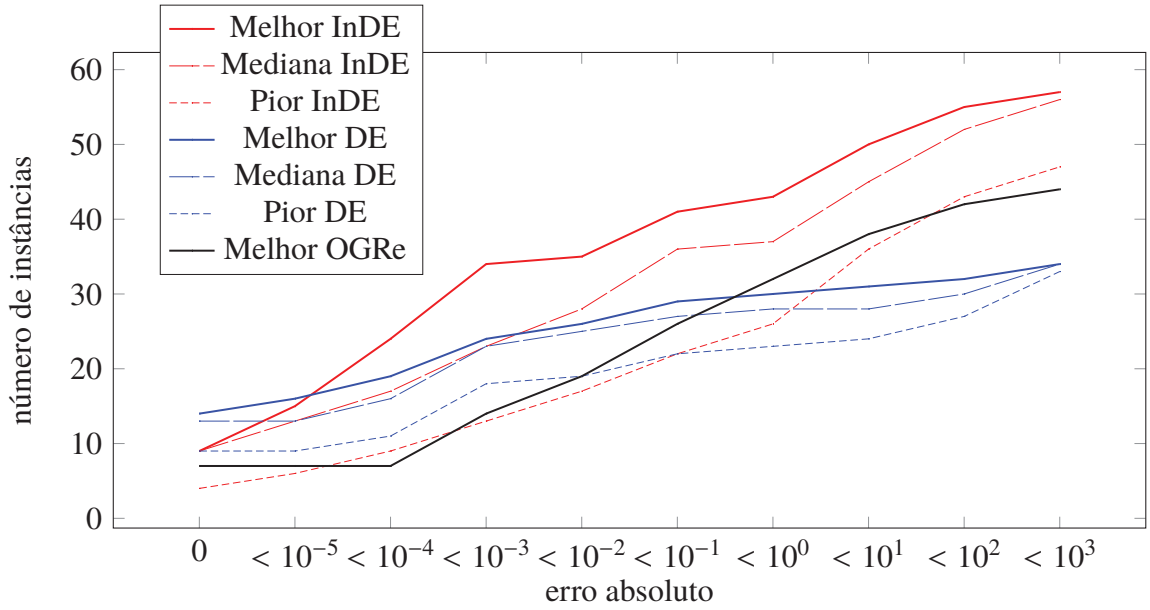


Figura 5.2: Número acumulado de instâncias factíveis dados alguns limites de erro absoluto.

A Figura 5.2 exibe o número acumulado de instâncias factíveis (e ε-factíveis, no caso do OGRe) resolvidas dados alguns limites de erro absoluto (distância da solução obtida ao ponto ótimo da instância). As linhas representam a melhor, a mediana e a pior das 25 execuções do InDE e do DE para otimização caixa-preta. Para o OGRe, é exibida somente a melhor atribuição de valores de cada instância dentre os conjuntos de parâmetros testados. Instâncias com erro absoluto maior que 10^3 foram desconsideradas.

O primeiro aspecto a ser analisado é a quantidade de soluções factíveis e ε-factíveis com erro absoluto igual a zero, ou seja, soluções que representam o ponto ótimo global da instância. Nesse caso, tem-se 9 instâncias (11,25%) nas melhores execuções do InDE contra 14 instâncias (17,5%) do DE e 7 instâncias (8,75%) do OGRe.

O DE caixa-preta encontrou um número maior de instâncias com erro absoluto menor que 10^{-5}. No entanto, essa vantagem se deteriora a partir de 10^{-5}. De modo geral, dado um erro absoluto máximo (*maxEA*) o InDE encontrou mais soluções com erro menor que *maxEA* se comparado ao DE e ao OGRe. Vale observar que os resultados apresentados em Derenievicz

(2018b) foram arredondados para facilitar a visualização, por esse motivo a quantidade de soluções do OGRe com erro absoluto entre 0 e 10^{-4} se manteve constante.

Considerando as melhores execuções de cada algoritmo, o InDE obteve 57 soluções factíveis com erro absoluto abaixo de 10^3, o que representa $71,25\%$ do total de instâncias. Se desconsiderarmos o limite de erro absoluto, esse número sobe para 62 soluções ($77,5\%$). No caso do DE, tem-se 34 soluções factíveis ($42,5\%$) que respeitam o limite de erro absoluto e 36 ou 45% totais. Por fim, o OGRe obteve 44 soluções (55%) ε-factíveis que respeitam o limiar de erro absoluto. Considerando todas as instâncias ε-factíveis, tem-se 46 soluções ou $57,5\%$ para o OGRe. É importante notar que o InDE obteve desempenho superior ao OGRe utilizando o mesmo procedimento de consistência local e a mesma decomposição Epífita.

Por fim, as Tabelas 5.4, 5.5 e 5.6 apresentam em detalhes os resultados da melhor execução do três algoritmos utilizados na análise comparativa de desempenho. Tanto para o InDE quanto para o DE caixa-preta são exibidas a atribuição de valores encontrada, o valor de violação de restrições se a atribuição for infactível, a porcentagem de atribuições factíveis encontradas nas 25 execuções e o tempo médio em segundos das execuções de cada instância. No caso do OGRe, é exibida a melhor atribuição encontrada, o valor de tolerância ε utilizado e o tempo de execução.

Vale salientar que a violação do InDE representa a soma das violações de restrições do conjunto Ω da representação ternária da instância, enquanto que a violação do DE representa a soma das violações da instância original. No caso do OGRe é exibida a tolerância ε máxima utilizada para satisfação de cada uma das restrições do conjunto Ω. Não apresentamos a soma das tolerâncias do OGRe pois elas representam o valor máximo considerado, mas não há garantia de que foi utilizado esse valor em cada restrição.

As instâncias destacadas em negrito indicam a atribuição com menor erro absoluto dentre os três algoritmos, sendo que atribuições factíveis e ε-factíveis têm prioridade sobre atribuições infactíveis. Em caso de empate, as atribuições envolvidas foram destacadas. Para os três algoritmos, TLE (Tempo Limite Excedido) indica que a execução excedeu o limite de 7200 segundos. É interessante notar que utilizando esses critérios, o InDE obteve 43 vitórias contra 31 do DE e 16 do OGRE.

Tendo em vista que o tempo de execução de uma instância não está diretamente relacionado à qualidade ou factibilidade da atribuição de valores, consideramos todas as 80 instâncias selecionadas do *benchmark* COCONUT para a análise de tempo. O algoritmo InDE obteve 13 ocorrências de TLE, contra 21 do OGRE. No DE nenhuma execução excedeu o tempo limite estabelecido. Considerando todas as ocorrências de TLE como 7200 segundos, a soma do tempo médio de todas as 80 instâncias foi de $151298,75$ segundos para o InDE, $15,34$ segundos para o DE e $180665,44$ segundos para o OGRe. Dessa forma, o tempo médio de execução das 80 instâncias foi de $1891,23$ segundos para o InDE, $0,19$ segundos para o DE e $2258,318$ segundos para o OGRe.

Essa discrepância de tempo de execução entre o DE caixa-preta e os demais algoritmos é decorrente da utilização de consistência local, da análise intervalar, da representação de multi-intervalos e do procedimento de atribuição de valores para as variáveis através da rede de restrições. Acredita-se que a reestruturação e otimização da implementação dos módulos responsáveis por essas tarefas possa reduzir significativamente o tempo de execução tanto do InDE quanto do OGRe.

	InDE				DE				OGRe			
instância	solução	violação	% fact.	tempo (s)	solução	violação	% fact.	tempo (s)	solução	ε	tempo (s)	ótimo
abel	**$-1 \cdot 10^{17}$**	**8924,4197**	**0**	**TLE**	**$1 \cdot 10^{17}$**	**$1 \cdot 10^{17}$**	**0**	**0,357**	$1,4 \cdot 10^{21}$	5	TLE	225,195
alkyl	**-1,7611**	**-**	**64**	**829,259**	-1,7178	-	100	0,252	-1,5232	10^{-2}	TLE	-1,765
chance	**28,4108**	**-**	**100**	**392,094**	0	27	0	0,073	28,4095	10^{-4}	2256	29,8944
circle	4,5742	-	44	962,886	0	$1 \cdot 10^{17}$	0	0,051	**4,5742**	**10^{-2}**	**2,196**	4,5742
dispatch	**3155,2898**	**-**	**64**	**899,663**	2152,3543	$1 \cdot 10^{17}$	0	0,064	3155,3	10^{-4}	TLE	3155,29
ex14_1_1	**0,0001**	**-**	**28**	**TLE**	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,101	13,096	10^{-1}	TLE	0
ex14_1_2	**0,0110**	**-**	**72**	**2746,1**	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,086	-0,0647	1	280,2	0
ex14_1_5	**0**	**-**	**100**	**948,307**	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,084	-0,1875	10^{-1}	580,7	0
ex14_1_6	**0,4612**	**-**	**100**	**TLE**	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,11	0,1577	1	0,021	0,461
ex2_1_1	-16,9997	-	52	280,221	**-17**	**-**	**100**	**0,065**	1,1703	10^{-2}	388,6	-17
ex2_1_2	-212,9998	$2,2 \cdot 10^{-5}$	0	332,754	**-213**	**-**	**100**	**0,088**	-212,688	5	3,125	-213
ex2_1_3	-14,9995	-	100	1409,387	**-15**	**-**	**100**	**0,229**	-3,9986	10^{-2}	TLE	-15
ex2_1_4	-10,9999	-	88	998,204	**-11**	**-**	**100**	**0,122**	-11,721	5	0,705	-11
ex2_1_5	-267,1670	-	44	TLE	**-268,0146**	**-**	**100**	**0,188**	-299,34	5	1,056	-268,015
ex2_1_6	**-24,6547**	**-**	**4**	**4140,723**	-49	8	0	0,138	-251,357	5	5,091	-39
ex2_1_8	45222,6542	0,0029	0	2582,374	**15639,0000**	**-**	**100**	**0,439**	25453,8	5	TLE	15639
ex3_1_1	**7055,5186**	**-**	**60**	**375,141**	2100	1,208	0	0,115	5218,12	10^{-1}	5445	7049,25
ex3_1_2	-30667,0696	-	100	1322,242	**-30665,5386**	**-**	**100**	**0,106**	-32217,4	5	0,001	-30665,5
ex3_1_3	-310,00009	2,7475	0	95,172	**-310**	**-**	**100**	**0,088**	-310,003	10^{-4}	3,343	-310
ex3_1_4	**-3,9999**	**-**	**24**	**1075,346**	-7	1	0	0,047	-3,9981	10^{-2}	TLE	-4
ex4_1_1	**-7,4873**	**-**	**100**	**1069,204**	**-7,4873**	**-**	**100**	**0,039**	-0,084	10^{-1}	TLE	-7,4873
ex4_1_3	-443,6716	-	100	39,368	**-443,6717**	**-**	**100**	**0,038**	-4,2948	10^{-1}	1,085	-443,672
ex4_1_4	**0**	**-**	**100**	**1016,907**	**0**	**-**	**100**	**0,039**	**0**	**10^{-2}**	**36,05**	0
ex4_1_5	$5 \cdot 10^{8}$	-	76	34,845	**1991,6666**	**-**	**100**	**0,068**	-19938,9	10	305,3	0
ex4_1_6	**7**	**-**	**100**	**15,3**	**7**	**-**	**100**	**0,034**	6,9425	10^{-4}	105,4	7
ex4_1_7	6	-	100	56,28	**-7,5**	**-**	**100**	**0,031**	-3,9062	10^{-1}	TLE	-7,5
ex4_1_8	**-16,7388**	**-**	**100**	**17,145**	-16,7388	-	100	0,047	-16,7338	10^{-3}	0,208	-16,739

Tabela 5.4: Resultados da melhor execução do InDE, do DE e do OGRe - parte 1.

	InDE				DE				OGRe			
instância	solução	violação	% fact.	tempo (s)	solução	violação	% fact.	tempo (s)	solução	ε	tempo (s)	ótimo
ex4_1_9	-3	-	40	123,617	**-5,508**	**-**	**100**	**0,074**	-6,2861	10^{-2}	0,365	-5,508
ex5_2_2_case1	**-351,4791**	**-**	**36**	**688,734**	-3900	300	0	0,137	-769,892	1	994,8	-400
ex5_2_2_case2	**-386,7566**	**-**	**40**	**590,803**	-8400	800	0	0,172	-684,812	1	3635	-600
ex5_2_2_case3	**-749,6909**	**-**	**52**	**548,994**	-2900	200	0	0,133	-328,748	1	2414	-750
ex5_2_4	-435,5308	-	100	886,185	**-450**	**-**	**96**	**0,102**	-506,884	10^{-1}	2204	-450
ex5_2_5	-2012,8411	1,1217	0	TLE	**-3499,0018**	**-**	**100**	**0,682**	-4986,116	1	TLE	-3500
ex5_3_2	**1,8642**	**-**	**84**	**1240,345**	0,9979	20	0	0,433	1,8527	10^{-2}	190	1,8642
ex5_4_2	**7514,9515**	**-**	**100**	**228,377**	2100	569,9553	0	0,117	7158,75	10^{-1}	TLE	7512,23
ex7_2_10	**0,1**	**-**	**12**	**501,001**	**0,1**	**-**	**100**	**0,17**	0,1	10	0,001	0,1
ex7_2_1	1228,7935	-	76	4942,034	**1227,2260**	**-**	**100**	**0,141**	-411,251	1	0,02	1227,21
ex7_2_2	**-0,4516**	**-**	**100**	**224,506**	-1	0,6253	0	0,117	-0,4518	10^{-4}	612,8	-0,3888
ex7_2_3	14918,6167	0,0010	0	805,01	2100	1,1622	0	0,126	**4091,012**	$\mathbf{10^{-1}}$	**TLE**	7049,25
ex7_2_5	10122,5507	-	100	2192,539	**10122,4932**	**-**	**100**	**0,105**	10121,7	10^{-4}	1332	10122,5
ex7_2_6	-83,2497	-	92	174,961	-83,2497	-	48	0,053	**-83,2503**	$\mathbf{10^{-4}}$	**439,6**	-83,2503
ex7_3_2	**1,0898**	**-**	**96**	**94,903**	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,061	1,0897	10^{-4}	1849	1,0899
ex7_3_3	$1,4 \cdot 10^{17}$	-	44	539,368	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,101	**0,5893**	$\mathbf{10^{-1}}$	**269,1**	0,8175
ex7_3_4	$4,2 \cdot 10^{17}$	$6 \cdot 10^{-6}$	0	TLE	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,163	**0**	**5**	**10,65**	6,2746
ex7_3_5	$\mathbf{5 \cdot 10^{16}}$	**-**	**12**	**TLE**	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,198	0	10	16,38	1,2029
ex8_1_6	0	-	100	40,961	-10,086	-	100	0,041	**-10,086**	$\mathbf{10^{-4}}$	**0,187**	-10,086
ex8_1_7	**0,0293**	**-**	**100**	**3884,749**	607,0355	-	100	0,123	$+\infty$	10	22,07	0,0293
ex8_1_8	**-0,4516**	**-**	**100**	**224,735**	-1	0,6253	0	0,089	-0,4518	10^{-4}	615,6	-0,3888
ex8_4_1	0,6265	$8 \cdot 10^{-5}$	0	1252,332	**0,6186**	**-**	**28**	**0,413**	0	5	0,014	0,6186
ex8_4_2	1,4579	$1,7 \cdot 10^{-5}$	0	TLE	**0,4864**	**-**	**16**	**0,804**	0	5	0,007	0,4852
ex8_4_3	0,0539	0,0009	0	TLE	**0,0046**	**-**	**100**	**1,507**	0,0059	10^{-3}	TLE	0,0046
ex8_4_5	0,0004	$5 \cdot 10^{-6}$	0	4139,392	0	0,0388	0	0,258	**0,0004**	$\mathbf{10^{-2}}$	**TLE**	0,0003
ex9_1_4	-37,0000	-	100	137,292	**-37**	**-**	**16**	**0,15**	-37,0024	10^{-4}	0,115	-37
ex9_1_5	-0,9999	-	12	101,287	$-2 \cdot 10^{17}$	$4 \cdot 10^{17}$	0	0,196	**-1**	$\mathbf{10^{-1}}$	**0,154**	-1

Tabela 5.5: Resultados da melhor execução do InDE, do DE e do OGRe - parte 2.

	InDE				DE				OGRe			
instância	solução	violação	% fact.	tempo (s)	solução	violação	% fact.	tempo (s)	solução	ε	tempo (s)	ótimo
ex9_1_8	**-3,2499**	**-**	**100**	**125,422**	$-2 \cdot 10^{17}$	$3 \cdot 10^{17}$	0	0,213	-3,2502	10^{-4}	0,009	-3,25
ex9_2_1	**16,9999**	**-**	**36**	**529,783**	26	3	0	0,137	16,9996	10^{-4}	7,134	17
ex9_2_2	99,9999	-	100	116,594	100	60	0	0,137	**100**	$\mathbf{10^{-2}}$	**0,094**	100
ex9_2_3	4,9999	-	100	398,285	$-\infty$	$1 \cdot 10^{17}$	0	0,294	**0,0006**	$\mathbf{10^{-1}}$	**5,704**	0
ex9_2_4	**0,5**	**-**	**100**	**58,298**	2	$1 \cdot 10^{17}$	0	0,104	**0,5**	$\mathbf{10^{-4}}$	**0,018**	0,5
ex9_2_5	**4,9999**	**-**	**100**	**146,205**	$1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,136	4,9997	10^{-4}	0,339	5
ex9_2_6	9898,747	2,7512	0	132,974	0	1	0	0,259	**1,0962**	$\mathbf{10^{-1}}$	**872**	-1
ex9_2_7	**16,9999**	**-**	**40**	**280,422**	26	3	0	0,141	17,0005	10^{-1}	0,483	17
ex9_2_8	**1,5**	**-**	**100**	**40,004**	**1,5**	**-**	**100**	**0,067**	1,5005	1	0,015	1,5
harker	$7,7 \cdot 10^{16}$	0,0258	0	2036,669	**-519,18**	**-**	**100**	**0,506**	$2,5 \cdot 10^{17}$	10	TLE	-986,51
haverly	**-399,9994**	**-**	**72**	**604,584**	$-1 \cdot 10^{17}$	$9,9 \cdot 10^{16}$	0	0,175	-369,48	1	TLE	-400
himmel11	-30562,0162	-	80	623,815	-32217,431	2,3096	0	0,128	**-30666,9**	$\mathbf{10^{-1}}$	**36,07**	-30665,5
himmel16	**0,061**	$\mathbf{1,8 \cdot 10^{-5}}$	**0**	**TLE**	$-\infty$	$1 \cdot 10^{17}$	0	0,242	-3,5	5	0,003	-0,866
house	**-4500,0001**	**-**	**100**	**189,594**	$+\infty$	$1 \cdot 10^{17}$	0	0,135	-4500,1	10^{-4}	TLE	-4500
hydro	**4368843,908**	**-**	**100**	**1033,37**	862444,8	4359,4636	0	0,639	3018718	1	1,829	4366944
immun	**0**	**-**	**100**	**2068,999**	3625000000	-	100	0,366	0	1	0,002	0
linear	-2000	2,7434	0	TLE	**0**	$\mathbf{1 \cdot 10^{17}}$	**0**	**0,416**	-2000	10	0,003	89
meanvar	**5,6578**	$6 \cdot 10^{-6}$	**0**	**TLE**	0	1,115	0	0,098	6,7883	1	TLE	5,2434
mhw4d	**0,0957**	**-**	**48**	**248,839**	$1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,119	0	5	0	0,0293
process	**-1161,2937**	**-**	**96**	**1189,655**	-1117,0914	-	100	0,182	-13765,7	1	0,014	-1161,34
prolog	$3,7 \cdot 10^{16}$	-	96	TLE	**1,0621**	**-**	**100**	**0,397**	0	5	TLE	0
rbrock	**0**	**-**	**100**	**7,255**	**0**	**-**	**100**	**0,056**	0	10	0	0
sambal	**3,9681**	**-**	**48**	**910,921**	$1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,197	3,9562	10^{-1}	TLE	3,9682
sample	**726,6793**	**-**	**100**	**53,564**	**726,6793**	**-**	**100**	**0,072**	726,66	10^{-4}	4517	726,679
ship	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	32	1575,121	$-\infty$	$1 \cdot 10^{17}$	0	0,213	**2,1282**	**5**	**TLE**	0
wall	-1,0135	0,512	0	127,303	$-1 \cdot 10^{17}$	$1 \cdot 10^{17}$	0	0,11	**-1**	$\mathbf{10^{-3}}$	**4,779**	1

Tabela 5.6: Resultados da melhor execução do InDE, do DE e do OGRe - parte 3.

Ainda que o tempo de execução do InDE seja consideravelmente maior que o tempo do DE caixa-preta, o InDE encontrou 72,22% de soluções factíveis a mais que o DE e 34,78% a mais que o OGRe. Além disso, em geral as soluções encontradas pelo InDE possuem maior aproximação ao ponto ótimo global de cada instância do que as encontradas pelos outros algoritmos.

Um aspecto a ser observado é que a decomposição Epífita utilizada no InDE e no OGRe adiciona variáveis auxiliares à instância, o que resulta em uma representação com maior dimensionalidade em relação à modelagem original. Foi observado na Seção 5.1 que mesmo se considerarmos somente o conjunto Ω de cada instância selecionada do *benchmark* COCONUT, são escassos os casos em que se encontra uma nova representação com dimensionalidade menor que a original. Ainda assim, os testes relatados neste capítulo revelam que a utilização de decomposição Epífita em conjunto com as demais estratégias do InDE é uma abordagem promissora para otimização numérica com restrições.

5.3 CONSIDERAÇÕES

Neste capítulo buscou-se avaliar o desempenho do método de Evolução Diferencial Intervalar (InDE) apresentado no Capítulo 4. O InDE utiliza informações de uma decomposição Epífita da instância e um procedimento de consistência local associados a diversas tecnologias estado-da-arte em DE. Assim, a otimização da instância NCOP ocorre por meio de operações sobre um subconjunto de variáveis consideradas críticas e as atribuições de valores são propagadas para as demais por meio do hipergrafo de restrições ternárias da instância.

A avaliação experimental deste estudo foi realizada a partir de 80 instâncias selecionadas do *benchmark* COCONUT. Inicialmente, foi avaliado o impacto da decomposição Epífita sobre o número de variáveis e restrições da instância, o que revelou que na maior parte das instâncias o conjunto Ω possui maior dimensionalidade que a modelagem original do *benchmark*.

Na sequência, realizamos uma análise comparativa de desempenho entre o InDE, o OGRe e uma variante de DE que utiliza as mesmas tecnologias do InDE no contexto de otimização caixa-preta, o que significa que não são explorados os recursos de decomposição estrutural e consistência local. As atribuições de valores encontradas pelo InDE foram superiores às do OGRe e do DE caixa-preta tanto em termos de factibilidade quanto de aproximação ao ponto ótimo global de cada instância. Isso indica que os recursos de decomposição estrutural e consistência local utilizados no InDE melhoram consideravelmente o desempenho do DE.

O InDE apresentou menor tempo de processamento em relação ao OGRe. No entanto, o DE caixa-preta encontrou atribuições de valores em um tempo consideravelmente menor que os outros algoritmos. Uma alternativa para redução do tempo de processamento seria a reestruturação e otimização da implementação responsável pela propagação dos valores pelo hipergrafo de restrições e pela consistência local utilizada no InDE e no OGRe. De modo geral, ainda que o tempo de processamento do InDE seja superior ao DE caixa-preta, os resultados obtidos demonstram que a utilização dos recursos de representação intervalar, decomposição estrutural, consistência local e otimização pelo conjunto Ω da decomposição Epífita são direções de pesquisa promissoras para trabalhos futuros.

6 CONCLUSÃO E TRABALHOS FUTUROS

O principal objetivo deste trabalho foi avaliar a associação entre abordagens estado-da-arte da área de otimização restrita por meta-heurísticas ao método de decomposição Epífita, que utiliza informações da estrutura analítica da instância codificada na forma ternária no decorrer do processo de otimização.

Inicialmente, foram estudados os principais conceitos relacionados à otimização intervalar, técnicas de consistência local e decomposição Epífita. Na sequência, foi realizada uma revisão bibliográfica acerca das principais abordagens adaptativas e técnicas de tratamento de restrições da literatura recente e dos algoritmos das últimas edições da *Special Session & Competitions on Real-Parameter Single Objective Optimization*, organizada durante o *IEEE Congress on Evolutionary Computation* (CEC). Identificou-se a tendência da utilização de Evolução Diferencial (DE) (Storn e Price, 1995) associada à mecanismos de adaptação de parâmetros (Trivedi et al., 2018; Poláková, 2017) e ao método de tratamento de restrições ε*-constrained* (Takahama e Sakai, 2010).

A partir disso, foi desenvolvido o algoritmo InDE (do inglês *Interval Differential Evolution*), que integra tecnologias estado-da-arte em otimização por meta-heurísticas e recursos de decomposição Epífita e consistência local. A utilização dessa decomposição permite que o InDE aplique seus operadores apenas sobre um subconjunto de elementos da estrutura da instância considerados críticos para o processo de busca, enquanto os demais são atribuídos por propagação de valores pelo hipergrafo de restrições ternárias da instância. Dessa forma, a população do InDE é constituída por um conjunto de caixas que cobre parte do espaço de busca, em vez de somente pontos específicos, como no DE clássico. A cada avaliação de *fitness*, a técnica de consistência local (GAC) empregada poda atribuições de valores infactíveis dos intervalos, o que pode acelerar a convergência do processo de busca.

A avaliação experimental deste estudo foi realizada sobre um conjunto de 80 instâncias selecionadas do *benchmark* COCONUT. A análise da decomposição Epífita revelou que na maior parte das instâncias o conjunto Ω possui maior dimensionalidade que a modelagem original do *benchmark*. Para avaliação de desempenho foram considerados os algoritmos InDE, OGRe e uma versão de DE para otimização caixa-preta. Nos testes realizados, o InDE encontrou uma maior quantidade de soluções factíveis ($77,5\%$) e com maior aproximação ao ótimo global de cada instância em relação aos demais algoritmos. Dessa forma, evidenciou-se que a utilização dos recursos de decomposição estrutural e consistência local melhora significativamente o desempenho do algoritmo DE.

Uma das principais limitações do InDE é o tempo de processamento. Ainda que sua execução tenha sido mais rápida que a do OGRe, quando analisamos o tempo de processamento do DE para otimização caixa-preta observa-se que a utilização dos recursos de decomposição estrutural e consistência local tornam o InDE e o OGRe mais dispendiosos. Acredita-se que a reestruturação e otimização da implementação desses recursos possa reduzir significativamente o tempo de execução tanto do InDE quanto do OGRe.

Outra fragilidade encontra-se no ajuste das configurações de parâmetros do InDE. Ainda que tenham sido aplicadas diversas estratégias adaptativas a fim de reduzir a quantidade de parâmetros do DE clássico como F, CR e a escolha da estratégia de mutação, restaram parâmetros cujo ajuste possui impacto considerável no desempenho do otimizador. Trabalhos futuros incluem a análise e implementação de outras abordagens adaptativas que possam minimizar a necessidade de ajuste desses parâmetros.

Propõe-se a extensão das técnicas de decomposição Epífita e consistência local utilizadas no InDE para outras meta-heurísticas que têm se mostrado promissoras na literatura recente, como *Matrix Adaptation Evolution Strategy* (MA-ES), comentada na Seção 3.2. Da mesma forma, a utilização de outras técnicas de consistência local que aceleram a busca ou podam atribuições infactíveis pode melhorar significativamente os resultados do processo de otimização do InDE.

A biblioteca multi-intervalar do OGRe empregada no InDE atualmente considera apenas operadores algébricos básicos. A implementação dessa biblioteca pode ser estendida a outros operadores e, com isso, espera-se atacar uma variedade maior de instâncias. Além disso, pretende-se estudar a viabilidade da adaptação das abordagens utilizadas no InDE para o tratamento de variáveis e restrições de instâncias de otimização discreta.

REFERÊNCIAS

Araya, I. e Reyes, V. (2016). Interval branch-and-bound algorithms for optimization and constraint satisfaction: a survey and prospects. *Journal of Global Optimization*, 65(4):837–866.

Benhamou, F., McAllester, D. e van Hentenryck, P. (1994). CLP(intervals) revisited. Em *Proceedings of the 1994 International Symposium on Logic Programming*, ILPS '94, páginas 124–138, Cambridge, MA, USA. MIT Press.

Benhamou, F. e Older, W. J. (1992). Applying interval arithmetic to real, integer and boolean constraints. Relatório técnico, BNR, Bell Northern Research.

Benhamou, F. e Older, W. J. (1997). Applying interval arithmetic to real, integer, and boolean constraints. *The Journal of Logic Programming*, 32(1):1 – 24.

Berge, C. (1985). *Graphs and Hypergraphs*. Elsevier Science Ltd., Oxford, UK, UK.

Beyer, H.-G. e Sendhoff, B. (2017). Simplify your covariance matrix adaptation evolution strategy. *IEEE Transactions on Evolutionary Computation*, 21(5):746–759.

Boussaïd, I., Lepagnot, J. e Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82–117.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287.

Das, S., Maity, S., Qu, B.-Y. e Suganthan, P. N. (2011). Real-parameter evolutionary multimodal optimization - a survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338.

Dechter, R. e van Beek, P. (1997). Local and global relational consistency. *Theoretical Computer Science*, 173(1):283 – 308.

Derenievicz, G. A. (2018a). OGRe - otimização global relaxada. `https://gitlab.c3sl.ufpr.br/gaderenievicz/OGRe`. Acessado em 02/05/2019.

Derenievicz, G. A. (2018b). *Uma condição suficiente para otimização global sem retrocesso*. Tese de doutorado, Universidade Federal do Paraná, Curitiba - Brazil.

Derenievicz, G. A. e Silva, F. (2018). Epiphytic trees: Relational consistency applied to global optimization problems. Em *International Conference on the Integration of Constraint Programming, Artificial Intelligence and Operations Research*, páginas 153–169. Springer.

Fan, Z., Fang, Y., Li, W., Yuan, Y., Wang, Z. e Bian, X. (2018). LSHADE44 with an improved ε constraint-handling method for solving constrained single-objective optimization problems. Em *2018 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Gong, W. e Cai, Z. (2013). Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics*, 43(6):2066–2081.

Gonçalves Júnior, J. d. S. (2016). Abordagem metaheurística híbrida para otimização do planejamento de estiva de navios porta-contêineres. Dissertação de Mestrado, Universidade do Vale do Rio dos Sinos, São Leopoldo - RS.

Hansen, E. e Walster, G. W. (2004). *Global optimization using interval analysis*. Monographs and textbooks in pure and applied mathematics. Marcel Dekker, New York.

Hansen, N. (2006). The CMA evolution strategy: a comparing review. Em *Towards a New Evolutionary Computation*, páginas 75–102. Springer.

Hellwig, M. e Beyer, H.-G. (2018). A matrix adaptation evolution strategy for constrained real-parameter optimization. Em *2018 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320.

Homaifar, A., Qi, C. X. e Lai, S. H. (1994). Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–253.

Kearfott, R. B. (1992). An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2(3):259–280.

Liang, J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello, C. C. e Deb, K. (2006). Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41(8):8–31.

Mackworth, A. K. (1977). On reading sketch maps. Em *5th International Joint Conference on Artificial Intelligence*, páginas 598–606, San Francisco. Morgan Kaufmann.

Mallipeddi, R. e Suganthan, P. N. (2010). Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 24.

Mezura-Montes, E. e Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194.

Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary Programming*, 4:135–155.

Moore, R. E. (1966). *Interval Analysis*. Prentice-Hall Englewood Cliffs, N.J.

Morales, A. K. e Quezada, C. V. (1998). A universal eclectic genetic algorithm for constrained optimization. Em *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing*, volume 1, páginas 518–522.

Neri, F. e Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33(1-2):61–106.

Ortiz, G. A. (2012). Rosenbrock's function in 3D. `https://goo.gl/z5Yojq`, sob a licença CC BY-SA 3.0, disponível em `https://goo.gl/L3bYmf`. Acessado em 17/05/2018.

Poláková, R. (2017). L-SHADE with competing strategies applied to constrained optimization. Em *2017 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1683–1689. IEEE.

Poláková, R., Tvrdík, J. e Bujok, P. (2016a). Evaluating the performance of L-SHADE with competing strategies on CEC2014 single parameter-operator test suite. Em *2016 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1181–1187. IEEE.

Poláková, R., Tvrdík, J. e Bujok, P. (2016b). L-SHADE with competing strategies applied to CEC2015 learning-based test suite. Em *2016 IEEE Congress on Evolutionary Computation (CEC)*, páginas 4790–4796. IEEE.

Potra, F. A. e Wright, S. J. (1997). *Primal-dual interior-point methods*. SIAM.

Price, K. e Storn, R. (1995). Differential evolution (DE) for continuous function optimization. `http://www1.icsi.berkeley.edu/~storn/DenewC.zip`. Acessado em 02/05/2019.

Qin, A. K., Huang, V. L. e Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417.

Reeves, C. R. (1993). *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc.

Runarsson, T. P. e Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.-H. e Nguyen, T.-V. (2003). *Benchmarking Global Optimization and Constraint Satisfaction Codes*, páginas 211–222. Springer Berlin Heidelberg, Berlin, Heidelberg.

Storn, R. e Price, K. (1995). Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous space. *Technical Report TR-95-012, International Computer Science Institute*.

Suganthan, P. N. (2018). Benchmarks for evaluation of evolutionary algorithms congress on evolutionary computation. `http://www.ntu.edu.sg/home/epnsugan/index_files/cec-benchmarking.htm`. Acessado em 02/05/2019.

Takahama, T. e Sakai, S. (2005). Constrained optimization by ε constrained particle swarm optimizer with ε-level control. Em *Soft Computing as Transdisciplinary Science and Technology*, páginas 1019–1029. Springer.

Takahama, T. e Sakai, S. (2010). Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. Em *2010 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–9. IEEE.

Tanabe, R. e Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. Em *2013 IEEE Congress on Evolutionary Computation (CEC)*, páginas 71–78. IEEE.

Tanabe, R. e Fukunaga, A. S. (2014). Improving the search performance of SHADE using linear population size reduction. Em *2014 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1658–1665. IEEE.

Tang, L., Dong, Y. e Liu, J. (2015). Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation*, 19(4):560–574.

Trivedi, A., Sanyal, K., Verma, P. e Srinivasan, D. (2017). A unified differential evolution algorithm for constrained optimization problems. Em *2017 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1231–1238. IEEE.

Trivedi, A. e Srinivasan, D. (2018). Empirical investigations into the composite differential evolution on CEC2017 constrained optimization problems. Em *2018 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, páginas 280–285. IEEE.

Trivedi, A., Srinivasan, D. e Biswas, N. (2018). An improved unified differential evolution algorithm for constrained optimization problems. Relatório técnico, 2018 IEEE Congress on Evolutionary Computation (CEC).

Tvrdık, J. (2006). Competitive differential evolution. Em *12th International Conference on Soft Computing*, páginas 7–12.

Tvrdík, J. e Poláková, R. (2017). A simple framework for constrained problems with application of L-SHADE44 and IDE. Em *2017 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1436–1443. IEEE.

Van Hentenryck, P. (1997). Numerica: A modeling language for global optimization. Em *15th International Joint Conference on Artificial Intelligence*, páginas 1642–1647, San Francisco. Morgan Kaufmann.

Wang, B.-C., Li, H.-X., Li, J.-P. e Wang, Y. (2018). Composite differential evolution for constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Wang, Y., Cai, Z. e Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55–66.

Wu, G., Mallipeddi, R. e Suganthan, P. (2017). Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. Relatório técnico.

Zamuda, A. (2017). Adaptive constraint handling and success history differential evolution for CEC 2017 constrained real-parameter optimization. Em *2017 IEEE Congress on Evolutionary Computation (CEC)*, páginas 2443–2450. IEEE.

Zhang, J. e Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958.

Zhang, W.-J., Xie, X.-F. e Bi, D.-C. (2004). Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. Em *2004 Congress on Evolutionary Computation*, volume 2, páginas 2307–2311. IEEE.