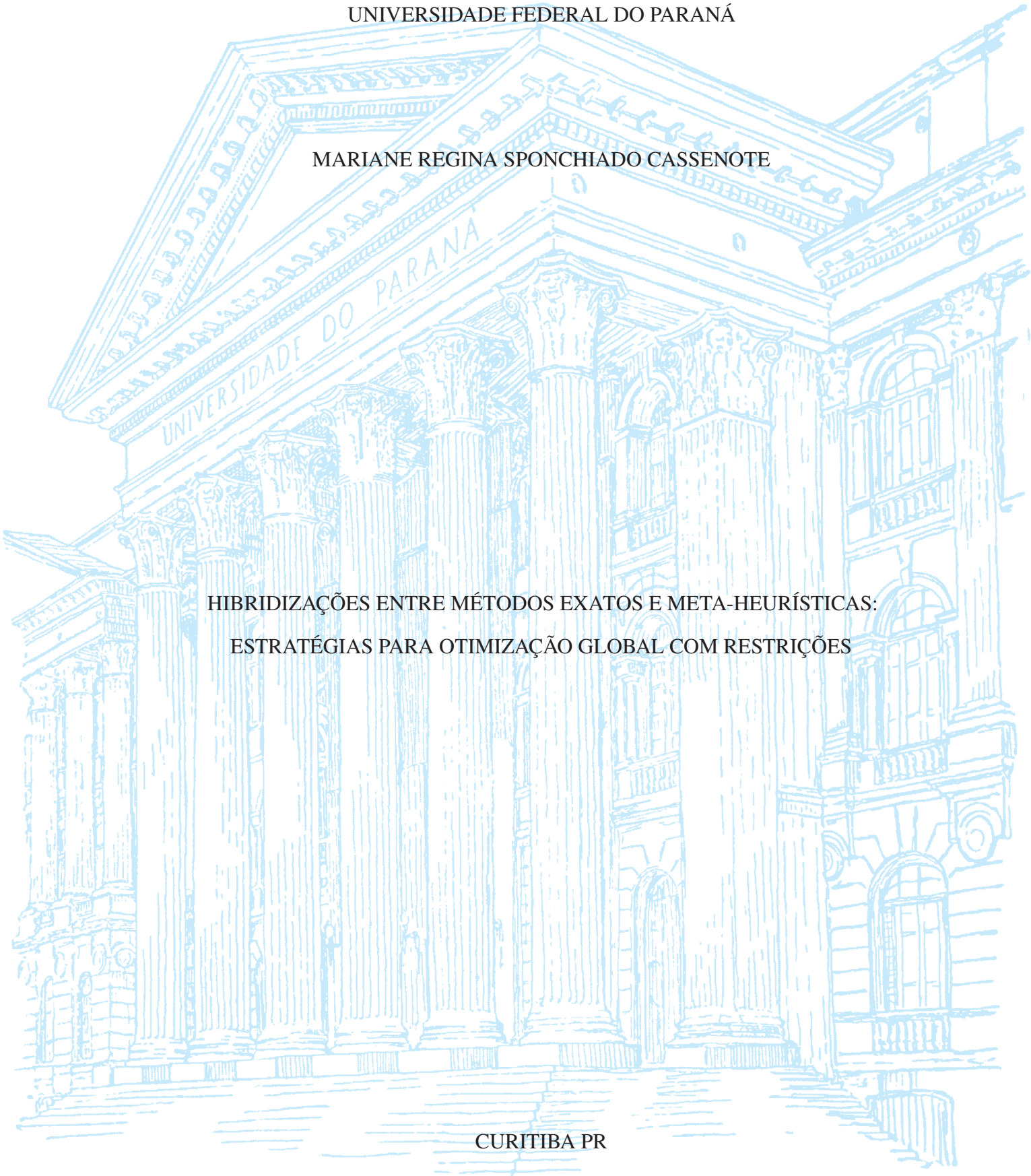
UNIVERSIDADE FEDERAL DO PARANÁ

MARIANE REGINA SPONCHIADO CASSENOTE




# HIBRIDIZAÇÕES ENTRE MÉTODOS EXATOS E META-HEURÍSTICAS: ESTRATÉGIAS PARA OTIMIZAÇÃO GLOBAL COM RESTRIÇÕES

CURITIBA PR

2024

MARIANE REGINA SPONCHIADO CASSENOTE

# HIBRIDIZAÇÕES ENTRE MÉTODOS EXATOS E META-HEURÍSTICAS: ESTRATÉGIAS PARA OTIMIZAÇÃO GLOBAL COM RESTRIÇÕES

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

Área de concentração: *Computação*.

Orientador: Fabiano Silva.

Coorientador: Guilherme Alex Derenievicz.

CURITIBA PR

2024

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Cassenote, Mariane Regina Sponchiado
Hibridizações entre métodos exatos e meta-heurísticas: estratégias para otimização global com restrições / Mariane Regina Sponchiado Cassenote. – Curitiba, 2024.
1 recurso on-line : PDF.

Tese (Doutorado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.

Orientador: Fabiano Silva
Coorientador: Guilherme Alex Derenievicz

1. Otimização global. 2. Programação não convexa. 3. Meta-heurísticas. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Informática. III. Silva, Fabiano. IV. Derenievicz, Guilherme Alex. V. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

# TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **MARIANE REGINA SPONCHIADO CASSENOTE** intitulada: **Hibridizações entre métodos exatos e meta-heurísticas: estratégias para otimização global com restrições**, sob orientação do Prof. Dr. FABIANO SILVA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutora 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, 09 de Agosto de 2024.

Assinatura Eletrônica
12/08/2024 15:11:05.0
FABIANO SILVA
Presidente da Banca Examinadora

Assinatura Eletrônica
12/08/2024 10:40:46.0
MARCELO RODRIGUES BESSA
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
12/08/2024 14:23:03.0
ANDRÉ LUIZ PIRES GUEDES
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
12/08/2024 13:51:05.0
MYRIAM REGATTIERI DE BIASE DA SILVA DELGADO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
12/08/2024 10:33:17.0
GUILHERME ALEX DERENIEVICZ
Coorientador(a) (UNIVERSIDADE FEDERAL DO PARANÁ)

*À minha família, fonte de toda força,*
*coragem e amor que existem em mim.*

## AGRADECIMENTOS

Certa vez, um sábio professor (alô, Roberto Pereira) me disse com confiança: “O maior produto do seu doutorado não será a sua tese, será você”. Com isso, ele não diminuía o valor do meu trabalho, mas se referia a algo muito maior do que as dezenas de páginas repletas de termos técnicos e equações. Hoje, ao olhar para a minha jornada na pós-graduação, entendo plenamente a vastidão de experiências, desafios, momentos de pura euforia, e tantos outros de incertezas aos quais ele se referia. A universidade me transformou em muitos aspectos. Me fez mais madura, humana, empática, curiosa e apaixonada por (buscar) descobrir as coisas da vida. À UFPR, que me acolheu e foi minha segunda casa por sete anos, minha profunda gratidão.

Ao meu orientador e amigo, Fabiano Silva, agradeço pela oportunidade, confiança e inesgotável dedicação. Ao meu coorientador, Guilherme Derenievicz, pela parceria e por tornar didáticos conceitos que eu não imaginava que dominaria. Aos professores Marcos Castilho, André Guedes, André Grégio, Roberto Pereira, Leticia Peres, Eduardo Spinosa, Myriam Delgado e Marcelo Bessa, que contribuíram direta ou indiretamente para a minha formação, meu sincero agradecimento.

Ao Centro de Computação Científica e Software Livre (C3SL), pelo suporte técnico que possibilitou todas as avaliações experimentais realizadas durante meu período na pós-graduação.

Aos meus pais e ao meu irmão, que estiveram presentes todos os dias, sem exceção, mesmo a mais de 700 km de distância. Ter um lar amoroso para onde voltar é o maior privilégio e a maior riqueza que eu poderia ter. Cada olhar carinhoso, palavra de incentivo e gesto de apoio me deram coragem e força para seguir em frente. Esta conquista também é de vocês.

Aos meus tios, Ieda e Edgar, pelo carinho em sempre buscarem saber como eu estava. Aos meus avós e à minha cunhada, por todo o apoio. Em especial, ao meu avô Angelino, que desde muito cedo incentivou e investiu em minha educação e que certamente estaria feliz em comemorar a conclusão desta etapa em minha vida. Embora não esteja presente, sua memória me trouxe, em muitos momentos, a coragem e a determinação que eu precisava.

Ao Felipe, pelo carinho, apoio e companheirismo diários e, especialmente, por estar ao meu lado durante alguns dos momentos mais críticos do desenvolvimento deste trabalho.

Aos amigos do LIAMF, que também se tornaram irmãos de caminhada, Leonam Oliveira, Tamy Beppler, Marcelino Ulica e Jonivan Oliveira, pelo muito que vivemos juntos.

A Deus e todas as energias boas que regem o universo, que sempre me levaram de volta para casa.

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

*"My beautiful mother*
*She told me: Son, in life you're gonna go far*
*If you do it right, you'll love where you are*
*Just know, wherever you go*
*You can always come home*
[...]
*My irrefutable father*
*He told me: Son, sometimes it may seem dark*
*But the absence of the light is a necessary part*
*Just know, you're never alone*
*You can always come back home"*

*(93 Million Miles – Jason Mraz)*

## RESUMO

A otimização numérica global tem sido amplamente utilizada na modelagem de problemas em diversas áreas do conhecimento. Este processo consiste em encontrar uma valoração para variáveis reais que minimize uma função de custo. Embora os métodos propostos pelas comunidades de programação matemática, métodos exatos e meta-heurísticas apresentem desempenho satisfatório em alguns subconjuntos de instâncias, variantes com restrições, não-convexidade, não-linearidade ou alta dimensionalidade ainda representam desafios significativos. Nesse contexto, métodos híbridos surgem como alternativas promissoras, combinando vantagens e características complementares de diferentes abordagens. Com o objetivo de abrir novas perspectivas e metodologias em otimização numérica global mono-objetivo com restrições, esta tese propõe três otimizadores híbridos que combinam técnicas de contração do espaço de busca, típicas de métodos exatos, com a meta-heurística Evolução Diferencial (DE), destacada na literatura recente. O primeiro otimizador, BBDE, baseia-se em DE e incorpora diversas estratégias de resolvedores recentes e técnicas de consistência local, melhorando substancialmente seu desempenho. Em seguida, o HIDE integra uma versão intervalar de DE com técnicas de consistência local e uma derivação do BBDE, resultando em um desempenho superior ao da abordagem anterior. Por último, o HIBB combina um método híbrido de *Branch & Bound* (B&B) com consistência local e uma meta-heurística baseada em BBDE, avaliando diferentes formas de exploração do espaço de busca para melhor aproveitamento dos recursos computacionais disponíveis. As abordagens híbridas propostas apresentaram resultados significativamente superiores às suas versões originais, explorando de maneira eficaz as vantagens individuais de métodos exatos e meta-heurísticas.



Palavras-chave: Otimização Global. Métodos híbridos. Consistência local. Meta-heurísticas.

## ABSTRACT

Global numerical optimization has been widely used in modeling problems across various fields of knowledge. This process involves finding a valuation for real variables that minimizes a cost function. Although the methods proposed by the communities of mathematical programming, exact methods, and metaheuristics show satisfactory performance in some subsets of instances, variants with constraints, non-convexity, non-linearity, or high dimensionality still pose significant challenges. In this context, hybrid methods emerge as promising alternatives, combining different approaches' advantages and complementary characteristics. Aiming to introduce new perspectives and methodologies in single-objective global numerical optimization with constraints, this thesis proposes three hybrid optimizers that combine search space contraction techniques, typical of exact methods, with the Differential Evolution (DE) metaheuristic, highlighted in recent literature. The first optimizer, BBDE, is based on DE and incorporates various strategies from recent solvers and local consistency techniques, substantially improving its performance. Next, the HIDE integrates an interval version of DE with local consistency techniques and a derivation of BBDE, resulting in superior performance compared to the previous approach. Finally, the HIBB combines a hybrid Branch and Bound (B&B) method with local consistency and a BBDE-based metaheuristic, evaluating different ways of exploring the search space to utilize the available computational resources better. The proposed hybrid approaches presented significantly superior results to their original versions, effectively leveraging the individual advantages of exact methods and metaheuristics.



Keywords: Global Optimization. Hybrid methods. Local consistency. Metaheuristics.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| AMPL | *A Mathematical Programming Language* |
| BBDE | *Black-box Differential Evolution* |
| B&B | *Branch and Bound* |
| BS | *Beam Search* |
| C3SL | Centro de Computação Científica e Software Livre |
| CAL-SHADE | *Constraint Handling with Success History Adaptive Differential Evolution* |
| CEC | *IEEE Congress on Evolutionary Computation* |
| CMA-ES | *Covariance Matrix Adaptation Evolutionary Strategy* |
| CSP | *Constraint Satisfaction Problem* |
| DAC | *Directionally Arc Consistent* |
| DE | *Differential Evolution* |
| En-MODE | *Enhanced Multi-Operator Differential Evolution* |
| EPSCONT | $\varepsilon$-contração |
| EQPC | Ordenação por estagnação e qualidade do ponto da caixa |
| FCHA | *Fuzzy Controlled Cooperative Heterogeneous Algorithm* |
| GA | *Genetic Algorithm* |
| GAC | *Generalized Arc-Consistency* |
| IDE | *Individual-Dependent Mechanism* |
| I2DE | *Improved Interval Differential Evolution* |
| InDE | *Interval Differential Evolution* |
| IUDE | *Improved Unified Differential Evolution* |
| HIBB | *Hybrid Interval Branch and Bound* |
| HIDE | *Hybrid Interval Differential Evolution* |
| MA | *Memetic Algorithm* |
| MA-ES | *Matrix Adaptation Evolution Strategy* |
| MINLPLib | *Mixed-Integer Nonlinear Programming Library* |
| *NB* | Número de frentes de busca |
| *NP* | Tamanho da população |
| PPGINF | Programa de Pós Graduação em Informática |
| PSO | *Particle Swarm Optimization* |
| QPC | Ordenação por qualidade do ponto da caixa |
| SASS | *Self-Adaptive Spherical Search* |
| SAT | Problema de satisfatibilidade booleana |
| SHADE | *Success-History based Adaptive Differential Evolution* |

| | |
|---|---|
| SOF | *Superiority of Feasible Points* |
| TLE | Tempo Limite Excedido |
| UDE | *Unified Differential Evolution* |
| UFPR | Universidade Federal do Paraná |

**SUMÁRIO**

# 1 INTRODUÇÃO

A otimização numérica é uma área de pesquisa que permeia diversos campos da ciência e da indústria, incluindo matemática, engenharias, física e economia (Malik et al., 2021; Martins e Ning, 2021; Hussain et al., 2019; Gilli et al., 2019; Gmys et al., 2020). Em contextos industriais e científicos, inúmeros problemas são modelados como problemas de otimização global, cujo objetivo é identificar a melhor solução possível para uma instância de um problema. A capacidade de resolver esses problemas de forma eficaz e eficiente tem um impacto significativo no avanço tecnológico e econômico, destacando a importância contínua do desenvolvimento de métodos de otimização robustos e inovadores (Hillier e Lieberman, 2013).

Uma instância de problema de otimização é modelada matematicamente, geralmente por uma *função objetivo* ou *função de custo*, que está associada a variáveis e restrições (Bertsekas et al., 1999). Existem diferenças significativas entre problemas de otimização combinatória e problemas de otimização numérica. Problemas combinatórios envolvem a busca pela melhor solução em um conjunto finito ou contável de possibilidades, muitas vezes relacionadas à organização ou escolha de subconjuntos, com o objetivo de otimizar uma função objetivo sob restrições (Papadimitriou e Steiglitz, 2013; Korte et al., 2011; Schrijver et al., 2003). Já na otimização numérica, o objetivo é encontrar valores contínuos que minimizem ou maximizem a função objetivo dentro de um espaço de busca definido por restrições, sendo frequente o uso de métodos para problemas não-lineares e não-convexos (Nocedal e Wright, 2006; Bazaraa et al., 2006).

Em geral, um *problema de otimização numérica global mono-objetivo com restrições* sobre as variáveis reais $x = (x_1, \ldots, x_D)$ pode ser descrito como a seguir:

$$\begin{aligned} \min \quad & f(x) \\ \text{sujeito a} \quad & g_j(x) \leq 0 \quad j \in \{1, \ldots, m\}, \\ & h_k(x) = 0 \quad k \in \{1, \ldots, n\}, \end{aligned}$$

em que $f$ é a função objetivo a ser minimizada, $g_j$ denota uma das $m$ restrições de desigualdade, e $h_k$ denota uma das $n$ restrições de igualdade. Uma função $f$ a ser maximizada pode ser representada como a minimização de $(-f)$. Aqui, foca-se em problemas mono-objetivo, ou seja, com uma única função objetivo.

Uma valoração das variáveis que não viole as restrições é chamada de *solução candidata*. Uma solução candidata é dita *factível* quando os valores associados às variáveis não violam nenhuma das restrições da instância. O conjunto de todas as valorações possíveis para as variáveis da instância define o *espaço de busca*. Um *ponto ótimo* no espaço de busca de uma instância de minimização é uma atribuição de valores que resulta no menor valor possível da função objetivo, ou seja, a *solução ótima*. Pode haver mais de um ponto ótimo em alguns casos, com diferentes soluções candidatas resultando no mesmo valor mínimo da função objetivo.

Em problemas de minimização, um *mínimo local* ocorre quando o valor da função objetivo é menor que todos os outros valores em uma pequena região do espaço de busca, mas não necessariamente em todo o espaço. Por outro lado, um *mínimo global* representa um ponto no qual o valor da função objetivo é menor em toda a extensão do espaço de busca factível.

Instâncias convexas são aquelas em que, para quaisquer dois pontos no domínio, a linha reta entre eles permanece acima ou sobre a superfície da função, garantindo que qualquer mínimo local seja também o mínimo global (Boyd e Vandenberghe, 2004). Instâncias não-convexas,

por outro lado, podem conter múltiplos mínimos locais, sendo classificadas como multimodais. Nessas instâncias, identificar o mínimo global é especialmente desafiador, uma vez que algoritmos de otimização podem ficar presos em mínimos locais, exigindo estratégias mais sofisticadas para escapar e continuar a busca pelo mínimo global (Das et al., 2011). Além disso, funções não-lineares introduzem complexidade matemática, frequentemente tornando inviável o uso de métodos analíticos diretos (Bertsekas, 2016; Nocedal e Wright, 2006). A Figura 1.1 ilustra uma função multimodal, destacando os mínimos locais e o mínimo global.

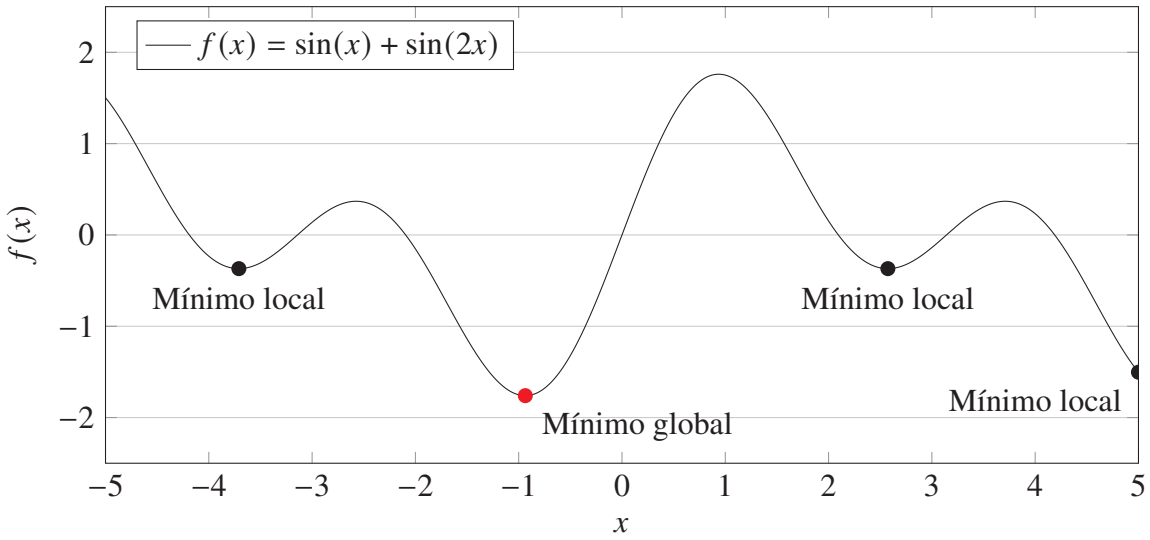



Figura 1.1: Representação de mínimos locais e global em uma função multimodal.

## 1.1 MOTIVAÇÃO

Métodos que empregam informações de convexidade e diferenciabilidade da função objetivo têm sido extensivamente explorados por pesquisadores nas áreas de programação matemática e pesquisa operacional. Exemplos disso são o Lagrangeano Aumentado (Hestenes, 1969) e os métodos de Pontos Interiores (Potra e Wright, 1997). Entretanto, a realidade prática apresenta desafios significativos, pois a maioria dos problemas de otimização no mundo real exibe características como não-convexidade, não-linearidade, multimodalidade e alta dimensionalidade. Essas características frequentemente tornam impraticáveis muitos dos algoritmos matemáticos clássicos, que podem não garantir a otimalidade das soluções e estagnar em mínimos locais do espaço de busca (Bandaru e Deb, 2016).

Métodos exatos, tais como *Branch & Bound* (B&B), busca com retrocesso e métodos intervalares, são capazes de garantir a otimalidade global das soluções encontradas ao percorrerem sistematicamente todo o espaço de busca. Especificamente, os métodos de B&B intervalares (Araya e Reyes, 2016; Kearfott, 1992; Hansen e Walster, 2004a) segmentam o espaço de busca e eliminam subespaços que certamente não contêm a solução ótima da instância. Para acelerar esse processo, técnicas de *consistência local* (por exemplo, o AC3 (Mackworth, 1977a)) são empregadas para refinar os domínios das variáveis enquanto a solução ótima é mantida. Contudo, apesar dessas vantagens, o número de subespaços processados pode crescer exponencialmente em razão da dimensionalidade da instância, o que resulta no consumo significativo de memória e no aumento no tempo de processamento. Esses fatores tornam a aplicação de métodos de B&B impraticável para a resolução de instâncias com alta dimensionalidade (Sun e Johnson, 2005).

Por outro lado, meta-heurísticas têm se destacado nas últimas décadas por sua capacidade de encontrar soluções de maneira computacionalmente eficiente, embora sem garantia de otimalidade da solução (Malik et al., 2021). A maioria dos métodos meta-heurísticos possuem características estocásticas e imitam algum princípio natural, físico ou biológico que se assemelha a um processo de busca ou otimização. Contudo, a forte dependência do ajuste de parâmetros pode prejudicar seu desempenho ao abordar instâncias complexas. Como não é possível provar a otimalidade das soluções obtidas, comumente é estabelecido um limite máximo de iterações para a execução desses algoritmos. Entretanto, ajustar esse parâmetro não é uma tarefa trivial e pode resultar em soluções substancialmente inferiores ao ótimo ou em um consumo excessivo de recursos computacionais.

As comunidades científicas de programação matemática, de pesquisa operacional, de métodos exatos e de meta-heurísticas têm demonstrado sua importância prática ao resolver uma grande diversidade de problemas (Osaba et al., 2021; Kaveh e Eslamlou, 2020; Hussain et al., 2019; Mérel e Howitt, 2014; Grossmann, 2014; Bhurjee e Panda, 2012; Van Kampen, 2010). No entanto, enfrentam desafios ao abordar instâncias de alta dimensionalidade, que excedem a capacidade de resolução eficiente de métodos exatos, ou com características que tornam o espaço de busca particularmente difícil para métodos convencionais de programação matemática e meta-heurísticas (Bandaru e Deb, 2016). Nesse contexto se encaixam os métodos híbridos, que combinam vantagens e características complementares de métodos exatos e de meta-heurísticas para alcançar boas soluções em um tempo de execução razoável.

A revisão da literatura revela que a maior parte das pesquisas desenvolvidas na área de métodos híbridos tende a focar em otimização combinatória ou em otimização numérica sem restrições (Raidl et al., 2019; Ozkan et al., 2019; Talbi, 2016; Blum e Raidl, 2016), deixando de contemplar uma grande quantidade de problemas do mundo real, particularmente aqueles envolvendo otimização numérica global com restrições. A partir dessa constatação, identifica-se uma oportunidade de pesquisa que abre caminho para o desenvolvimento de novas abordagens híbridas.

Nesse contexto, esta tese visa responder a seguinte pergunta de pesquisa: Métodos híbridos que combinam técnicas de otimização exata e meta-heurísticas podem melhorar a qualidade das soluções em relação às suas versões originais em problemas de otimização numérica global com restrições?

## 1.2 OBJETIVO

Esta tese tem como objetivo desenvolver e avaliar métodos híbridos de otimização que combinam técnicas de otimização exata e meta-heurísticas para melhorar o desempenho em relação às abordagens originais para problemas de otimização numérica global com restrições.

Especificamente, esse objetivo geral é explorado por meio de duas frentes principais:

- Integração de Evolução Diferencial com Técnicas de Consistência Local: Desenvolver versões híbridas da meta-heurística Evolução Diferencial que incorporem técnicas de consistência local, visando aprimorar a qualidade das soluções em comparação com sua versão original;

- Combinação de Branch & Bound com Consistência Local e Evolução Diferencial: Propor e avaliar um método híbrido que integra Branch & Bound, técnicas de consistência local e Evolução Diferencial, permitindo uma exploração mais eficiente do espaço de busca em problemas de otimização numérica global com restrições.

## 1.3 CONTRIBUIÇÕES

As principais contribuições elencadas neste documento incluem o desenvolvimento de um conjunto de ferramentas para a preparação e processamento de instâncias de otimização, realizado pelo grupo de pesquisa ao qual esta tese está associada, e a criação de três resolvedores híbridos que combinam as vantagens de métodos exatos e meta-heurísticas, desenvolvidos especificamente como parte desta pesquisa.

No que se refere às ferramentas desenvolvidas (Seção 4.1), destaca-se um *parser* que converte instâncias no formato AMPL (*A Mathematical Programming Language*) para duas representações distintas: uma modelagem em rede de restrições ternárias, aplicável a técnicas de consistência local, e um código-fonte que avalia a função objetivo e as violações de restrições a partir das valorações das soluções candidatas. É importante notar que a modelagem algébrica da instância é utilizada somente na etapa de consistência local dos otimizadores híbridos propostos. Além disso, foi realizada uma extensão significativa do núcleo multi-intervalar previamente introduzido por Derenievicz (2018), utilizado na etapa de consistência local dos otimizadores híbridos propostos. Originalmente, este núcleo abordava apenas operações básicas como $+$, $-$, $*$, $/$, $\wedge$ e $\sqrt{}$. Ao longo desta pesquisa, a implementação foi ampliada para incluir as operações *log*, *exp*, *sin*, *cos*, *tan*, *abs*, *sign*, *max* e *min*, permitindo abordar uma maior diversidade de instâncias.

Em relação aos resolvedores híbridos, o primeiro a ser introduzido é o BBDE (Seção 4.2), um otimizador meta-heurístico baseado em Evolução Diferencial que explora estocasticamente pontos específicos do espaço de busca. A versão básica do BBDE, publicada em Cassenote et al. (2021) incorpora várias heurísticas e estratégias adaptativas, tendo superado o desempenho de diversos otimizadores recentes da área. Nesta tese, foi adicionado ao BBDE um esquema de reinícios para evitar a estagnação do processo de busca e uma hibridização com técnicas de consistência local. Essas inovações permitiram que o BBDE alcançasse desempenho superior em comparação à sua versão básica.

O HIDE, detalhado na Seção 4.3, representa uma hibridização em três eixos: uma versão aprimorada de Evolução Diferencial Intervalar, técnicas de consistência local e uma busca local em subespaços baseada em BBDE. As estratégias de mutação intervalar para Evolução Diferencial, originalmente propostas em Cassenote et al. (2019) e Cassenote (2019), foram reformuladas para minimizar erros numéricos. A análise experimental indica que a etapa de consistência local eleva significativamente a qualidade das soluções, posicionando o HIDE como uma meta-heurística de desempenho superior ao BBDE.

Por fim, o HIBB (Capítulo 5) foi proposto como um método híbrido de *Branch & Bound* intervalar que combina uma técnica de consistência local, tipicamente usada em métodos de otimização exatos, com uma busca local baseada na meta-heurística BBDE. Este método foi enriquecido com a avaliação de diversas heurísticas que influenciam diretamente na maneira como o espaço de busca é explorado, incluindo diferentes estratégias de ordenação da fila de prioridade e o uso de múltiplas frentes de busca para uma distribuição mais eficaz dos recursos computacionais. A combinação dessas estratégias resultou em um otimizador robusto que demonstrou resultados superiores tanto ao BBDE quanto ao HIDE.

Até o momento, esta pesquisa gerou as seguintes publicações:

- Cassenote, M. R., Derenievicz, G. A., & Silva, F. (2021). I2DE: Improved Interval Differential Evolution for Numerical Constrained Global Optimization. Em Brazilian Conference on Intelligent Systems (pp. 186-201). Cham: Springer International Publishing. `https://doi.org/10.1007/978-3-030-91702-9_13`.

- Cassenote, M. R., Derenievicz, G. A., & Silva, F. (2024). A Hybrid Approach Integrating Generalized Arc Consistency and Differential Evolution for Global Optimization. Em International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (pp. 190-207). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-60597-0_13.

## 1.4 ORGANIZAÇÃO DA TESE

Esta tese está organizada da seguinte maneira:

- No Capítulo 2, são definidos os conceitos básicos relacionados a métodos exatos de otimização, aritmética intervalar, *Branch & Bound Intervalar* e técnicas de consistência local. Também são abordadas as principais características de meta-heurísticas para otimização global, especificamente Evolução Diferencial, e algumas das técnicas de tratamento de restrições mais comuns na literatura recente;

- No Capítulo 3, é realizada uma revisão bibliográfica de abordagens adaptativas utilizadas no contexto de meta-heurísticas para otimização numérica com restrições. Além disso, são elencadas diversas abordagens de hibridização entre métodos exatos e meta-heurísticas a fim de destacar algumas possibilidades de integração entre essas duas áreas de pesquisa;

- No Capítulo 4, são descritos os materiais e métodos utilizados na implementação e avaliação experimental desta tese. Em seguida, são apresentados o BBDE, uma Evolução Diferencial (DE) que integra diversas heurísticas para exploração estocástica de pontos do espaço de busca, e o HIDE, uma hibridização em três eixos que combina o BBDE com técnicas de consistência local e uma meta-heurística baseada em intervalos;

- No Capítulo 5, é proposto o HIBB, um método híbrido de *Branch & Bound* intervalar que integra técnicas de otimização exata e busca local baseada em meta-heurística. Diferentes heurísticas para exploração do espaço de busca são discutidas, assim como uma análise comparativa do desempenho com outros otimizadores;

- No Capítulo 6, conclui-se a tese revisando as principais contribuições e limitações da pesquisa e indicando possíveis trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Nas últimas décadas, diversas pesquisas têm sido desenvolvidas no contexto de otimização numérica. Métodos exatos conseguem garantir a otimalidade da solução encontrada, pois exploram informações de todo o espaço de busca da instância. No entanto, essas abordagens possuem um alto custo computacional, o que muitas vezes inviabiliza sua utilização em problemas com muitas dimensões. Por outro lado, meta-heurísticas são métodos estocásticos que buscam fornecer uma solução aproximada para uma instância em um tempo computacional aceitável, mas sem nenhum tipo de garantia sobre a qualidade dessa solução.

Na Seção 2.1 são introduzidos conceitos relacionados à Aritmética Intervalar, métodos de Branch & Bound Intervalar e técnicas de Consistência Local. Na Seção 2.2 são comentadas as principais características de meta-heurísticas, especialmente Evolução Diferencial (DE), além das técnicas de tratamento de restrições da literatura recente.

## 2.1 MÉTODOS EXATOS DE OTIMIZAÇÃO

Desde a década de oitenta, Aritmética Intervalar tem sido utilizada para resolução de problemas de otimização numérica contínua por meio da combinação de técnicas de consistência local com algoritmos de Branch & Bound (B&B) (Araya e Reyes, 2016; Hansen e Walster, 2004a; Kearfott, 1992). O processo de busca consiste em iterativamente dividir o espaço de busca em subespaços menores, ou "ramos" (*branches*), e explorá-los a fim de identificar regiões promissoras que podem conter a solução ótima. Ao longo desse processo, são aplicadas estratégias de "poda" (*bounding*) para descartar ramos que certamente não contêm a solução ótima, o que acaba por reduzir o espaço de busca. Adicionalmente, métodos de busca local são utilizados para detectar se o domínio atual contém uma solução factível que possa auxiliar no processo de poda, o que contribui para a aceleração da convergência (Sun e Johnson, 2005).

Nas próximas seções serão apresentadas as definições básicas de Aritmética Intervalar, métodos de Branch & Bound e técnicas de Consistência Local, que são utilizados na proposta desta tese.

### 2.1.1 Aritmética Intervalar

A Aritmética Intervalar (Moore, 1966) foi originalmente proposta como uma abordagem formal para analisar erros de arredondamento em sistemas computacionais. Um valor real $x$ representado de forma exata em aritmética de ponto-flutuante, pode também ser visto como um intervalo $[x, x]$. Por exemplo, a constante $\pi$ arredondada para três dígitos decimais é 3,141 ou 3,142, enquanto seu valor exato se encontra em algum lugar no intervalo $[3{,}141; 3{,}142]$.

Um intervalo $X$ é um conjunto contínuo de números reais. Por exemplo, $X = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \| \ \underline{x} \leq x \leq \overline{x}\}$ é um intervalo fechado em que $\underline{x}, \overline{x} \in \mathbb{R}$ são os *extremos* de $X$. Neste documento também é utilizada a notação $\underline{X}$ para o extremo inferior $\underline{x}$ e $\overline{X}$ para o extremo superior $\overline{x}$ do intervalo $X$. Um intervalo fechado é dito *degenerado* quando $\underline{x} = \overline{x}$. 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: $X = [\mu(X) - \omega(X)/2,\ \mu(X) + \omega(X)/2]$.

Dados os intervalos $X$, $Y$ e $Z$, a *extensão intervalar* de qualquer operação binária (unária) $\circ$ ($\diamond$) bem definida em $\mathbb{R}$ é definida por:

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

É possível computar $X \circ Y$ ou $\diamond Z$ para todas as funções algébricas e as principais transcendentais somente pela análise dos extremos de $X$ e $Y$ (Moore, 1966; Hansen e Walster, 2004b). Por exemplo, as operação de adição, subtração, multiplicação e divisão intervalares são dadas, respectivamente, por:

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

As operações de potenciação, radiciação e outras funções algébricas e trigonométricas podem ser definidas de maneira análoga, analisando os extremos dos intervalos (Kearfott, 2013; Hansen e Walster, 2004b; Jaulin et al., 2001).

Os conjuntos $X \circ Y$ e $\diamond Z$ podem ser intervalos, conjuntos vazios ou conjuntos não contínuos de números reais (*multi-intervalos*). Esse último caso pode ser observado na divisão por um intervalo que contém o valor zero. Por exemplo, $[1,2]/[-1,1] = (-\infty, -1] \cup [1, \infty)$. Dessa forma, um multi-intervalo $\mathcal{X}$ é um conjunto ordenado de intervalos disjuntos. Dados os multi-intervalos $\mathcal{X}$, $\mathcal{Y}$ e $\mathcal{Z}$, a operação $\mathcal{X} \circ \mathcal{Y}$ é o menor multi-intervalo que contém $\{X \circ Y \mid X \in \mathcal{X}, Y \in \mathcal{Y}\}$, e $\diamond\mathcal{Z}$ é o menor multi-intervalo que contém $\{\diamond Z \mid Z \in \mathcal{Z}\}$. A *envoltória* de um multi-intervalo $\mathcal{X}$ é o menor intervalo que contém $\bigcup_{X \in \mathcal{X}} X$.

A *extensão intervalar natural* de uma função é obtida pela substituição de cada variável por seu domínio e cada operação elementar por sua operação intervalar correspondente na expressão $f$. Por exemplo, na função $f(x) = x^2 + x + 1$, sua extensão intervalar natural é $\mathcal{F}(X) = X^2 + X + 1$.

### 2.1.2 Métodos de Branch & Bound Intervalar

Branch & Bound (Lawler e Wood, 1966) foi originalmente proposto como um método genérico para resolução de problemas combinatórios, ou seja, cujo domínio é um conjunto discreto. Por outro lado, métodos de Branch & Bound Intervalares (B&B) têm sido utilizados na abordagem de problemas contínuos de satisfação de restrições e problemas de otimização global com restrições (Araya e Reyes, 2016).

Durante a execução do método B&B, o espaço de busca é segmentado em *caixas*, aqui definidas como tuplas de (multi-)intervalos. Cada caixa é representada pela tupla $(\mathbf{X}, li)$, em que $\mathbf{X}$ corresponde ao domínio das variáveis e $li$ é o limite inferior da avaliação da função objetivo na caixa $\mathbf{X}$, ou seja, $li = \underline{F(\mathbf{X})}$. A melhor solução factível conhecida atual é representada por $\tilde{\mathbf{x}}$, enquanto $\tilde{f}$ é a avaliação dessa melhor solução.

O espaço de busca inicial é segmentado em duas caixas que compartilham uma face entre si. A função objetivo e as restrições da instância são avaliadas em cada caixa por meio de aritmética intervalar, definindo o valor $li$ de cada caixa. Caso seja detectado que determinada caixa não pode conter uma solução de melhor qualidade que a melhor solução conhecida $\tilde{\mathbf{x}}$, ela é descartada. Se a caixa não puder ser descartada, ela é bissectada e as caixas resultantes são

inseridas na fila de prioridade $Q$ e posteriormente processadas. Esse processo é repetidamente aplicado até que determinado critério de parada seja atingido. Um B&B genérico é descrito no Algoritmo 1.

**Algoritmo 1** Método de Branch & Bound Intervalar

**Entrada:** $\mathbf{X}_0$: caixa inicial, $F$ : função objetivo, $C$: conjunto de restrições
**Saída:** $\tilde{\mathbf{x}}$: melhor solução conhecida, $\tilde{f}$: avaliação da melhor solução conhecida
1: $(\tilde{\mathbf{x}}, \tilde{f}) \leftarrow (\emptyset, +\infty)$
2: $Q \leftarrow (\mathbf{X}_0)$
3: **enquanto** $Q \neq \emptyset$ **faça**
4: Extrai uma caixa $\mathbf{X}$ da fila $Q$
5: Aplica a etapa de poda em $\mathbf{X}$
6: Avalia a caixa $\mathbf{X}$
7: **se** $\mathbf{X}$ não pode ser descartada **então**
8: Atualiza a melhor solução conhecida $(\tilde{\mathbf{x}}, \tilde{f})$
9: Bissecta $\mathbf{X}$
10: Insere as sub-caixas resultantes $\mathbf{X}'$ e $\mathbf{X}''$ na fila $Q$
11: **fim se**
12: **fim enquanto**

Existem algumas escolhas, geralmente dependentes da instância que está sendo abordada, que são determinantes para a velocidade de convergência do B&B. Uma delas é a escolha da variável a ser particionada (linha 9 do Algoritmo 1). Em geral, opta-se por aquela cujo intervalo tenha o maior comprimento ou particiona-se uma após a outra de maneira circular. Outro ponto importante é a escolha da próxima caixa a ser extraída da fila $Q$ (linha 4). De acordo com Vanaret (2015), as estratégias mais utilizadas são:

- Melhor primeiro: a caixa $\mathbf{X}$ com o menor limite inferior na função objetivo $li = \underline{F(\mathbf{X})}$ é explorada para direcionar a busca para a região mais promissora;
- Maior primeiro: a maior caixa é extraída da fila a fim de priorizar a exploração de caixas mais antigas;
- Busca em profundidade: as caixas exploradas mais recentemente são priorizadas.

A cada caixa avaliada e não descartada, a melhor solução conhecida $\tilde{\mathbf{x}}$ pode ser atualizada (linha 8). Porém, essa atualização só acontece caso seja encontrada uma solução factível com valor de função objetivo (custo) melhor que a atual $\tilde{f}$. Com isso, pode-se obter uma redução do espaço de busca da instância sem perda da solução ótima. Araya e Reyes (2016) enumeram diversas técnicas utilizadas para atualização da melhor solução conhecida, tais como a avaliação do ponto médio da caixa (considerando o ponto médio de cada um dos intervalos que a compõem) e a aplicação dos métodos Simplex ou de Newton.

A etapa de poda descrita na linha 5 do Algoritmo 1 é executada por uma ou mais técnicas de Consistência Local, também chamadas *contratores*, que são responsáveis por filtrar o domínio da instância de forma a descartar valores sub-ótimos ou infactíveis sem perda de soluções. Se todos os valores de uma caixa são descartados nesse processo, uma caixa vazia é retornada. O processo de contração de domínio utilizado na proposta desta tese é detalhado na Seção 2.1.3.

A Figura 2.1 ilustra um exemplo de execução de B&B adaptado de Araya e Reyes (2016). A caixa 1 representa o domínio inicial da instância. A área dentro da curva na caixa representa o espaço de soluções factíveis. A estrela indica a localização da solução ótima, com custo de $-2{,}5$.

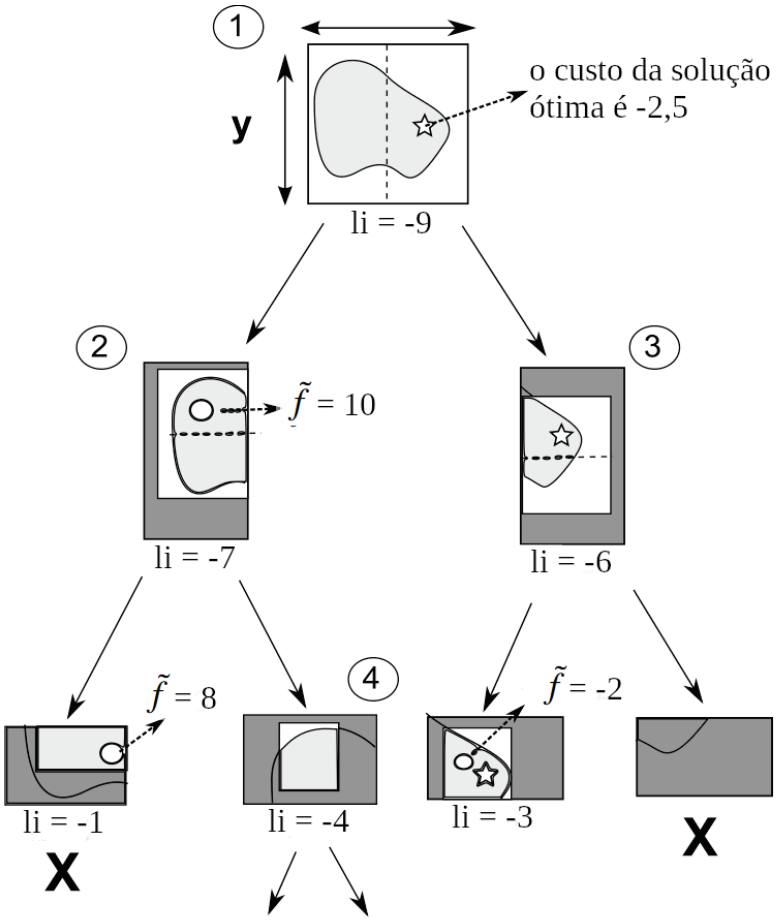



Figura 2.1: Exemplo de funcionamento de um método de Branch & Bound (Adaptado de Araya e Reyes (2016)).

Na primeira iteração, a caixa 1 é bissectada. As caixas resultantes (2 e 3) são contraídas por um método de consistência local e computam-se os novos limites inferiores da função objetivo para cada uma delas ($li = -7$ e $li = -6$, respectivamente). A partir disso, procuram-se novas soluções factíveis para atualização da melhor solução conhecida e $\tilde{f} = 10$ é encontrada, definindo um limite superior para a solução ótima. Ambas as caixas são adicionadas à fila $Q$. Na segunda iteração é selecionada a caixa que possui o menor limite inferior. A caixa 2 ($li = -7$) é então bissectada e o processo de poda é aplicado sobre as caixas resultantes. É importante notar que nesse passo podem ser descartadas regiões factíveis que sejam sub-ótimas em relação à melhor solução conhecida $\tilde{f}$. Um novo limite superior para a solução ótima é encontrado na caixa da esquerda, com $\tilde{f} = 8$. Na terceira iteração, a caixa 3 é selecionada por conter o menor limite inferior $li = -6$. Após as etapas de bissecção e poda, a caixa da direita é descartada e obtém-se $\tilde{f} = -2$ na caixa da esquerda. Com isso, a caixa com $li = -1$ pode ser descartada da fila porque o valor de seu limite inferior é maior que a melhor solução conhecida até o momento, o que garante de que a caixa não contém a solução ótima. A busca continua até que a fila $Q$ esteja vazia.

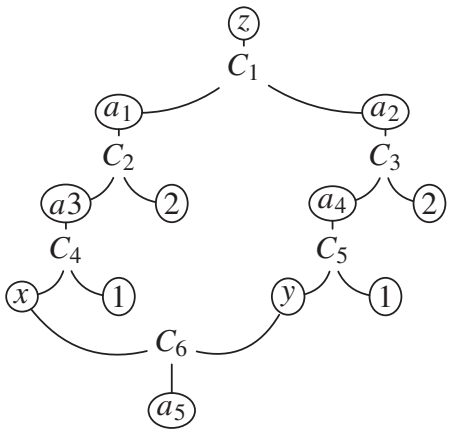
### 2.1.3 Consistência Local

Nesta tese são utilizados como sinônimos os termos consistência local, filtragem de domínio, contração de domínio e poda do espaço de busca. Aqui são abordadas instâncias de otimização global cujas funções objetivo e restrições podem ser reescritas como uma *rede de restrições* ternárias (Faltings e Gelle, 1997), ou seja, uma rede composta por restrições que envolvem no máximo três variáveis. Nesse contexto, as restrições possuem a forma $x = y \circ z$

ou $x = \diamond y$, na qual $\circ$ ($\diamond$) é um operador binário (unário) bem definido. 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 $x_1 \cdot z_1 = 0$. Outro exemplo é a instância descrita a seguir:

$$\begin{aligned} \min \quad & (x-1)^2 + (y-1)^2 \\ \text{sujeito a} \quad & x + y \leq 1, \\ & x \geq 0, \\ & y \geq -10, \\ & y \leq 10, \end{aligned}$$

que pode ser reescrita na forma da rede de restrições ternárias da Figura 2.2, em que se adiciona a variável $z$ para representar a função objetivo, e as variáveis auxiliares $\{a_1, \ldots, a_5\}$ com domínios $D = (-\infty, +\infty)$. O uso das variáveis $x$, $y$ e $a_5$ com domínios $D_x = [0, \infty)$, $D_y = [-10, 10]$ e $D_{a_5} = (-\infty, 1]$, respectivamente, permite a representação das inequações presentes nas restrições. A rede de restrições ternárias também é representada na figura como um hipergrafo em que os vértices são as variáveis e, para cada restrição $C$, existe uma hiperaresta conectando seus respectivos vértices.




$$\begin{aligned} & \min z \\ & \text{sujeito a} \\ & C_1 : (z = a_1 + a_2) \\ & C_2 : (a_1 = {a_3}^2) \\ & C_3 : (a_2 = {a_4}^2) \\ & C_4 : (a_3 = x - 1) \\ & C_5 : (a_4 = y - 1) \\ & C_6 : (a_5 = x + y) \end{aligned}$$

Figura 2.2: Codificação da rede de restrições ternárias de uma instância e seu hipergrafo correspondente.

Na área de programação por restrições, técnicas de consistência local (também chamadas técnicas de contração ou de filtragem de domínios) permitem a poda dos domínios das variáveis pela exclusão de valores que não respeitam a rede de restrições da instância. A mais importante dessas consistências, chamada Consistência de Arco Generalizada (do inglês, *Generalized Arc Consistency* ou GAC), garante que uma atribuição de valores para uma única variável pode ser estendida para outras variáveis para satisfazer uma restrição (Mackworth, 1977b).

No caso contínuo, uma restrição ternária $x = y \circ_1 z$ é GAC em relação a uma caixa $(X, Y, Z)$ se e somente se $X \subseteq Y \circ_1 Z$, $Y \subseteq X \circ_2 Z$ e $Z \subseteq X \circ_3 Y$, em que $\circ_2$ e $\circ_3$ são as operações inversas de $\circ_1$ que mantêm a condição $(x = y \circ_1 z) \iff (y = x \circ_2 z) \iff (z = x \circ_1 y)$. Por exemplo: $(x = y + z) \Leftrightarrow (y = x - z) \Leftrightarrow (z = x - y)$, em que $\circ_1 \equiv +$, $\circ_2 \equiv -$ e $\circ_3 \equiv -$.

GAC é alcançada em restrições ternárias por meio de:

$$\texttt{contrator\_GAC}(x_1 = x_2 \circ_1 x_3) := \begin{cases} X_1 \leftarrow X_1 \cap (X_2 \circ_1 X_3) \\ X_2 \leftarrow X_2 \cap (X_1 \circ_2 X_3) \\ X_3 \leftarrow X_3 \cap (X_1 \circ_3 X_2). \end{cases} \tag{2.1}$$

Nesse caso, a caixa $(X_1,X_2,X_3)$ contraída pode conter multi-intervalos. Consistências relaxadas, como Consistência de Envoltória (*Hull Consistency*) (Benhamou e Older, 1992) têm sido aplicadas para evitar o tratamento de multi-intervalos, uma vez que não há consenso na literatura sobre as vantagens de se manter uma representação multi-intervalar (Chabert et al., 2004; Faltings, 1998; Sidebottom e Havens, 1992).

Muitos algoritmos têm sido propostos para alcançar GAC em toda a rede de restrições da instância. O algoritmo AC3 (Mackworth, 1977a) consiste em iterativamente aplicar o `contrator_GAC` sobre as restrições de acordo com uma fila $K$. Inicialmente, todas as restrições são inseridas na fila. Enquanto $K$ não estiver vazia, uma restrição $C$ é extraída de $K$ e `contrator_GAC`$(C)$ é executado. Se o domínio de alguma variável $x$ é contraído, todas as outras restrições que contém $x$ são incluídas em $K$. Se for gerado domínio vazio, uma inconsistência é detectada e GAC retorna.

Em geral, esse processo só termina se um número máximo de passos for atingido ou se a rede de restrições tiver propriedades estruturais específicas (Cohen e Jeavons, 2017; Faltings, 1998). Uma rede de restrições ternárias é considerada cíclica se for possível percorrer seu hipergrafo de restrições de modo que haja uma sequência de variáveis em que a primeira e a última coincidam, mas nenhuma outra seja repetida. Se a rede de restrições não for cíclica, então GAC é alcançada por meio de uma ordenação $d = (C_1, C_2, \ldots, C_m)$ das restrições de modo que cada restrição $C_i$ possui somente uma variável em suas restrições predecessoras $C_{j<i}$. Com isso, basta aplicar `contrator_GAC` para frente e para trás sobre as restrições de $d$. Esse procedimento não garante que GAC seja alcançada em uma rede cíclica, mas reduz os domínios das variáveis de forma que cada restrição $C_i$ será GAC em relação a sua variável que pertence às restrições predecessoras na ordenação $d$. Neste caso, a rede de restrições é dita direcionalmente arco consistente (do inglês *directionally arc consistent* ou DAC) sobre $d$.

O processo de filtragem de domínio por GAC utilizado nas propostas desta tese utiliza informações estruturais da rede de restrições da instância para aplicar uma combinação de DAC e AC3. Derenievicz e Silva (2018) propuseram o conceito de *decomposição epífita* de uma rede de restrições como uma tupla $(\mathcal{A}, \Omega, t)$, em que $\mathcal{A}$ é um conjunto ordenado de redes acíclicas obtido pela remoção de um conjunto de restrições $\Omega$ da rede. Nesse caso, $\Omega$ e $t : \Omega \mapsto V_\Omega$ é uma função que associa cada restrição $C \in \Omega$ a uma de suas variáveis $t(C)$ satisfazendo: se $t(C)$ pertence à rede $A_i$ então as demais variáveis de $C$ pertencem à rede anterior $A_{j<i}$ e não há outra restrição $C' \in \Omega$ tal que $t(C')$ pertence a $A_i$.

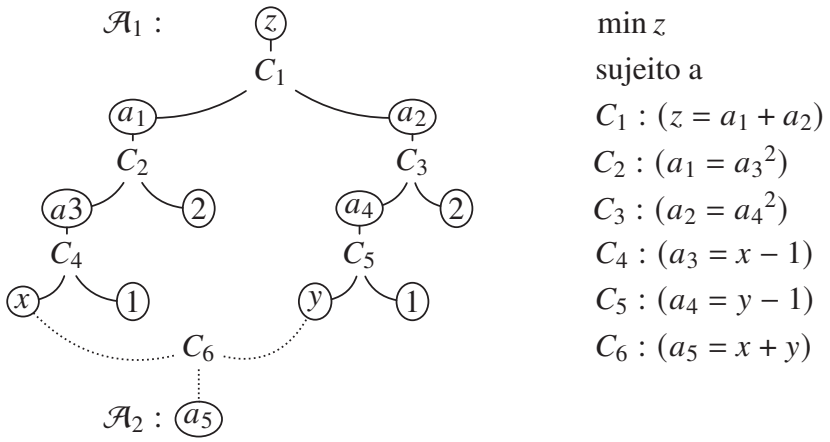



Figura 2.3: Uma decomposição epífita da rede de restrições ternárias ilustrada na Figura 2.2.

Uma rede de restrições pode ter muitas decomposições epífitas diferentes, mas é desejado que $|\Omega|$ seja mínimo (Derenievicz e Silva, 2018). Dessa forma, é utilizada uma abordagem

heurística que visa maximizar o tamanho da primeira rede acíclica da decomposição, já que o processo para alcançar GAC em redes acíclicas é mais simples. A Figura 2.3 mostra uma decomposição epífita do exemplo da Figura 2.2 em que $\Omega = \{C_6\}$.

**Algoritmo 2** Contração de domínio por GAC.

**Entrada:** decomposição epífita $(\mathcal{A}, \Omega, t)$ da rede de restrições ternárias, variável $z$ que representa a função objetivo, limite superior $ls$, caixa $B$ e número máximo de passos $MAX$.

**Saída:** A qualquer momento, se for obtido um intervalo vazio, retorna **conflito**. Caso contrário, $B$ é contraída.

1: $D_z \leftarrow D_z \cap (-\infty, ls)$ ▷ *contrai o domínio de z utilizando o limite superior ls de* $\tilde{f}$
2: **para** $A_i \in \mathcal{A}$, com $i$ de 1 a $|\mathcal{A}|$ **faça**
3: seja $(C_1, C_2, \ldots, C_m)$ uma ordenação das restrições de $A_i$
4: `contrator_GAC`$(C_j, B)$, para cada $j = m \ldots 1$
5: `contrator_GAC`$(C_j, B)$, para cada $j = 1 \ldots m$
6: **fim para**
7: Cria uma fila vazia $K$
8: **para todo** $C \in \Omega$ **faça** ▷ *processa o conjunto* $\Omega$ *da decomposição*
9: enfileira$(K,$ `contrator_GAC`$(C, B))$ ▷ *enfileira as variáveis que foram contraídas*
10: **fim para**
11: $s \leftarrow 0$
12: **enquanto** $K$ não estiver vazia **e** $s < MAX$ **faça** ▷ *algoritmo similar ao AC3*
13: $x \leftarrow$ desenfileira$(K)$
14: **para todo** restrição $C$ de $\mathcal{A}$ ou de $\Omega$ que contém a variável $x$ **faça**
15: enfileira$(K,$ `contrator_GAC`$(C, B))$
16: **fim para**
17: $s \leftarrow s + 1$
18: **fim enquanto**
19: **retorna** envoltória$(B)$

O Algoritmo 2 detalha a execução da contração de domínio por GAC. Inicialmente, o domínio da variável $z$, que representa a função objetivo, é contraído em relação ao limite superior $ls$ da melhor solução conhecida $\tilde{f}$. Em seguida, a poda por DAC é aplicada em cada rede acíclica identificada pela decomposição epífita (linhas 2-6). Se considerarmos o exemplo da Figura 2.3, existem duas redes acíclicas, $\mathcal{A}_1$ e $\mathcal{A}_2$. No entanto, a rede $\mathcal{A}_2$ é composta somente por uma variável $a_5$, o que dispensa seu processamento. Dessa forma, na linha 4 é aplicado o `contrator_GAC` sobre as restrições $\{C_5 \ldots C_1\}$ e na linha 5 o processo de contração é aplicado na ordem inversa $\{C_1 \ldots C_5\}$.

Para fins didáticos, o processo de contração descrito no Algoritmo 2 sobre o exemplo da Figura 2.3 é detalhado a seguir. Sabe-se que os domínios iniciais das variáveis $x$, $y$ e $a_5$ são $D_x = [0, \infty)$, $D_y = [-10, 10]$ e $D_{a_5} = (-\infty, 1]$, respectivamente, e que as variáveis auxiliares $\{a_1, \ldots, a_4\}$ possuem domínios $D = (-\infty, +\infty)$. Sendo assim, o processamento descrito na linha 4 do algoritmo se inicia pela restrição $C_5$ e suas operações inversas:

$$\texttt{contrator\_GAC}(C_5) := \begin{cases} D_{a_4} = D_y - 1 \rightarrow (-\infty,\infty) \cap ([-10{,}10] - [1{,}1]) \rightarrow D_{a_4} = [-11{,}9] \\ D_y = D_{a_4} + 1 \rightarrow [-10, 10] \cap ([-11, 9] + [1, 1]) \rightarrow D_y = [-10, 10] \\ [1, 1] = D_y - D_{a_4} \rightarrow [1, 1] \cap ([-10, 10] - [-11, 9]) \rightarrow [1, 1]. \end{cases}$$

$D_{a_4}$ foi contraído nesse processo, enquanto $D_y$ permanece o mesmo. Quanto ao intervalo [1,1], a única possibilidade de alteração seria a detecção de um intervalo vazio durante o processamento, o que faria o algoritmo retornar conflito e a caixa ser descartada. A seguir,

o `contrator_GAC` é aplicado sobre as restrições $\{C_4, \ldots, C_1\}$, o que resulta nos domínios a seguir:

$$
\begin{array}{lll}
D_z = [0, \infty) & D_{a_1} = [0, \infty) & D_{a_4} = [-11, 9] \\
D_x = [0, \infty) & D_{a_2} = [0, 121] & D_{a_5} = (-\infty, 1] \\
D_y = [-10, 10] & D_{a_3} = [-1, \infty) &
\end{array}
$$

é interessante notar que nessa primeira etapa as variáveis $z$, $x$ e $y$ não tiveram seus domínios contraídos, mas as demais reduziram consideravelmente. Seguindo o fluxo do algoritmo, na linha 5 o `contrator_GAC` deve ser aplicado sobre as restrições na ordem inversa, ou seja, $\{C_1 \ldots C_5\}$. Com a execução desse passo, nenhuma variável teve seu domínio contraído.

Exceto por erros numéricos decorrentes do uso de aritmética de ponto flutuante, GAC é eficientemente obtida em todas as sub-redes, uma vez que a representação multi-intervalar da caixa é mantida. No entanto, a rede como um todo pode não ser consistente devido ao fato de as restrições do conjunto $\Omega$ da decomposição ($C_6$ no exemplo da Figura 2.3) ainda não terem sido processadas. Portanto, nas linhas 8-10 o `contrator_GAC` é aplicado às restrições do conjunto $\Omega$, adicionando à fila $K$ as variáveis que foram contraídas e devem ser processadas pelo AC3 (linhas 12-18) para que essa contração seja propagada para o restante da rede.

Retornando ao exemplo da Figura 2.3, tem-se a aplicação do `contrator_`$GAC$ sobre a restrição $C_6$ que compõe o conjunto $\Omega$. Considerando os domínios atuais das variáveis, o processamento abaixo é executado e as variáveis $x$, $y$ e $a_5$ recém contraídas são adicionadas à fila $K$:

$$
\texttt{contrator\_GAC}(C_6) := \begin{cases}
D_{a_5} = D_x + D_y \rightarrow (-\infty,1] \cap ([0,\infty) + [-10,10]) \rightarrow D_{a_5} = [-10,1] \\
D_x = D_{a_5} - D_y \rightarrow [0,\infty) \cap ([-10,1] - [-10,10]) \rightarrow D_x = [0,11] \\
D_y = D_{a_5} - D_x \rightarrow [-10,10] \cap ([-10,1] - [0,11]) \rightarrow D_y = [-10,1].
\end{cases}
$$

A implementação do AC3 (linhas 11-18) utiliza um contador $s$ de variáveis processadas: a cada iteração, são exploradas todas as restrições que contêm a variável em questão. Se a execução da linha 15 implicar na contração dos domínios de outras variáveis, elas são adicionadas à fila. Por exemplo, na primeira iteração a variável $x$ é desenfileirada e a restrição $C_4$ processada, o que gera $D_{a_3} = D_x - 1 \rightarrow [-1,\infty) \cap ([0,11] - [1,1]) \rightarrow D_{a_3} = [-1,10]$. Com essa contração, $a_3$ é adicionada à fila $K$ e a restrição $C_2$ que a contém deverá ser posteriormente processada. No exemplo, a execução do algoritmo se encerra com $s = 4$ e os domínios resultantes são listados a seguir:

$$
\begin{array}{lll}
D_z = [0, 221] & D_{a_1} = [0, 100] & D_{a_4} = [-11, 0] \\
D_x = [0, 11] & D_{a_2} = [0, 121] & D_{a_5} = [-10, 1] \\
D_y = [-10, 1] & D_{a_3} = [-1, 10] &
\end{array}
$$

O número total de execuções de `contrator_GAC` depende da estrutura da rede de restrições, uma vez que redes densas em geral podem ter mais restrições por variável. É importante notar que o Algoritmo 2 tem como entrada uma caixa intervalar, mas durante o processo de contração podem ser gerados multi-intervalos. Na implementação utilizada nesta tese, a representação multi-intervalar é mantida ao longo de todo o processo, mas o retorno é dado pela menor caixa que contém todos os multi-intervalos gerados (envoltória ($B$)).

Na próxima seção serão apresentados os principais conceitos da área de otimização global por meta-heurísticas. Em especial, serão exploradas as características de Evolução

Diferencial (DE), a meta-heurística utilizada na proposta desta tese, e de técnicas de tratamento de restrições da literatura recente.

## 2.2 MÉTODOS META-HEURÍSTICOS DE OTIMIZAÇÃO

No contexto de otimização, a palavra "heurística" é denotada como um método de resolução de problemas que fornece uma solução aproximada, muitas vezes em tempo razoável, mas não garante a otimalidade dessa solução. Em termos práticos, heurísticas se baseiam em experiência ou conhecimento específico do domínio para resolver problemas que são intratáveis para métodos exatos devido à sua complexidade computacional (Yang et al., 2015).

O conceito de meta-heurística abrange métodos que facilitam interações entre procedimentos de busca local e estratégias de alto nível com o objetivo de criar abordagens que sejam capazes de evitar ótimos locais e realizar buscas mais robustas. Além disso, em geral não utilizam conhecimento específico do domínio, apresentando as seguintes características: são inspiradas em princípios encontrados na natureza, como física, biologia ou etologia; incorporam componentes estocásticos, envolvendo variáveis aleatórias; não dependem de conhecimento sobre as propriedades da função objetivo ou das restrições da instância em questão; e incluem diversos parâmetros que podem ser ajustados (Boussaïd et al., 2013).

Além disso, Bilal et al. (2020) comenta que meta-heurísticas são comumente subdivididas em duas categorias:

- Algoritmos baseados em vizinhança: agem como uma busca local na qual a cada iteração a solução se move em direção a uma região vizinha a fim de melhorar o valor atual da função objetivo. Dois desses algoritmos são Têmpera Simulada (*Simulated Annealing*) (1979) e Busca Tabu (*Tabu Search*) (1989);
- Algoritmos baseados em população: operam sobre um conjunto de soluções candidatas (ou uma população). Nessa categoria existem ainda duas divisões:
  - Inteligência de Enxames: algoritmos baseados no comportamento sócio-cooperativo exibido por várias espécias da natureza geralmente na busca por alimentos. Alguns dos algoritmos mais populares dessa classe são Otimização por Colônia de Formigas (*Ant Colony Optimization* – ACO) (2006), Otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO) (1995) e Otimização por Colônias Artificiais de Abelhas (*Artificial Bee Colony Optimization* – ABC) (2007);
  - Algoritmos Evolutivos: algoritmos inspirados na teoria da evolução das espécies, tais como Algoritmos Genéticos (*Genetic Algorithms* – GA) (1957), Estratégias Evolutivas (*Evolutionary Strategies*) – ES) (1964) e Evolução Diferencial (*Differential Evolution* – DE) (1995).

Uma característica bastante comum em meta-heurísticas baseadas em população é a busca pelo equilíbrio entre diversidade (exploração global) e busca local ou intensificação (exploração local). A diversidade refere-se à dispersão das soluções no espaço de busca, o que possibilita a identificação de regiões factíveis (em que nenhuma restrição é violada) e evita que a população fique estagnada em mínimos locais. Por outro lado, a busca local é responsável por explorar a vizinhança das soluções atuais a fim de melhorá-las. No entanto, ajustar adequadamente os parâmetros que definem a forma que a meta-heurística explora o espaço de busca não é uma tarefa trivial. Quando as soluções se concentram em um ponto sub-ótimo precocemente e têm dificuldades para se deslocar para outras regiões, diz-se que houve convergência prematura.

De acordo com Bandaru e Deb (2016), meta-heurísticas podem levar a soluções boas o suficiente para problemas NP-difíceis, ou seja, problemas para os quais não existe um algoritmo exato conhecido que possa resolvê-los em um período de tempo razoável. Além disso, ao contrário da maioria dos métodos clássicos, meta-heurísticas não requerem informações de gradiente e, portanto, podem ser utilizadas em funções objetivo de caixa-fechada ou baseadas em simulação. Por fim, a maioria das meta-heurísticas consegue escapar de regiões sub-ótimas devido à sua natureza estocástica ou a heurísticas desenvolvidas especificamente para esse propósito.

A seguir será brevemente apresentada a versão clássica do método de Evolução Diferencial (DE), uma das meta-heurísticas que mais têm se destacado em pesquisas recentes da área de otimização numérica.

### 2.2.1 Evolução Diferencial

Storn e Price (1995) propuseram o método de Evolução Diferencial (do inglês *Differential Evolution* – DE) a fim de abordar funções não-lineares, não-convexas, multimodais e não diferenciáveis no contexto de otimização numérica em espaços de busca contínuos. A principal ideia do DE é combinar soluções atuais de acordo com determinada probabilidade, a fim de gerar novas soluções candidatas.

Assim como na maior parte das meta-heurísticas evolutivas, o método DE clássico consiste em um laço de $G$ *gerações* (iterações) sobre uma população de soluções candidatas, ditas 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, $0 \leq g < G$. Ao longo do processo evolutivo, a qualidade de um indivíduo $\mathtt{x}$, chamada *fitness*, geralmente é calculada por meio de uma composição entre o valor da função objetivo e das violações das restrições da instância. Quando um indivíduo não viola nenhuma restrição, diz-se que ele é *factível*.

Na versão clássica, a população inicial é formada por um conjunto aleatório uniformemente distribuído de soluções candidatas amostradas no espaço de busca da instância. Durante cada geração, os operadores de mutação, cruzamento e seleção são aplicados à população até que determinado critério de parada seja satisfeito, como um número máximo de gerações ou de avaliações de *fitness*.

Muitas estratégias de mutação têm sido propostas no contexto de DE (Chakraborty, 2008; Bilal et al., 2020; Malik et al., 2021). A fim de distinguir cada uma delas, a notação "DE / a / b" é utilizada, sendo que "a" especifica o indivíduo sobre o qual será aplicada a operação de mutação e "b" é o número de indivíduos diferenciais envolvidos na operação. Em alguns casos, é adicionado um termo "c" a essa nomenclatura para definir o tipo de esquema de cruzamento utilizado.

Na etapa de mutação, uma operação específica é aplicada a fim de gerar um *indivíduo mutante* $\mathtt{v}_i$. A estratégia de mutação mais conhecida é DE/rand/1, ilustrada na Figura 2.4 e descrita a seguir:

$$\mathtt{v}_i = \mathtt{r}_1 + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{2.2}$$

em que $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ são três indivíduos diferentes de $\mathtt{x}_i$ (*indivíduo alvo*) e mutuamente distintos entre si selecionados da população atual $\mathtt{p}_g$. Indivíduos aleatórios são novamente escolhidos para cada um dos indivíduos mutantes a serem criados. Os indivíduos $\mathtt{r}_1$ e $\mathtt{r}_2$ são chamados *base* e *terminal*, respectivamente. A diferença $(\mathtt{r}_2 - \mathtt{r}_3)$ é escalada pelo fator $F$, um parâmetro de controle que tipicamente se encontra no intervalo $[0, 1]$. É importante notar que um fator

escalar pequeno promove a intensificação da busca local em torno do indivíduo $\mathtt{r}_1$, enquanto que um valor alto promove a exploração de outras regiões do espaço de busca.

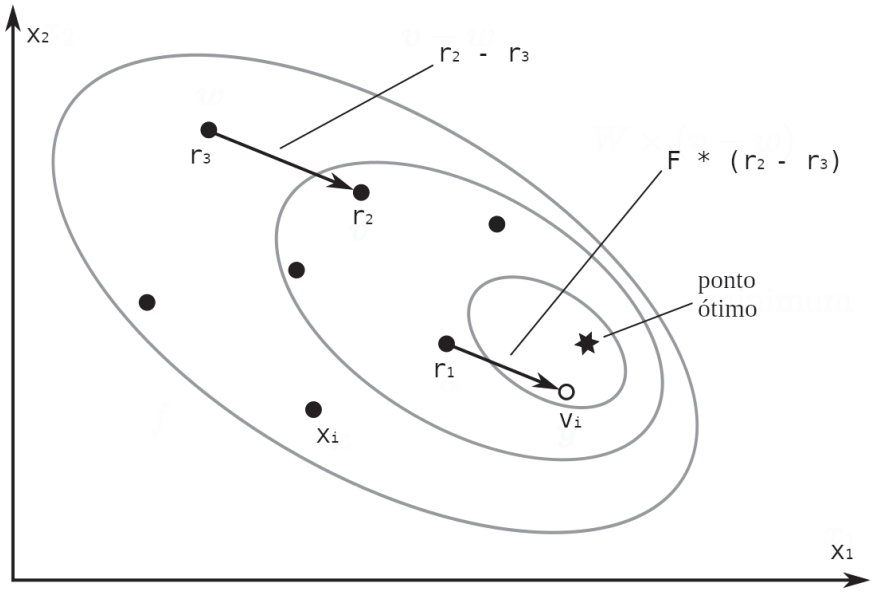



Figura 2.4: Exemplo de operação de mutação pela estratégia DE/rand/1 sobre uma instância de duas dimensões.

Outros operadores de mutação utilizados na literatura recente são DE/current-to-rand/1 (Iorio e Li, 2004) (Equação 2.3) e DE/current-to-*p*best/1 (Zhang e Sanderson, 2009) (Equação 2.4).

$$\mathtt{v}_i = \mathtt{x}_i + s \cdot (\mathtt{r}_1 - \mathtt{x}_i) + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{2.3}$$

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

em que $s$ é um valor aleatório entre 0 e 1, e $\mathtt{r}_p$ é um indivíduo aleatoriamente selecionado entre as 25% melhores soluções da população atual.

A fim de aumentar a diversidade da população enquanto se mantém parte das informações dos indivíduos atuais, a operação de cruzamento é aplicada a cada par de indivíduo alvo $\mathtt{x}_i$ e seu indivíduo mutante correspondente $\mathtt{v}_i$ para gerar o *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.

As duas estratégias de cruzamento mais comuns em DE são binomial e exponencial. O *cruzamento binomial* é descrito a seguir:

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

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

No *cruzamento exponencial*, primeiro escolhe-se um inteiro $k \in [1, D]$ como índice inicial da operação. Outro valor inteiro $l \in [1, D]$ é determinado com probabilidade $\mathsf{P}(l \geq L) =$

$CR^{L-1}$ para qualquer $L > 0$ e denota quantas variáveis consecutivas são selecionadas do vetor mutante a partir da posição $k$. Então, o indivíduo experimental é gerado como a seguir:

$$\mathtt{u}_{ij} = \begin{cases} \mathtt{v}_{ij}, & \text{se } j \in \{m_D(k), m_D(k+1), \ldots, m_D(k+l-1)\}, \\ \mathtt{x}_{ij}, & \text{caso contrário,} \end{cases} \tag{2.6}$$

em que $m_D(n) = 1 + ((n-1) \mod D)$ permite iterar ciclicamente pelo vetor.

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 aleatoriamente seleciona uma variável para ser herdada do indivíduo mutante. Conforme será detalhado na Seção 3.1, nos últimos anos têm sido propostas diversas estratégias de mutação que visam melhor se adaptar às particularidades do espaço de busca de cada problema (Zhang et al., 2023). Da mesma forma, foram desenvolvidos diversos esquemas que visam tornar os parâmetros *F* e *CR* adaptativos de acordo com a região do espaço de busca que está sendo percorrida e com o estágio do processo de otimização.

Tendo em vista que as estratégias de mutação mais conhecidas não levam em consideração o domínio $X$ de valorações possíveis para as variáveis, o indivíduo mutante pode conter alguma valoração que não está contida no domínio da variável. Nesse caso, existem algumas alternativas para gerar uma nova valoração para a variável $x_j$ dentro dos limites do domínio $X_j$, $1 \leq j \leq D$: reposicionar a variável no limite mais próximo, escolher um valor aleatório dentro do domínio, ou selecionar um valor entre a valoração do indivíduo base $\mathtt{r}_1 j$ e o limite inferior $\underline{X}_j$ ou superior $\overline{X}_j$ do domínio $X_j$:

$$x_j = \begin{cases} \mathtt{r}_{1j} + s \cdot (\overline{X}_j - \mathtt{r}_{1j}) & \text{se } x_j > \overline{X}_j \\ \mathtt{r}_{1j} + s \cdot (\underline{X}_j - \mathtt{r}_{1j}) & \text{se } x_j < \underline{X}_j, \end{cases} \tag{2.7}$$

em que $s \in [0, 1]$ é um valor aleatório.

Com o intuito de manter o tamanho da população constante ao longo das gerações, é aplicada a operação de *seleção* para determinar quais indivíduos sobrevivem para a próxima geração. Para tanto, o indivíduo experimental $\mathtt{u}_i$ é comparado ao seu indivíduo alvo correspondente $\mathtt{x}_i$ de acordo com determinado critério que, em geral, é dado pela combinação entre os valores de custo e de violação de restrições. O melhor dos indivíduos passa para a próxima geração.

Um dos critérios de parada mais comuns em DE se refere aos recursos disponíveis, tais como tempo de processamento, número de iterações ou número de indivíduos avaliados. Além disso, é possível considerar a qualidade da melhor solução conhecida (proximidade do valor ótimo previamente informado) ou a estagnação do processo de busca (um número de iterações consecutivas em que não se obteve melhorias significativas).

Além da violação dos limites do domínio da instância, é possível que um indivíduo viole restrições de igualdade ou desigualdade. Na Seção 2.2.2 são comentadas algumas técnicas utilizadas na literatura recente para tratamento de violação de restrições em otimização numérica.

### 2.2.2 Técnicas de Tratamento de Restrições

Apesar de serem alternativas interessantes para resolução de instâncias que não podem ser atacadas por métodos exatos de forma eficiente, a maioria das meta-heurísticas foram originalmente desenvolvidas para problemas de otimização sem restrições de igualdade ou

desigualdade. Assim, diversas técnicas de tratamento de restrições têm sido desenvolvidas com o intuito de guiar o processo de busca em direção a regiões factíveis do espaço de busca.

Uma das técnicas mais utilizadas para tratamento de restrições é a aplicação de funções de penalização (Coello, 2002). Esse tipo de abordagem transforma uma instância de um problema de otimização com restrições em uma versão sem restrições de maneira que, em instâncias de minimização, é adicionado um determinado valor à função objetivo com base na violação de restrições do indivíduo, o que o desfavorece na operação de seleção.

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 e acabam por dificultar sua aplicação (Michalewicz, 1995). 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 brevemente apresentados alguns tipos de funções de penalização.

A penalidade de morte consiste na rejeição de indivíduos que violem alguma restrição, de forma a gerar novos indivíduos até que seja encontrado algum factível para substituí-lo. 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, como o próprio nome sugere, encontram-se as funções de penalização que permanecem constantes ao longo de todo o processo de busca. 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).

Penalização dinâmica é definida como qualquer função em que o estágio do processo de busca (geração atual) é considerado no cálculo da penalidade aplicada a um indivíduo (Mezura-Montes, 2009). Penalidades pequenas são utilizadas no início do processo de otimização a fim de estimular a exploração global por regiões factíveis. Penalidades maiores são empregadas conforme o processo de busca avança a fim de preservar os indivíduos factíveis e promover a exploração de suas vizinhanças em busca do ponto ótimo global.

Por fim, existem as penalidades adaptativas que utilizam informações sobre a qualidade das soluções atuais. Na abordagem mais utilizada, o fator de penalização em uma determinada geração é reduzido se os melhores indivíduos das últimas $k$ gerações forem factíveis ou aumentado se forem infactíveis. Se houver indivíduos factíveis e infactíveis dentre os 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 (Coello, 2022).

Outra forma de abordar instâncias de otimização numérica com restrições é considerar a função objetivo e as restrições separadamente. Para tanto, é estabelecida uma ordem lexicográfica em que a violação de restrições precede a função objetivo. O modelo mais simples é de Superioridade de Soluções Factíveis (*Superiority of Feasible Points* - SOF), proposto por Deb (2000), que utiliza um operador de seleção que compara dois indivíduos entre si de acordo com os seguintes critérios:

1. Um indivíduo factível é sempre melhor que um indivíduo infactível;
2. Entre dois indivíduos factíveis, o que possuir o melhor valor de função objetivo é escolhido;
3. Entre dois indivíduos infactíveis, o que possuir menor violação de restrições é escolhido.

Comparações realizadas entre pares de indivíduos visam priorizar regiões factíveis do espaço de busca. A maior desvantagem dessa abordagem é que não existe nenhum tipo de tolerância à violação de restrições, o que pode fazer com que soluções promissoras sejam descartadas e o processo de convergência desacelere, além de aumentar a probabilidade de estagnação em um ponto mínimo local.

Nesse contexto, Takahama e Sakai (2010) propuseram o método *ε-constrained*, no qual é considerada uma tolerância $\varepsilon$ nas comparações entre soluções candidatas. Sejam $f(\mathrm{x})$ e $\phi(\mathrm{x})$ os valores de função objetivo e violação de restrições do indivíduo $\mathrm{x}$, respectivamente, para qualquer $\varepsilon$ que satisfaça $\varepsilon \geq 0$, as comparações $<_\varepsilon$ e $\leq_\varepsilon$ entre dois indivíduos $\mathrm{x}_1$ e $\mathrm{x}_2$ são dadas por:

$$(f(\mathrm{x}_1), \phi(\mathrm{x}_1)) \leq_\varepsilon (f(\mathrm{x}_2), \phi(\mathrm{x}_2)) \Leftrightarrow \begin{cases} f(\mathrm{x}_1) \leq f(\mathrm{x}_2) & \text{se } \phi(\mathrm{x}_1), \phi(\mathrm{x}_2) \leq \varepsilon \\ f(\mathrm{x}_1) \leq f(\mathrm{x}_2) & \text{se } \phi(\mathrm{x}_1) = \phi(\mathrm{x}_2) \\ \phi(\mathrm{x}_1) \leq \phi(\mathrm{x}_2) & \text{caso contrário.} \end{cases} \tag{2.8}$$

Em casos em que $\varepsilon = \infty$, as comparações $<_\infty$ e $\leq_\infty$ pelo valor $\varepsilon$ 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 $f(\mathrm{x})$. Normalmente, instâncias simples podem ser resolvidas por meio de uma ordem lexicográfica com $\varepsilon = 0$. No entanto, em instâncias mais complexas o valor de $\varepsilon$ deve ser controlado a fim de buscar o equilíbrio entre exploração global e local do espaço de soluções.

Assim, Takahama e Sakai (2010) apresentaram um esquema em que $\varepsilon$ é atualizado até o número de gerações $g$ atingir uma determinada geração de controle $T$. A partir desse ponto, o valor de $\varepsilon$ é definido como 0 a fim de priorizar soluções com a menor violação de restrições possível. O valor de $\varepsilon$ é decrementado conforme o número de gerações aumenta, como descrito a seguir:

$$\varepsilon = \begin{cases} \varepsilon_0 \cdot (1 - \frac{g}{T})^{cp} & \text{se } 0 < g < T \\ 0 & \text{se } g \geq T, \end{cases} \tag{2.9}$$

em que $\varepsilon_0$ é a violação de restrições do melhor $\theta$-ésimo indivíduo da população inicial, $\theta$ é um índice em $1 \leq \theta \leq NP$ definido por parâmetro e $cp$ 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, os trabalhos relacionados elencados na Seção 3.1 revelam que grande parte das 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. Foram exploradas noções de aritmética intervalar, métodos de Branch & Bound Intervalar e técnicas de consistência local, utilizadas para descarte de regiões sub-ótimas e infactíveis do espaço de busca. Além disso, foram observadas algumas características de otimizadores baseados em meta-heurísticas, o funcionamento do algoritmo de Evolução Diferencial (DE) clássico e algumas das técnicas de tratamento de restrições mais comuns.

No próximo capítulo é realizada uma revisão bibliográfica sobre algumas abordagens adaptativas recentes no contexto de DE, além dos algoritmos participantes das últimas edições das competições do *IEEE Congress on Evolutionary Computation (CEC)* em otimização numérica com restrições. Ademais, são analisados alguns otimizadores híbridos que reúnem características de métodos exatos e meta-heurísticas. Com isso, espera-se localizar o presente trabalho de pesquisa em relação aos métodos já existentes na literatura.

# 3 TRABALHOS RELACIONADOS

Meta-heurísticas são amplamente utilizadas para resolver problemas de otimização onde é necessário encontrar soluções de alta qualidade em um curto período de tempo. No entanto, devido à sua natureza estocástica, essas técnicas dependem fortemente do ajuste preciso dos parâmetros de execução para atingir soluções próximas ao ótimo (Bandaru e Deb, 2016). Ajustar esses parâmetros manualmente pode ser desafiador, o que motivou o desenvolvimento de diversas abordagens adaptativas. Quando combinadas com técnicas de tratamento de restrições, essas abordagens têm se mostrado promissoras na resolução de problemas de otimização numérica com restrições.

Por outro lado, nas últimas décadas, diversos otimizadores têm se beneficiado da união entre métodos exatos e meta-heurísticas. A principal motivação para essas pesquisas é aproveitar as características complementares de diferentes estratégias de otimização e explorar a sinergia entre elas. Essa integração permite o desenvolvimento de resolvedores híbridos capazes de encontrar boas soluções para problemas que são difíceis de serem resolvidos por métodos exatos ou meta-heurísticas de forma isolada (Raidl et al., 2019).

Na Seção 3.1, são discutidas algumas abordagens adaptativas utilizadas no contexto de meta-heurísticas e os algoritmos das competições recentes do *IEEE Congress on Evolutionary Computation* (CEC) em otimização numérica com restrições. Na Seção 3.2, são explorados diferentes tipos de hibridizações entre métodos exatos e meta-heurísticas presentes na literatura.

## 3.1 OTIMIZADORES META-HEURÍSTICOS

O uso de meta-heurísticas é uma alternativa interessante para encontrar boas soluções para instâncias complexas que não podem ser eficientemente resolvidas por métodos exatos. Em meta-heurísticas evolutivas, um indivíduo representa uma solução candidata, ou seja, um ponto no espaço de busca da instância. Um dos métodos meta-heurísticos amplamente utilizados na literatura recente para abordar instâncias de otimização em espaços de busca contínuos é a Evolução Diferencial (*Differential Evolution* – DE) (Seção 2.2.1). No entanto, o sucesso do DE na resolução de uma instância depende fortemente do ajuste apropriado de seus parâmetros, como as estratégias de mutação e cruzamento, os valores dos parâmetros $F$ e $CR$, e o tamanho da população $NP$ (Viktorin et al., 2020).

Especificamente, (Zhang et al., 2023) afirma que algumas estratégias para a construção de novos indivíduos são mais exploratórias do que outras. A definição de um valor alto para $F$ pode resultar em uma melhor distribuição dos indivíduos no espaço de busca e aumentar a diversidade populacional, enquanto valores baixos de $F$ incentivam a busca local em torno dos indivíduos atuais e podem acelerar a convergência para um mínimo local. Da mesma forma, valores altos de $CR$ podem tornar os novos indivíduos significativamente diferentes dos atuais, o que aumenta a diversidade, mas pode desacelerar a convergência. Esse equilíbrio entre a ampla exploração do espaço de busca e a aceleração da convergência também é observado na definição do tamanho $NP$ da população.

Empregar um esquema de tentativa e erro para encontrar as configurações de parâmetros mais adequadas para cada instância é uma tarefa exaustiva, demandando muito tempo e recursos computacionais. Além disso, essa escolha pode variar de acordo com a região do espaço de busca explorada por cada indivíduo e com a dispersão da população. Para mitigar esse problema, diversos esquemas adaptativos têm sido propostos com o intuito de selecionar automaticamente

as configurações de parâmetros a partir da análise do processo de busca, tornando o DE mais robusto.

A associação de mecanismos de ajuste automático de parâmetros com técnicas de tratamento de restrições tem permitido que variantes do DE obtenham desempenho promissor ao abordar instâncias de problemas de otimização numérica com restrições. Algumas abordagens adaptativas encontradas na literatura recente são brevemente comentadas a seguir. Em seguida, são discutidas as características de alguns resolvedores participantes de competições recentes na área de otimização numérica por meta-heurísticas.

### 3.1.1 Abordagens Adaptativas

Tanabe e Fukunaga (2013) propuseram uma versão adaptativa de DE denominada *Success-History based Adaptive Differential Evolution* (SHADE). Essa estratégia tornou-se muito popular nos últimos anos e serviu como base para diversos resolvedores meta-heurísticos estado-da-arte. Essa notoriedade deve-se, principalmente, à utilização de um histórico de memórias com configurações bem-sucedidas dos parâmetros $F$ e $CR$ para guiar a escolha dos valores desses parâmetros nas gerações futuras.

O SHADE armazena as configurações bem-sucedidas dos parâmetros $F$ e$CR$ em vetores de tamanho $H = 5$, que são inicialmente definidos com o valor 0,5. Durante a execução do algoritmo, os valores de $F$ e $CR$ que resultaram em indivíduos experimentais melhores que os correspondentes na população atual são armazenados em vetores auxiliares. No final de cada geração, as memórias são atualizadas com base na média ponderada dos valores armazenados nos vetores auxiliares. Se nenhum valor de $F$ e $CR$ gerou um indivíduo melhor, as memórias não são atualizadas. A posição das memórias a ser atualizada é determinada de forma circular, garantindo que todas as posições sejam periodicamente atualizadas. Essa abordagem permite que o método SHADE adapte dinamicamente os valores dos parâmetros $F$ e $CR$, com base no histórico de sucesso das gerações anteriores, melhorando assim a eficiência do processo de busca.

Além disso, o método SHADE emprega uma adaptação da estratégia de mutação DE/current-to-$p$best/1 com arquivo, originalmente proposta por Zhang e Sanderson (2009). O arquivo opcional dessa estratégia refere-se a uma memória que contém indivíduos recentemente explorados que não obtiveram sucesso na operação de seleção. O objetivo de manter essas soluções é o reaproveitamento de informações potencialmente úteis e o aumento da diversidade populacional. A Equação 3.1 mostra a versão sem arquivo dessa estratégia:

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

em que $1 \leq i \leq D$, $D$ é o número de dimensões da instância, $\mathtt{r}_{p\text{best}}$ é aleatoriamente escolhido entre os $p$ melhores indivíduos da população atual, tal que $p \in (0, 1]$, $F$ é o fator escalar associado ao indivíduo $\mathtt{x}_i$, e $\mathtt{r}_1$ e $\mathtt{r}2$ são indivíduos aleatórios distintos entre si e diferentes de $\mathtt{r}p\text{best}$. A estratégia com arquivo difere apenas na seleção de $\mathtt{r}_2$, que é aleatoriamente escolhido entre a população atual e o arquivo. Quando o tamanho do arquivo excede um limite predefinido, indivíduos são aleatoriamente removidos até que esse limite seja atingido.

Sabe-se que o tamanho da população desempenha um papel significativo no controle do processo de convergência em meta-heurísticas. Populações pequenas tendem a convergir mais rapidamente, mas aumentam o risco de estagnação em um ponto mínimo local. Por outro lado, populações muito grandes tendem a explorar mais amplamente o espaço de busca, mas a convergência pode ser lenta. Buscando o equilíbrio entre exploração e intensificação, Tanabe e Fukunaga (2014) propuseram uma versão estendida do método SHADE, denominada L-SHADE, que utiliza a redução linear do tamanho da população durante o processo evolutivo. A cada

atualização, os piores indivíduos são removidos até que a população atinja um tamanho de $NP = 4$, que é o tamanho mínimo necessário para a aplicação da Equação 3.1. Seu bom desempenho, fácil implementação e a ausência de novos parâmetros a serem ajustados tornaram o L-SHADE uma estratégia muito popular, servindo como base para diversos resolvedores estado-da-arte elencados nas Seções 3.1.2 e 3.1.3.

Posteriormente, Brest et al. (2016) apresentaram uma versão melhorada do L-SHADE chamada iL-SHADE, que introduziu mecanismos para estimular a seleção de valores mais altos para $CR$. No ano seguinte, Brest et al. (2017) propuseram o jSO, que aprimora esse trabalho com algumas alterações nas configurações de parâmetros e introduziram um novo operador de mutação baseado na Equação 3.1. Esse operador utiliza um fator escalar ponderado ($F$), que dá menos ênfase à distância entre os indivíduos nos estágios iniciais do processo evolutivo para estimular a diversidade populacional. Nos estágios finais, são utilizados valores mais altos para priorizar a busca local em torno de soluções promissoras.

Organizadas durante o IEEE *Congress on Evolutionary Computation* (CEC), as competições *Special Session & Competitions on Real-Parameter Single Objective Optimization* introduzem ambientes experimentais específicos para avaliação e comparação de algoritmos de busca estocástica de última geração. Por conta disso, 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, suas heurísticas de busca e de adaptação de parâmetros são bons pontos de referência sobre o comportamento de diferentes estratégias nesse cenário de otimização.

Zhang et al. (2023) e Viktorin et al. (2020) destacaram diversas competições do CEC em que variantes do método SHADE (Tanabe e Fukunaga, 2013) participaram. Em 2013, o SHADE foi introduzido e avaliado no *benchmark* do CEC2013 em otimização numérica de parâmetros reais com objetivo único (Liang et al., 2013a), obtendo o 4º lugar como o melhor algoritmo baseado em DE. No ano seguinte, o L-SHADE (Tanabe e Fukunaga, 2014) venceu a competição do CEC2014 (Liang et al., 2013b). A competição do CEC2015 de cenários voltados para aprendizagem (Liang et al., 2014) foi vencida por uma versão do L-SHADE incorporada a um cruzamento baseado em autovetor e a um *framework* de seleção de pais bem-sucedidos (SPS-L-SHADE-EIG) (Guo et al., 2015). Em 2017, algoritmos baseados em L-SHADE conquistaram o 2º (Brest et al., 2017), 3º (Awad et al., 2017) e 4º (Mohamed et al., 2017) lugares na competição *bound constrained* (Awad et al., 2016). Similarmente, o 2º (Stanovov et al., 2018) e o 3º lugar (ELSHADE SPACMA) na competição do CEC2018 (Awad et al., 2016) foram obtidos por algoritmos inspirados no L-SHADE. Na competição do CEC2018 de otimização de parâmetros reais com restrições (Wu et al., 2017), cinco dos sete competidores utilizaram heurísticas propostas no L-SHADE. Esse padrão se repetiu no CEC2020 de otimização de problemas com restrições do mundo real com objetivo único (Kumar et al., 2020d), em que cinco dos oito competidores se inspiraram nas heurísticas do L-SHADE. Mais recentemente, as competições do CEC2021 e do CEC2022 foram vencidas por duas variantes de DE: APGSK-IMODE (Mohamed et al., 2021) e EA4eig (Bujok e Kolenovsky, 2022), respectivamente.

Para fornecer os subsídios necessários à compreensão de algumas das principais heurísticas da literatura recente de otimização numérica com restrições, as próximas seções apresentam os competidores do CEC2018 e do CEC2020. Vale notar que nas edições mais recentes do CEC não foram realizadas competições específicas dessa área.

### 3.1.2 Competição do CEC2018

A competição de otimização numérica com restrições do CEC2018 avaliou os participantes por meio de um conjunto de 28 funções (com o número de dimensões $D = \{10, 30, 50, 100\}$) pertencentes ao *benchmark* proposto por Wu et al. (2017). É importante notar que essas funções utilizam conceitos de otimização caixa-fechada, ou seja, os detalhes da estrutura analítica das instâncias são desconhecidos. Nesse caso, o algoritmo envia uma solução candidata para o código-fonte da função disponibilizado pela organização do evento, que retorna o valor da função objetivo e os valores das violações das restrições da instância. Cada instância do *benchmark* tem seu espaço de busca parametrizado por uma matriz de rotação e um vetor de deslocamento, gerados aleatoriamente e divulgados somente durante a competição.

Os competidores devem realizar 25 execuções de cada instância, mantendo as mesmas configurações de parâmetros. 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 competidores em 2018 foi:

**1º** IUDE (2018);

**2º** $\varepsilon$MAg-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 (2017) - 7º lugar

O competidor *Constraint Handling with Success History Adaptive Differential Evolution* (CAL-SHADE), proposto por Zamuda (2017), é uma versão do L-SHADE (Seção 3.1.1) adaptada para instâncias de otimização com restrições por meio do método $\varepsilon$*-constrained* (Seção 2.2.2). Nesse método, os autores utilizam o valor médio de violação de restrições para comparar dois indivíduos, levando em consideração um valor de tolerância $\varepsilon$. O valor $\varepsilon$ decresce até atingir um limite de gerações predeterminado e, ao ultrapassar esse limite, $\varepsilon$ é definido como zero, priorizando indivíduos factíveis.

#### LSHADE44 (2017) - 4º lugar

O resolvedor LSHADE44, originalmente proposto por Poláková et al. (2016) para otimização *bound-constrained* e adaptado por Poláková (2017) para instâncias de otimização com restrições, introduz melhorias significativas em relação ao L-SHADE. A principal inovação do LSHADE44 é a utilização de quatro combinações de mutação e cruzamento que competem

entre si para aprimorar o desempenho da busca. Essas combinações incluem as operações de mutação DE/current-to-$p$best/1 e DE/randrl/1, ambas combinadas com cruzamento binomial e exponencial. A estratégia DE/randrl/1 se diferencia da DE/rand/1 ao escolher o melhor dos três indivíduos aleatórios.

As probabilidades de cada estratégia ser escolhida são ajustadas conforme seu sucesso na geração de soluções melhores, começando iguais e sendo adaptadas com base no desempenho. Se alguma probabilidade cai abaixo de um limite, os valores são reinicializados para evitar a estagnação da busca. Cada uma das quatro estratégias possui um par de memórias para autoadaptação dos parâmetros $F$ e $CR$. O sufixo "44"no nome do resolvedor refere-se a essas quatro combinações de memórias e estratégias. Por fim, o método SOF (Seção 2.2.2) é utilizado para a operação de seleção e é aplicada redução linear do tamanho da população da mesma forma que no L-SHADE.

LSHADE-IEpsilon (2018) - 3º lugar

Fan et al. (2018) propuseram uma versão do LSHADE44 associada ao método $\varepsilon$-*constrained*, chamada LSHADE-IEpsilon. Esta abordagem ajusta adaptativamente o valor de $\varepsilon$ de acordo com a proporção de indivíduos factíveis na população atual, balanceando a busca entre regiões factíveis e infactíveis do espaço de soluções. Quando a proporção de indivíduos factíveis é maior que um determinado limiar, o valor de $\varepsilon$ aumenta, incentivando a exploração de regiões infactíveis. Quando a proporção é menor, o valor de $\varepsilon$ diminui, priorizando indivíduos factíveis. Esta adaptação distingue o IEpsilon do método $\varepsilon$-*constrained* original, que apenas diminui o valor de $\varepsilon$.

As estratégias de mutação no LSHADE44-IEpsilon incluem DE/current-to-$p$best/1 e DE/randr1*/1. A última utiliza o indivíduo atual para substituir um dos pais aleatórios, permitindo uma busca mais eficiente em torno das soluções mais promissoras.

LSHADE44+IDE (2017) - 6º lugar

Tvrdík e Poláková (2017) propuseram o resolvedor LSHADE44+IDE, um framework que integra duas fases principais utilizando variantes adaptativas de DE. Na primeira fase, o LSHADE44 é aplicado, priorizando a busca por indivíduos factíveis. A seleção considera a violação de restrições dos indivíduos, similar ao método SOF, até encontrar um número definido de indivíduos factíveis ou atingir o limite de avaliações de fitness. Esses indivíduos formam a população inicial para a fase seguinte. A segunda fase, utilizando o Individual-Dependent Mechanism (IDE), visa minimizar o valor da função objetivo. Um indivíduo experimental é selecionado se sua função objetivo for menor que a do indivíduo atual correspondente e sua violação de restrições não exceder um limite definido. O IDE equilibra a exploração do espaço de busca e a intensificação em torno de regiões promissoras.

O IDE aplica duas estratégias de mutação. Inicialmente, utiliza o próprio indivíduo base para aumentar a diversidade. Na segunda fase, o indivíduo base é escolhido aleatoriamente para acelerar a convergência. A população é dividida em dois conjuntos com base na função objetivo: superior e inferior. A mutação varia dependendo de qual conjunto o indivíduo base pertence, com a estratégia de perturbação ajudando a evitar mínimos locais.

Os parâmetros $F$ e $CR$ são adaptados com base na qualidade dos indivíduos. Indivíduos com melhores valores de função objetivo recebem valores menores de $F$ e $CR$, promovendo a busca local, enquanto indivíduos com piores valores recebem valores maiores, promovendo a diversidade. Assim, a população é mantida ordenada pelo valor da função objetivo.

UDE (2017) - 5º lugar

O resolvedor *Unified Differential Evolution* (UDE), proposto por Trivedi et al. (2017), integra heurísticas de várias variantes adaptativas de DE. Utiliza três estratégias de mutação: DE/rand/1/bin, DE/current-to-rand/1, e DE/current-to-$p$best/1 sem operador de cruzamento. O uso do arquivo externo sugerido por Zhang e Sanderson (2009) não é mencionado no trabalho. UDE incorpora uma versão modificada de operações em que os indivíduos base e terminal das três estratégias de mutação são escolhidos entre os 50% melhores indivíduos da população. Existem duas configurações de parâmetros possíveis, selecionadas aleatoriamente: $(F = 0{,}9; CR = 0{,}9)$ e $(F = 0{,}5; CR = 0{,}5)$.

A cada geração, UDE divide a população em duas subpopulações. Na subpopulação superior, correspondente aos 50% 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 indivíduo contabiliza um sucesso. Na metade inferior da população, apenas uma estratégia de mutação, escolhida adaptativamente, é aplicada a cada indivíduo. A probabilidade de aplicação de uma estratégia de mutação é baseada na taxa de sucesso de cada estratégia na metade superior da população nas últimas $L$ gerações.

Para o tratamento de restrições, o UDE utiliza o método de penalidade estática. Em vez da seleção tradicional do DE, os autores empregaram a estratégia de substituição geracional. Assim, a população de indivíduos experimentais $\mathtt{u}_i$ e a população atual composta pelos indivíduos $\mathtt{x}_i$ são combinadas, e os $NP$ melhores indivíduos são selecionados para a próxima geração.

IUDE (2018) - 1º lugar

O algoritmo *Improved Unified Differential Evolution* (IUDE), proposto por Trivedi et al. (2018), é uma versão aprimorada do UDE (Trivedi et al., 2017). Ele mantém o mesmo esquema de subpopulações e o conjunto de estratégias de mutação: DE/rand/1/bin, DE/current-to-rand/1, e DE/current-to-$p$best/1. Na última estratégia, o arquivo externo proposto por Zhang e Sanderson (2009) é utilizado em instâncias com mais de 50 dimensões para armazenar indivíduos recentemente explorados, fornecendo informações adicionais sobre direções promissoras do espaço de busca. O operador de cruzamento binomial é usado em instâncias com menos de 100 dimensões, enquanto o cruzamento exponencial é empregado em instâncias com maior dimensionalidade.

O IUDE adota um esquema de adaptação de parâmetros similar ao proposto no LSHADE44 (Poláková, 2017). Como o IUDE utiliza três estratégias de mutação, ele emprega um par de memórias $MF$ e $MCR$ para a adaptação dos parâmetros $F$ e $CR$ correspondentes a DE/rand/1 e DE/current-to-$p$best/1, e $MF$ para a adaptação do valor de $F$ correspondente a DE/current-to-rand/1. Apenas a subpopulação superior é usada para a adaptação de parâmetros, pois ela utiliza todas as três estratégias de mutação. A seleção de indivíduos para a operação de mutação segue a mesma estratégia utilizada no UDE.

No início de cada geração, os membros de ambas as subpopulações são ordenados de acordo com o método de superioridade de soluções factíveis (SOF) (Seção 2.2.2). Isso 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 $B$. Assim, o esquema de três estratégias de mutação é sempre implementado na metade superior da população, promovendo uma busca local em torno dos melhores indivíduos e levando a uma convergência mais rápida.

Para comparar os três indivíduos experimentais gerados para cada indivíduo da metade superior da população, é utilizado o método SOF. Em seguida, o método $\varepsilon$-*constrained* é empregado na operação de seleção entre o melhor indivíduo experimental $\mathtt{u}_i$ e o indivíduo

correspondente na população $x_i$. Além disso, o melhor indivíduo em relação ao método SOF é adicionado ao arquivo externo ao final de cada geração. A substituição geracional da população utilizada no UDE foi substituída pela estratégia tradicional de seleção entre dois indivíduos, pois esta apresentou melhores resultados.

$\varepsilon$MAg-ES (2018) - 2º lugar

Hellwig e Beyer (2018) propuseram um método que, diferentemente dos algoritmos das competições passadas do CEC, não se baseia em DE. Trata-se do $\varepsilon$MAg-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 $D$ representando o número de dimensões da instância. Assim como em outros algoritmos evolutivos, o processo de busca ocorre em ciclos através 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, enquanto a operação de mutação corresponde a adicionar um vetor com uma perturbação aleatória de média zero aos indivíduos amostrados. As dependências entre pares de variáveis da instância na distribuição são representadas por uma matriz de covariância, cuja atualização é realizada pelo método *Covariance Matrix Adaptation* (CMA) (Hansen, 2006).

O $\varepsilon$MAg-ES trata restrições de borda (limites 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, sendo que soluções factíveis são consideradas superiores a infactíveis. O método *$\varepsilon$-constrained* é utilizado para a classificação das soluções candidatas. Além disso, o $\varepsilon$MAg-ES utiliza uma abordagem de vizinhanças baseada em gradiente. A aplicação dessas duas técnicas foi inspirada no $\varepsilon$DEga (Takahama e Sakai, 2010), vencedor da competição CEC2010 em otimização de parâmetros reais com restrições (Mallipeddi e Suganthan, 2010).

Conforme pode ser observado nesta seção, alguns participantes da competição CEC2017 concorreram novamente em 2018 e foram superados pelos novos algoritmos. Vale ressaltar que cinco participantes empregam de alguma forma as heurísticas propostas no L-SHADE (Seção 3.1.1). Além disso, identificou-se uma tendência à utilização do método *$\varepsilon$-constrained* (Takahama e Sakai, 2010) para manipulação de restrições.

### 3.1.3 Competição do CEC2020

A competição CEC2020 em otimização não-convexa com restrições de problemas do mundo real avaliou os participantes através de um conjunto de 57 funções, com dimensões variando de 2 a 158, pertencentes ao *benchmark* proposto por Kumar et al. (2020d). Esses problemas incluem processos químicos industriais, síntese e projeto de processos, engenharia mecânica, sistemas de energia, problemas eletrônicos de energia e otimização de alimentação em pecuária.

Na competição, as funções objetivo das instâncias foram tratadas como caixas-fechadas, ou seja, a equação explícita não deveria ser utilizada. As restrições poderiam ser tratadas como caixas-abertas, exceto em um subconjunto de seis funções. Quando as equações das restrições eram utilizadas, as avaliações dessas equações ou de suas derivadas deveriam ser contabilizadas como avaliações de *fitness*.

Os resultados de 25 execuções de cada função foram avaliados utilizando uma métrica que considera uma composição normalizada da melhor solução, da média e da mediana. Quanto maior a dimensionalidade da função, maior seu peso na avaliação. Mais informações sobre essa métrica podem ser obtidas em Kumar et al. (2020c). A classificação geral dos competidores foi:

**1º** SASS;

**2º** COLSHADE;

**3º** sCMAgES;

**4º** EnMODE;

**5º** BP-$\varepsilon$MAg-ES;

**6º** VMCH;

**7º** DEQL;

**8º** FCHA.

A seguir serão apresentados alguns detalhes das heurísticas utilizadas por cada um dos competidores. A fim de facilitar comparações entre suas características, optou-se por apresentá-los de forma a agrupar aqueles que utilizam estratégias semelhantes.

COLSHADE - 2º lugar

Gurrola-Ramos et al. (2020) introduziram uma versão do L-SHADE (Seção 3.1.1) para otimização com restrições, utilizando voos de Lévy para explorar o espaço de busca e um esquema de tolerância dinâmica para o tratamento de restrições. Voos de Lévy, aplicados em meta-heurísticas como *Cuckoo Search* (Yang e Deb, 2009) e *Firefly Algorithm* (Yang, 2010a), consistem em caminhadas aleatórias com passos simulados por uma distribuição com cauda longa, como a distribuição de probabilidade de Lévy (Lee e Yao, 2004).

O COLSHADE utiliza uma estratégia de mutação baseada em voos de Lévy adaptativos para obter uma ampla exploração do espaço de busca, enquanto a estratégia DE/current-to-$p$best/1 com arquivo complementa o processo de otimização reforçando a busca local. A probabilidade de cada estratégia ser escolhida é proporcional ao seu sucesso na geração anterior.

O tratamento de restrições é realizado por um método semelhante ao $\varepsilon$*-constrained* (Seção 2.2.2), onde um valor $\varepsilon$ é subtraído de cada uma das restrições de igualdade ao calcular a violação total do indivíduo. Cada restrição de igualdade possui seu próprio $\varepsilon$, que é atualizado apenas quando pelo menos 20% dos indivíduos são $\varepsilon$-factíveis. O valor inicial da tolerância é igual ao maior valor de violação na população inicial, e o valor final é $\varepsilon = 10^{-4}$. Esse valor é atualizado até que 60% do orçamento de avaliações de *fitness* seja consumido. Após isso, o método de superioridade de soluções factíveis (SOF) (Seção 2.2.2) é utilizado para a operação de seleção entre dois indivíduos.

En-MODE - 4º lugar

O algoritmo *Enhanced Multi-Operator Differential Evolution* (En-MODE), introduzido por Sallam et al. (2020), destaca-se pela sua capacidade de atualizar dinamicamente o tamanho de suas subpopulações com base na qualidade e diversidade dos indivíduos. São utilizadas as estratégias de mutação DE/current-to-$p$best/1 e DE/rand-to-$p$best/1, combinadas com o operador

de cruzamento binomial. Em ambas as estratégias, $p$ é selecionado entre os 10% melhores indivíduos da população atual. Além disso, é empregado um arquivo de soluções candidatas que não obtiveram sucesso na operação de seleção. Quando o arquivo está cheio, o pior indivíduo é removido.

A diversidade é medida com base na distância Euclidiana de cada indivíduo em relação ao melhor indivíduo da subpopulação. A combinação desse valor com a qualidade das soluções determina o número de indivíduos de cada subpopulação na próxima geração. Para evitar que alguma subpopulação fique muito pequena, é utilizado um tamanho mínimo de $NP/10$ indivíduos, onde $NP$ é o tamanho total da população. Ao final de cada geração, os indivíduos são aleatoriamente redistribuídos nas novas subpopulações. Durante o processo evolutivo, é empregado o esquema de redução linear do tamanho da população proposto no L-SHADE (Seção 3.1.1).

Para o tratamento de restrições, é utilizado um esquema que inicialmente ordena as restrições de acordo com a soma de todas as suas violações na população inicial. Em vez de abordar todas as restrições simultaneamente, o processo de busca começa com um subconjunto durante um número predefinido de gerações. Após esse período, um novo subconjunto de restrições é adicionado ao anterior, e o processo de busca tenta encontrar soluções factíveis. Esse ciclo se repete até que todas as restrições sejam incluídas e o critério de parada seja atendido. No processo de seleção entre dois indivíduos, é utilizado o método de superioridade de soluções factíveis (SOF) (Seção 2.2.2).

VMCH - 6º lugar

Wen et al. (2020) propuseram um método chamado *Voting-Mechanism for Constraint Handling* que utiliza um *framework* para tratamento de restrições baseado em voto, uma estratégia de redução de variáveis e o LSHADE44 (Poláková et al., 2016) como máquina de busca.

O *framework* para tratamento de restrições utiliza um mecanismo de votação em que quatro técnicas populares são incluídas. Na operação de seleção entre um indivíduo atual e um experimental, cada técnica vota de acordo com suas próprias regras. Indivíduos que obtém mais votos são selecionados para a próxima geração.

Além das técnicas de tratamento de restrições penalidade autoadaptativa, SOF e $\varepsilon$-*constrained* (Seção 2.2.2), é aplicado o *Ranking* Estocástico, que emprega uma probabilidade $pf = 0{,}475$ ao comparar dois indivíduos candidatos por meio da função objetivo ou da pela violação de restrições.

DEQL - 7º lugar

O DE-QL, apresentado por Kizilay et al. (2020), é um resolvedor que combina DE com o algoritmo de aprendizado por reforço *Q-learning*. Essa estratégia busca escolher a ação mais apropriada com base em experiências passadas, atribuindo recompensas ou penalizações às ações tomadas. As ações neste contexto são as diferentes estratégias de mutação e os valores dos parâmetros $F$ e $CR$.

As estratégias de mutação utilizadas são DE/rand/1, DE/current-to-best/1, DE/current-to-$p$best/1 e DE/best-from-best-to-current/1. Os valores possíveis para os parâmetros $F$ e $CR$ são definidos no intervalo $\{0{,}1, 0{,}2, 0{,}3, \ldots, 0{,}9\}$. Para comparar os indivíduos, é utilizado o método $\varepsilon$-*constrained* (Seção 2.2.2). Além disso, o tamanho da população diminui linearmente, similar ao método empregado no L-SHADE (Seção 3.1.1).

Quando um indivíduo é melhorado pela aplicação de um determinado conjunto de parâmetros (ação), uma recompensa de valor 1 é atribuída a cada um desses parâmetros. Na

próxima aplicação das operações de mutação e cruzamento, a melhor ação (valor ou estratégia) para cada parâmetro será escolhida com base no maior Q-valor, aprimorando assim a eficácia do processo de otimização.

FCHA - 8º lugar

Akhmedova e Stanovov (2020) apresentaram o *Fuzzy Controlled Cooperative Heterogeneous Algorithm* (FCHA), que combina um conjunto de meta-heurísticas evolutivas executadas paralelamente, competindo e colaborando entre si, mediadas por um controlador *fuzzy* autoadaptativo que determina o tamanho da população de cada uma delas.

O FCHA utiliza cinco meta-heurísticas: duas versões do SHADE com as estratégias de mutação DE/rand/1 e DE/current-to-best/1 (Tanabe e Fukunaga, 2013), *Particle Swarm Optimization* (PSO) (Kennedy e Eberhart, 1995), *Cuckoo Search* (Yang e Deb, 2009), e *Bat Algorithm* (Yang, 2010b). Comparações entre indivíduos são realizadas pelo método $\varepsilon$-*constrained* (Seção 2.2.2). As populações das meta-heurísticas trocam informações, onde os piores indivíduos de cada população são substituídos pelos melhores das outras populações, melhorando o desempenho do conjunto de meta-heurísticas de forma cooperativa.

O controlador *fuzzy* ajusta dinamicamente o tamanho da população de cada meta-heurística. Se o valor de *fitness* geral da população não melhora durante um determinado número de gerações, o tamanho da população é aumentado. O controlador também permite o crescimento da população quando uma meta-heurística apresenta a maior taxa de sucesso em uma etapa específica do processo de busca. A taxa de sucesso das meta-heurísticas é definida pela melhor avaliação de sua população.

SASS - 1º lugar

O algoritmo *Self-Adaptive Spherical Search* (SASS) (Kumar et al., 2020b) é uma adaptação do método de busca esférica para otimização com restrições, incorporando um procedimento de autoadaptação de parâmetros. A busca esférica é valorizada por seu equilíbrio entre exploração e busca local, sua capacidade de manter a diversidade populacional e sua invariância rotacional.

Para otimização com restrições, o SASS aplica o método $\varepsilon$-*constrained*. O processo de otimização começa com um conjunto de soluções distribuídas uniformemente no espaço de busca. As soluções candidatas são calculadas utilizando uma matriz de projeção, um fator de controle para o tamanho do salto e uma direção de busca. As soluções candidatas são então comparadas com as soluções anteriores, e as melhores são selecionadas para a próxima geração.

Duas estratégias são usadas para calcular novas soluções: *towards-rand* e *towards-best*. A *towards-rand* promove a exploração do espaço de busca e é aplicada aos 50% melhores indivíduos da população, enquanto a *towards-best* foca na busca local em torno das melhores soluções e é aplicada aos outros indivíduos. Assim, o SASS adapta-se dinamicamente ao processo de busca, equilibrando entre a exploração e a otimização local das soluções. A fim de adaptar o algoritmo de busca esférica para o contexto de otimização com restrições, é aplicado o método $\varepsilon$-*constrained*.

sCMAgES - 3º lugar

Kumar et al. (2020a) propuseram uma versão adaptada do *Covariance Matrix Adaptation Evolutionary Strategy* (CMA-ES) para a otimização de problemas com restrições. O CMA-ES (Hansen et al., 2003) é um método eficaz para a otimização não-convexa e sem derivadas em

espaços de busca contínuos, utilizando apenas a classificação das soluções candidatas para aprender a distribuição das amostras no espaço de busca.

O sCMAgES é uma versão aprimorada do $\varepsilon$MAg-ES (Hellwig e Beyer, 2018), que ficou em segundo lugar na competição de otimização com restrições do CEC2018. Os autores modificaram as etapas básicas de amostragem de novas soluções no CMA-ES para reduzir a complexidade de tempo. Além disso, incluíram um esquema de reinicialização que é acionado quando a melhor solução da população não é atualizada após um certo número de gerações ou quando as soluções convergem para um ponto fixo que pode não ser o ótimo global. Nesse caso, a reinicialização é feita por distribuição uniforme e todos os parâmetros são redefinidos para seus valores padrão.

Para o tratamento de restrições, o método *$\varepsilon$-constrained* é utilizado. Além disso, assim como no SASS (vencedor da competição do CEC2020, também proposto pelos mesmos autores) (Kumar et al., 2020b), é empregado um método de reparação baseado em gradiente para transformar soluções infactíveis em factíveis quando estas se encontram próximas do limite de uma região factível.

BP-$\varepsilon$MAg-ES - 5º lugar

Outra versão aprimorada do $\varepsilon$MAg-ES (Hellwig e Beyer, 2018), que ficou em segundo lugar na competição de otimização com restrições do CEC2018, foi apresentada por Hellwig e Beyer (2020a).

Os autores observaram que, em muitos casos, a melhor solução encontrada pelo $\varepsilon$MAg-ES ocorre após um número relativamente pequeno de avaliações de *fitness*, sugerindo uma convergência prematura. Para mitigar esse problema, eles adicionaram uma estratégia de reinicialização em diferentes áreas do espaço de busca, permitindo a exploração de regiões factíveis disjuntas e aumentando a probabilidade de encontrar o ponto ótimo global. Mesmo em casos de regiões factíveis conectadas, essa estratégia ajuda a superar a convergência prematura.

Além disso, a análise revelou que a escolha adequada do tamanho da população inicial depende significativamente da estrutura da instância. Portanto, o mecanismo de reinicialização utiliza populações de tamanhos variados, equilibrando a necessidade de exploração do espaço de soluções com a busca local eficiente.

Como em outros competidores, é aplicado um processo de reparação de soluções infactíveis baseado em gradiente. Para comparação entre duas soluções, utiliza-se uma versão modificada do *$\varepsilon$-constrained* que permite o aumento do valor da tolerância $\varepsilon$. Esse valor é decrementado quando a porcentagem de soluções $\varepsilon$-factíveis na população excede 20%. Caso contrário, o valor é incrementado para proporcionar mais flexibilidade na busca por regiões promissoras. Após um determinado número de gerações, a tolerância é definida como zero.

A análise dos competidores do CEC2020 revela que a meta-heurística DE continua se destacando na otimização numérica com restrições, com cinco dos oito participantes utilizando as heurísticas do L-SHADE (Seção 3.1.1). A CMA-ES também foi recorrente, inspirando dois competidores devido ao seu bom desempenho na competição do CEC2018 (Seção 3.1.2). A *Spherical Search* superou os demais competidores com sua simplicidade de parâmetros e equilíbrio entre exploração e busca local. As técnicas de tratamento de restrições, *$\varepsilon$-constrained* e superioridade de soluções factíveis (SOF), mantiveram sua relevância. Além disso, com a possibilidade de tratar as equações das restrições como caixas-abertas, três competidores utilizaram estratégias de correção de soluções baseadas em gradiente.

## 3.2 HIBRIDIZAÇÕES DE MÉTODOS DE OTIMIZAÇÃO

Nas últimas décadas, um número crescente de algoritmos têm sido desenvolvidos combinando componentes de diferentes áreas de otimização, indo além do paradigma de uma única meta-heurística tradicional. Esses métodos híbridos combinam elementos de várias técnicas, incluindo busca em árvore, programação dinâmica, métodos de programação matemática, programação por restrições e resolvedores de problemas de satisfatibilidade booleana (Raidl et al., 2019).

A principal motivação para a hibridização de diferentes algoritmos é explorar as vantagens particulares e o caráter complementar dessas estratégias. Métodos híbridos podem se beneficiar dessa sinergia, obtendo resultados significativamente melhores do que suas versões originais. De acordo com Blum e Raidl (2016), a escolha adequada de combinações de algoritmos complementares pode ser crucial para resolver muitos problemas de otimização difíceis.

Diversas classificações e taxonomias de meta-heurísticas híbridas têm sido propostas ao longo dos anos. Alguns trabalhos importantes nesse contexto incluem Cotta-Porras (1998), Talbi (2002), Blum e Roli (2003), Puchinger e Raidl (2005), El-Abd e Kamel (2005), Raidl (2006), Talbi et al. (2013), Talbi (2016) e Blum e Raidl (2016). Nesta tese, é utilizada a classificação de Raidl et al. (2019), que combina aspectos de taxonomias anteriores e apresenta uma visão atualizada dos recursos utilizados em hibridizações. A classificação considera quatro critérios principais: tipos de algoritmos hibridizados, nível de hibridização, ordem de execução e estratégia de controle utilizada.

Quanto aos tipos de algoritmos hibridizados, há várias abordagens possíveis. Primeiramente, pode-se combinar características de diferentes meta-heurísticas ou variações da mesma meta-heurística, utilizando diferentes configurações de parâmetros e componentes de busca. Outra abordagem é hibridizar meta-heurísticas com algoritmos específicos para o problema em questão, como simulações para avaliar soluções candidatas. Além disso, é comum combinar meta-heurísticas com técnicas de outras áreas, como Pesquisa Operacional e Inteligência Artificial. Exemplos incluem métodos exatos baseados em árvore de busca, como *Branch-and-Bound* (B&B), programação dinâmica, programação linear, programação inteira-mista, programação por restrições (CSP) e resolução de problemas de satisfazibilidade (SAT). Outras heurísticas, como redes neurais, lógica *fuzzy* e técnicas estatísticas, também podem ser integradas. Uma quarta possibilidade envolve combinar meta-heurísticas com componentes de busca guiada por humanos, útil em situações onde é difícil quantificar matematicamente a qualidade das soluções.

No que se refere ao nível de hibridização, as combinações podem ser de *alto-nível* ou *baixo-nível*. Em híbridos de alto-nível, os algoritmos originais cooperam sem uma integração profunda de seus funcionamentos internos, mantendo suas identidades próprias. Já em híbridos de baixo-nível, há compartilhamento de componentes, funções ou recursos, tornando os algoritmos interdependentes. As hibridizações podem ser executadas *sequencialmente*, onde os resultados de um algoritmo são usados como entrada para o próximo, ou de forma *intercalada* ou *paralela*, permitindo trocas frequentes de informações, geralmente de forma bidirecional.

Finalmente, as meta-heurísticas híbridas podem ser classificadas pela estratégia de controle, que pode ser *integrativa* ou *cooperativa*. Na abordagem integrativa, um algoritmo é subordinado ao outro, funcionando como um componente embutido no algoritmo principal. Exemplos incluem os *Algoritmos Meméticos* (Moscato, 1999), que usam métodos de busca local para refinar soluções. Na abordagem cooperativa, os algoritmos trocam informações, mas não fazem parte um do outro, como no modelo de ilha (Cohoon et al., 1987), onde subpopulações são otimizadas de forma independente com migração de soluções entre elas. As abordagens

cooperativas podem ser homogêneas (várias instâncias do mesmo algoritmo) ou heterogêneas (hibridização de diferentes algoritmos).

Sabe-se que meta-heurísticas e algoritmos exatos podem ser complementares em termos de qualidade das soluções e tempo de execução utilizado para encontrá-las. Por um lado, a combinação de meta-heurísticas com algoritmos exatos pode melhorar a efetividade dos métodos de busca estocásticos, possibilitando a obtenção de melhores soluções. Por outro lado, esse tipo de combinação pode possibilitar o desenvolvimento de métodos exatos mais eficientes, isto é, que encontram soluções ótimas em menos tempo (Tahami e Fakhravar, 2022). Sendo essa a principal motivação para o desenvolvimento dos modelos híbridos propostos nesta tese, alguns trabalhos relacionados são explorados a seguir. As Seções 3.2.1 e 3.2.2 listam exemplos de hibridizações integrativas e cooperativas da literatura, respectivamente. A fim de garantir a clareza da seção, os trabalhos são apresentados por ordem de similaridade, não por data de publicação.

### 3.2.1 Métodos Integrativos

A forma mais simples de hibridizar dois algoritmos de otimização é integrar um algoritmo dentro de outro a fim de obter soluções iniciais promissoras ou, possivelmente, melhorar soluções intermediárias (Blum e Raidl, 2016). Especialmente no caso de algoritmos evolutivos baseados em população, é extremamente importante iniciar o processo de busca com um conjunto de soluções diversas o suficiente entre si a fim de evitar convergência prematura. Da mesma forma, é desejável que se mantenha um certo nível de diversidade ao longo da execução a fim de explorar amplamente o espaço de busca.

Procedimentos de busca local são frequentemente aplicados dentro de meta-heurísticas para melhorar soluções candidatas intermediárias. A meta-heurística superior é responsável pela diversidade da busca, enquanto o algoritmo interno explora a vizinhança dessas soluções. Exemplos incluem os *Algoritmos Meméticos* (Moscato, 1999), onde o algoritmo evolutivo utiliza técnicas de busca local (como Têmpera Simulada, Gradiente Descendente, Busca Tabu, etc.) para melhorar a qualidade das soluções candidatas criadas a cada iteração.

De acordo com Talbi (2016), no cenário combinatório, boas soluções obtidas por meta-heurísticas podem auxiliar um algoritmo de *Branch and Bound* (B&B) a aplicar uma ordenação eficiente nas caixas a serem exploradas, priorizando aquelas que possuem valores em comum com essas soluções. O algoritmo de B&B constrói soluções parciais usadas para definir o espaço de busca explorado pela meta-heurística. Se a solução obtida pela meta-heurística for factível, ela fornece um limite superior associado a uma caixa do algoritmo exato.

Zhang e Liu (2007) propuseram um algoritmo híbrido que une a técnica de B&B com um Algoritmo Genético (GA). As caixas do B&B limitam o domínio de busca no qual o GA é aplicado, gerando um indivíduo dentro de cada caixa. O GA determina a caixa a ser particionada e o limite superior do ótimo global a cada iteração. Assim, as caixas que não contêm a solução ótima global e os indivíduos nelas contidos são removidos.

Soto et al. (2013) propuseram um algoritmo híbrido para resolver problemas de Sudoku, combinando a busca tabu clássica com o algoritmo de consistência de arco AC3. O objetivo é reduzir o espaço combinatório e acelerar o processo de resolução. A integração do AC3 ocorre tanto na fase de pré-processamento quanto em cada iteração da busca tabu, filtrando os domínios de valores que não levam a uma solução viável. Os experimentos mostraram que essa abordagem supera os melhores métodos incompletos relatados na literatura relacionada a esse problema, resolvendo Sudokus de dificuldade fácil, média e difícil com alta taxa de sucesso e em tempos significativamente menores.

No cenário de otimização combinatória, Cotta e Troya (2003) integraram um algoritmo de B&B na operação de recombinação de um algoritmo evolutivo para selecionar a melhor

solução possível. Os elementos em comum entre os indivíduos pais são mantidos e todas as outras possibilidades são testadas, garantindo a otimalidade local da solução.

Ainda no contexto de otimização combinatória, Ozkan et al. (2019) propuseram três hibridizações para o problema de projeto confiável de redes de comunicação. Na primeira, um algoritmo de B&B é utilizado para reparar uma solução infactível ou melhorar a qualidade da solução encontrada por um método de Têmpera Simulada a cada $k$ iterações. Na segunda, um algoritmo de B&B é usado para corrigir ou melhorar as soluções encontradas por um GA. Por fim, no terceiro método, o algoritmo de B&B é aplicado como operador de mutação no GA, sendo aplicado somente à melhor solução da população atual a cada $k$ iterações, onde $k$ é um valor definido por parâmetro.

Tahami e Fakhravar (2022) exploraram a integração de técnicas de programação matemática e meta-heurísticas no cenário de otimização combinatória, focando em exemplos específicos de como essas hibridizações são aplicadas em problemas de roteamento de veículos, utilizando abordagens como decomposição, heurísticas de melhoria e algoritmos baseados em *Branch & Price* (uma variação de B&B normalmente utilizada para problemas de programação linear inteira e programação linear inteira mista com muitas variáveis).

Bunnag (2014) apresentaram várias combinações entre algoritmos de B&B e de busca estocástica. Para otimização de problemas numéricos sem restrições, foram propostas uma Têmpera Simulada híbrida intervalar e um Algoritmo Genético (GA) híbrido intervalar. Na primeira hibridização, um determinado número de pontos da caixa explorada pelo B&B é selecionado aleatoriamente e otimizado pelo método de Têmpera Simulada para atualizar o mínimo global atual. Já no método que combina B&B e GA, a população é selecionada entre as $NP$ caixas com o menor limite inferior na função objetivo, criando um conjunto de soluções candidatas por meio de cruzamento linear. As informações de dois indivíduos são combinadas e a solução resultante é ajustada para permanecer dentro do domínio de busca. Essa última hibridização também é adaptada para problemas de otimização numérica com restrições. A qualidade de uma caixa é determinada por uma composição entre seu limite inferior da função objetivo, o melhor ponto encontrado pela busca local (Têmpera Simulada ou GA) e, no caso de problemas com restrições, um valor correspondente à estimativa de restrições violadas pela caixa.

Em Cassenote et al. (2019) e Cassenote (2019), foi proposto um otimizador baseado em Evolução Diferencial intervalar denominado *Interval Differential Evolution* (InDE). Essa abordagem introduz uma nova representação para as soluções candidatas: ao invés de uma atribuição de valores reais para todas as variáveis, são consideradas somente aquelas presentes no conjunto $\Omega$ da rede de restrições ternárias da instância (o processo de codificação da rede e de extração do conjunto $\Omega$ são detalhados na Seção 2.1.3). Uma solução candidata é dada por uma atribuição de intervalos para as variáveis, ou seja, é definida por uma caixa. Assim, a população é composta por um conjunto de caixas que cobrem partes do espaço de busca, em vez de apenas pontos específicos desse espaço, como em meta-heurísticas clássicas.

A representação das caixas permite a utilização do contrator GAC (Seção 2.1.3) para descarte de valores sub-ótimos do espaço de busca. A qualidade de uma caixa é determinada pela propagação de valores através da rede de restrições ternárias, utilizando o limite inferior do intervalo que representa a função objetivo da instância. Além disso, o InDE emprega adaptações intervalares de estratégias de DE usados nos otimizadores IUDE (Trivedi et al., 2018) e o LSHADE44 (Poláková, 2017), vencedores das competições do CEC 2017 e 2018, respectivamente, e detalhados na Seção 3.1. A avaliação experimental realizada sobre 80 instâncias do *benchmark* COCONUT (Shcherbina et al., 2003) apontam que a exploração de informações estruturais da instância e o uso de técnicas de consistência local melhoram significativamente o desempenho do DE clássico.

Em Cassenote et al. (2021), foi apresentada uma nova versão desse otimizador denominada *Improved Interval Differential Evolution* (I2DE). Uma das principais contribuições do I2DE é o fato de as soluções candidatas serem avaliadas com base na codificação algébrica original da instância em vez de sua rede de restrições ternárias. Mais especificamente, considera-se somente o valor de propagação pela rede de restrições ternárias das variáveis pertencentes à codificação original da instância, em vez da aproximação dada pela rede completa utilizada no InDE. Além disso, foi realizada a extensão das operações suportadas pelo núcleo multi-intervalar utilizado pelo contrator GAC, conforme detalhado na Seção 4.1.

A população inicial do I2DE é gerada pelos primeiros níveis da árvore de busca de um método de B&B para garantir a cobertura inicial de todo o espaço de busca da instância. Em seguida, a população é subdividida em duas partes, onde estratégias intervalares de mutação são aplicadas em um esquema adaptativo. Soluções candidatas descartadas são armazenadas em um arquivo, e o tamanho da população é reduzido linearmente ao longo da execução. As estratégias de mutação intervalares propostas no InDE foram reformuladas para reduzir erros numéricos. Além disso, foram implementadas diversas adaptações intervalares de estratégias de otimizadores meta-heurísticos, como Brest et al. (2017) e Brest et al. (2016).

A análise experimental do I2DE sobre as 155 instâncias selecionadas do *benchmark* COCONUT (Shcherbina et al., 2003) mostrou que a reformulação das estratégias intervalares de mutação e as novas heurísticas adotadas melhoraram significativamente o desempenho do método de busca, superando o InDE, o OGRe (um otimizador baseado em B&B proposto por Derenievicz (2018) e o BBDE (uma meta-heurística caixa-fechada baseada em DE proposta no mesmo trabalho). Considerando que o BBDE superou o desempenho de diversos otimizadores participantes do CEC2018 e foi superado pelo I2DE, mostrou-se que o uso de informações estruturais da instância no contexto de meta-heurísticas é uma área de pesquisa promissora.

Mais recentemente, em Cassenote et al. (2024), foi publicada uma nova versão do I2DE como um otimizador híbrido em três etapas: exploração, poda e busca local. A etapa de exploração é executada pelas operações intervalares de mutação propostas no I2DE. O contrator GAC é aplicado na etapa de poda para eliminar valores sub-ótimos do espaço de busca. Por fim, sobre cada caixa gerada pelas operações do I2DE e contraída pelo contrator GAC, é aplicada uma etapa de busca local executada pelo BBDE (explorado em detalhes na Seção 4.2) para explorar estocasticamente pontos dentro da caixa. O melhor ponto encontrado dentro de uma caixa representa sua qualidade em relação ao valor da função objetivo e da violação de restrições, guiando a busca em direção às caixas mais promissoras. Sempre que o BBDE encontra um novo ponto factível, seu valor de função objetivo pode ser utilizado para podar o limite superior da solução ótima da instância. Esse processo continua até que um número predefinido de soluções candidatas avaliadas seja atingido. É importante notar que a codificação ternária da instância é explorada apenas durante a execução do contrator GAC.

A avaliação experimental realizada sobre 157 instâncias do *benchmark* COCONUT revelou as vantagens de incorporar uma estratégia de busca local para encontrar o ponto representante da caixa, que difere do valor de propagação pela rede de restrições ternárias utilizado no InDE e no I2DE. Além disso, uma extensiva avaliação experimental com foco na adição da etapa de poda pelo contrator GAC no processo de busca mostrou que as vantagens de incorporar uma etapa de poda baseada em GAC não são limitadas a métodos exatos de otimização. A adição de GAC melhorou significativamente o desempenho da meta-heurística estocástica, com um ganho em qualidade das soluções de 37,15% diante de um acréscimo de apenas 7,82% no tempo de execução. Essa melhoria nas soluções obtidas se deve ao melhor direcionamento dos recursos computacionais disponíveis para as regiões mais promissoras do espaço de busca. A implementação utilizada nesta publicação é a mesma explorada em detalhes na Seção 4.3.

### 3.2.2 Métodos Colaborativos

Sotiropoulos et al. (1997) propuseram uma hibridização entre um B&B e um GA. Inicialmente, o B&B gera uma lista de caixas candidatas de tamanho mínimo definido por parâmetro. A população inicial do GA é definida a partir do ponto médio de cada uma dessas caixas. Sempre que o GA encontra uma nova melhor solução conhecida, o limite superior do intervalo do mínimo global atual é ajustado. Os autores introduzem o conceito de encolhimento de caixas (*shrinking box*), que é utilizado para ajuste do limite inferior do ótimo global atual. Uma caixa é encolhida quando um determinado número de indivíduos da população do GA se encontra dentro dos limites do mínimo global atual $\left[\underline{F^*}, \overline{F^*}\right]$. Sendo os limites da caixa encolhida definidos por $\left[\underline{F^S}, \overline{F^S}\right]$, os limites do mínimo global atual são atualizados de forma que $\underline{F^*} = \max\{\underline{F^*}, \underline{F^S}\}$ e $\overline{F^*} = \min\{\overline{F^*}, \overline{F^S}\}$. Os autores argumentam que a construção da caixa encolhida é um bom indicativo de que a população do GA não está estagnada em um mínimo local do espaço de busca.

Gallardo et al. (2007) apresentaram um modelo híbrido para otimização combinatória que intercala a execução de um Algoritmo Memético (MA) e uma Busca em Feixe (BS). Dado o tempo e memória consumidos pelos algoritmos tradicionais de B&B, a BS se mostra uma alternativa interessante, pois apenas os melhores estados em cada nível da árvore de busca são expandidos, com o número de estados (largura do feixe) previamente definido. Ambos os algoritmos são executados de forma intercalada, compartilhando soluções candidatas. A BS utiliza as informações do MA para descartar caixas que não contêm o mínimo global, enquanto o MA refina sua busca em regiões promissoras. Para avaliar o desempenho do modelo proposto, foram abordados os problemas da mochila multidimensional 0-1 e o problema da super-sequência comum mais curta. A análise experimental evidenciou que o modelo híbrido apresenta melhores resultados do que as versões originais dos algoritmos.

No contexto do problema de otimização numérica de Despacho de Carga Econômico e de Emissões, Gupta e Ray (2010) introduziram um método de Evolução Diferencial (DE) que utiliza análise intervalar em três etapas. Na primeira, os limites da função objetivo são definidos por meio de aritmética intervalar. Na segunda, um algoritmo de B&B intervalar é aplicado ao domínio da instância até que suas caixas atinjam um tamanho mínimo estabelecido por parâmetro, e os pontos médios dessas caixas são definidos como a população inicial do DE. Na terceira fase, é aplicado o encolhimento das caixas (*shrinking box*) para atualizar os limites do ótimo global atual a cada geração. Apesar de proverem um DE intervalar, assim como esta tese, a publicação do método proposto se concentra na modelagem do problema específico para a qual foi concebida, o que dificulta sua generalização.

Mais recentemente, Baltussen et al. (2023) propuseram um método híbrido que combina a meta-heurística de árvore de busca de Monte-Carlo com um B&B para resolver o problema de Dimensionamento e Composição de Frota com Janelas de Tempo. O método de Monte Carlo é utilizado para guiar a exploração do espaço de busca, enquanto o B&B particiona o problema em subproblemas menores, permitindo uma solução eficiente através de processamento paralelo. Os resultados experimentais mostram que a abordagem híbrida reduz significativamente o tempo de execução e melhora a convergência para a solução ótima, reiterando as vantagens da combinação de métodos exatos e meta-heurísticos em problemas complexos de otimização combinatória.

Alliot et al. (2012) propuseram uma hibridização B&B e GA para otimização numérica sem restrições. Os métodos são executados independentemente e trocam informações por meio de memória compartilhada. Enquanto o B&B particiona e remove caixas do espaço de busca, o GA envia ao B&B a melhor solução encontrada, a fim de atualizar o limite superior da função

objetivo e acelerar a poda dos intervalos. Quando o B&B encontra uma solução que melhora o limite superior do mínimo global, seu ponto médio é inserido na população atual do GA, substituindo a pior solução atual para prevenir a convergência prematura a um mínimo local. Periodicamente, uma terceira *thread* desloca os indivíduos da população do GA que se encontram fora das caixas atuais do B&B em direção ao limite da caixa mais próxima no espaço de busca. Essa hibridização garante a otimalidade da solução.

Argumentando que a abordagem de Alliot et al. (2012) ignora informações de monotonicidade local e técnicas de programação por restrições (CSP) que exploram a forma analítica da função objetivo da instância, Vanaret et al. (2013a) propuseram um novo método que une um algoritmo de B&B intervalar a um DE. Eles integraram uma etapa de contração das caixas para podar os limites das variáveis sem perda de soluções. O contrator `HC4` (Benhamou et al., 1999) realiza uma dupla exploração da árvore sintática de uma restrição para contrair cada ocorrência de uma variável, consistindo em uma fase de avaliação (de baixo para cima) que calcula a operação intervalar elementar de cada nó, e uma fase de propagação para trás (de cima para baixo) usando as funções inversas. Esse contrator processa uma restrição após a outra e garante a contração ótima se cada variável tiver somente uma ocorrência na modelagem da instância. O contrator `Box` (Van Hentenryck, 1997) contrai o intervalo de uma variável após a outra com relação a todas as restrições usando uma versão intervalar do método de Newton. Por fim, o contrator `Mohc` (Araya et al., 2010) explora a monotonicidade das restrições para melhorar a contração de `HC4` e do método de Newton intervalar.

Para determinar a estratégia de exploração do espaço de busca pelo B&B, Vanaret et al. (2013a) priorizam as caixas mais distantes da melhor solução atual em uma estratégia chamada *MaxDist*. Os autores argumentam que essa heurística mantém a fila de prioridade menor do que estratégias mais comuns, como busca em largura, busca em profundidade e ordenação pelo menor limite inferior da função objetivo, limitando o custo de inserção e extração das caixas na fila. A bissecção ocorre em cada dimensão de forma circular dentro de cada caixa. Periodicamente, as soluções da meta-heurística são deslocadas para uma posição aleatória dentro da caixa mais próxima no domínio admissível do B&B, evitando a exploração de regiões pouco promissoras do espaço de busca.

Considerando que os indivíduos da população do DE são avaliados com aritmética de ponto flutuante, sujeita a erros numéricos devido à representação computacional, Vanaret et al. (2013a) realizam uma avaliação da extensão intervalar da solução sempre que a melhor solução conhecida é atualizada. O limite superior da função objetivo calculada por aritmética intervalar é compartilhado com o B&B, favorecendo a poda do espaço de busca.

No mesmo ano, Vanaret et al. (2013b) propuseram um método semelhante à hibridização entre B&B e DE anterior, adaptado para problemas com restrições. A principal diferença é que, ao selecionar a próxima caixa a ser explorada, é priorizada aquela com o maior limite inferior na função objetivo. Na execução do DE, dadas as restrições de desigualdade $\{g_i \mid i = 1, \ldots, m\}$, a avaliação de um indivíduo $\mathtt{x}$ é computada como uma tripla $(f_{\mathtt{x}}, n_{\mathtt{x}}, s_{\mathtt{x}})$, em que $f_{\mathtt{x}}$ é o valor da função objetivo, $n_{\mathtt{x}}$ é o número de restrições violadas e $s_{\mathtt{x}} = \sum_{i=1}^{m} \max(g_i(\mathtt{x}), 0)$. Se pelo menos uma das restrições for violada, o valor da função objetivo não é computado. No processo de seleção, dados dois indivíduos $\mathtt{x}$ e $\mathtt{y}$, representados pelas triplas $(f_{\mathtt{x}}, n_{\mathtt{x}}, s_{\mathtt{x}})$ e $(f_{\mathtt{y}}, n_{\mathtt{y}}, s_{\mathtt{y}})$, respectivamente, $\mathtt{x}$ é escolhido para a próxima geração se: $(n_{\mathtt{x}} < n_{\mathtt{y}})$ ou $(n_{\mathtt{x}} = n_{\mathtt{y}} > 0$ e $s_{\mathtt{x}} < s_{\mathtt{y}})$ ou $(n_{\mathtt{x}} = n_{\mathtt{y}} = 0$ e $f_{\mathtt{x}} < f_{\mathtt{y}})$. Caso contrário, $\mathtt{y}$ substitui $\mathtt{x}$ na geração seguinte.

Vanaret et al. (2015) deram continuidade ao seu trabalho propondo um modelo híbrido semelhante ao anterior, mas utilizando `HC4` e `X-Newton` (Araya et al., 2012) no processo de contração de valores infactíveis dos intervalos. O contrator `X-Newton` calcula uma relaxação linear da função objetivo e das restrições, computando um limite inferior do problema inicial

usando técnicas de programação linear, como o algoritmo Simplex. Esse modelo híbrido, denominado Charibde, aplica-se ao cenário de otimização numérica com restrições. Assim, o indivíduo resultante das operações de mutação e cruzamento do DE somente substitui o indivíduo pai na próxima geração em três situações: (i) se violar uma menor quantidade de restrições, (ii) se a soma de suas violações de restrições for menor ou, (iii) no caso de ambos os indivíduos serem factíveis, se o valor de sua função objetivo for menor.

Além disso, Vanaret et al. (2015) argumentam que a prática comum de considerar uma pequena tolerância em restrições de desigualdade pode causar problemas, como $f(\mathrm{x})$ ser menor que o mínimo global ou $\mathrm{x}$ estar muito distante das soluções factíveis atuais no espaço de busca. Para garantir a factibilidade do indivíduo, as restrições de desigualdade são avaliadas por meio de aritmética intervalar, enquanto para as restrições de igualdade, o uso de uma tolerância é mantido. Analogamente, sempre que o DE encontra uma nova melhor solução de ponto-flutuante, ela é avaliada com aritmética intervalar para evitar erros de arredondamento. Se o limite superior da função objetivo dessa solução for melhor do que a melhor solução atual do B&B, então ela é utilizada para descartar soluções sub-ótimas.

Vanaret et al. (2015) também utilizam a estratégia *MaxDist*, priorizando a próxima caixa a ser processada como aquela que se encontra mais distante da melhor solução atual no espaço de busca. Periodicamente, é computada a menor caixa possível contendo todas as caixas da fila de prioridade, que passa a ser utilizada como domínio do DE. Essa operação progressivamente elimina mínimos locais e regiões infactíveis.

Os autores compararam o desempenho do Charibde com alguns otimizadores exatos intervalares (GlobSol, IBBA, Ibex) e otimizadores de programação não-linear que não garantem solução ótima (Couenne, BARON) em um subconjunto de 11 instâncias do benchmark COCONUT (Shcherbina et al., 2003). Os resultados mostram que a hibridização proposta é competitiva contra o segundo grupo de otimizadores e converge mais rapidamente do que os otimizadores do primeiro grupo.

No entanto, é importante notar que cinco das 11 instâncias abordadas tiveram sua codificação algébrica reformulada para reduzir o problema da dependência, que na presença de múltiplas ocorrências de uma mesma variável na codificação, superestima os intervalos resultantes da avaliação de uma solução. Além disso, as configurações de parâmetros do otimizador proposto foram individualmente ajustadas para cada instância. Esse ajuste individual de parâmetros apresenta várias desvantagens, como a falta de generalização, o elevado custo computacional associado ao tempo e esforço necessários para encontrar a melhor configuração para cada caso, a dependência de conhecimento prévio sobre o comportamento do otimizador e as características específicas de cada instância, e dificuldades na reprodutibilidade e comparação dos resultados. Além disso, para se ter uma perspectiva mais clara das potencialidades e limitações do método proposto, seriam necessários testes com um número maior de instâncias, o que permitiria uma avaliação mais abrangente e robusta de seu desempenho. Até onde esta pesquisa pode alcançar, a abordagem de Vanaret et al. (2015) é a que mais se aproxima da proposta relatada no Capítulo 5 deste documento.

## 3.3 CONSIDERAÇÕES

Na Seção 3.1, foram discutidas diversas abordagens que visam ajustar adequadamente os parâmetros das meta-heurísticas para evitar a convergência prematura a um ponto mínimo local em problemas de busca complexos, melhorando assim a qualidade das soluções obtidas. Embora essa área de pesquisa tenha evoluído significativamente nos últimos anos, a natureza estocástica dos métodos de busca e a existência de parâmetros que ainda precisam de ajuste

manual tornam a qualidade das soluções incerta e, em alguns casos, insuficiente para problemas de otimização que exigem resultados de alta qualidade.

Com relação aos métodos de otimização abordados na Seção 3.2, muitas hibridizações entre técnicas exatas e meta-heurísticas são desenvolvidas para o contexto de otimização combinatória, como Zhang e Liu (2007), Gallardo et al. (2007) e Ozkan et al. (2019). Já as abordagens desenvolvidas para o contexto de otimização numérica contínua tendem a se concentrar principalmente em problemas sem restrições, como demonstrado em Sotiropoulos et al. (1997), Zhang e Liu (2007), Gupta e Ray (2010) e Alliot et al. (2012).

Nos exemplos de otimização numérica contínua, geralmente um método de B&B é executado enquanto a meta-heurística realiza sua busca dentro dos subespaços na fila de processamento do B&B. Quando a meta-heurística encontra uma solução promissora, esta é utilizada para atualizar o limite superior da solução ótima do B&B. Por sua vez, quando o B&B encontra uma caixa promissora, seu ponto médio é injetado na população da meta-heurística para evitar a estagnação do processo de busca. Esse tipo de hibridização pode alcançar a solução ótima em um período de tempo mais curto do que as abordagens exatas convencionais.

A hibridização entre métodos exatos e meta-heurísticas que mais se assemelha ao objetivo desta tese é a de Vanaret et al. (2015). Essa abordagem utiliza diversas técnicas de consistência local para poda de regiões sub-ótimas do espaço de busca e é adaptada para problemas de otimização numérica com restrições. A análise comparativa em relação a otimizadores recentes revelou algumas limitações significativas. Cinco das onze instâncias abordadas tiveram sua codificação algébrica reformulada para mitigar o problema da dependência, que superestima os intervalos resultantes da avaliação de uma solução na presença de múltiplas ocorrências de uma mesma variável na codificação. Além disso, as configurações de parâmetros do otimizador proposto foram individualmente ajustadas para cada instância, apresentando desvantagens como falta de generalização e dependência de conhecimento prévio sobre o comportamento do otimizador e as características específicas de cada instância. Para uma avaliação mais precisa das potencialidades e limitações do método proposto, seria necessário realizar testes com um número maior de instâncias, permitindo uma análise mais abrangente e robusta de seu desempenho. Acredita-se que a exploração de heurísticas e técnicas adaptativas destacadas na literatura recente poderia acelerar o processo de busca e melhorar a qualidade das soluções encontradas.

No Capítulo 4 serão apresentadas duas meta-heurísticas híbridas que integram técnicas de consistência local para poda de soluções sub-ótimas e infactíveis do espaço de busca. Em seguida, no Capítulo 5, será introduzido o HIBB, um método de *Branch & Bound* intervalar híbrido que integra técnicas de consistência local e uma busca local baseada em meta-heurística. Com os otimizadores propostos, espera-se explorar de maneira eficaz as vantagens individuais de diferentes áreas, além de abrir caminhos para o desenvolvimento de novas abordagens em otimização numérica com restrições.

# 4 BBDE E HIDE: META-HEURÍSTICAS HÍBRIDAS

Esta tese visa explorar diferentes formas de hibridização entre meta-heurísticas e métodos exatos, a fim de prover boas soluções para instâncias de problemas com uma quantidade crescente de dimensões e com características que tornam sua resolução difícil para meta-heurísticas comuns. Para alcançar esse objetivo, diferentes estratégias de exploração e segmentação do espaço de busca serão analisadas, além da aplicação de técnicas de consistência local para descartar regiões sub-ótimas e infactíveis, acelerando o processo de busca.

Na próxima seção, serão abordados os materiais e métodos utilizados na implementação e avaliação experimental desta pesquisa, com ênfase nas características das instâncias abordadas e nos recursos computacionais empregados. Na Seção 4.2, será apresentado o BBDE, uma meta-heurística que integra diversas heurísticas propostas tanto em trabalhos recentes quanto nesta tese. Por fim, na Seção 4.3, é proposto o HIDE, uma hibridização em três eixos que combina uma meta-heurística baseada em intervalos, o BBDE, e técnicas de consistência local, além de introduzir um contrator de domínio puramente estocástico. Os otimizadores híbridos BBDE e HIDE são duas das contribuições centrais desta tese.

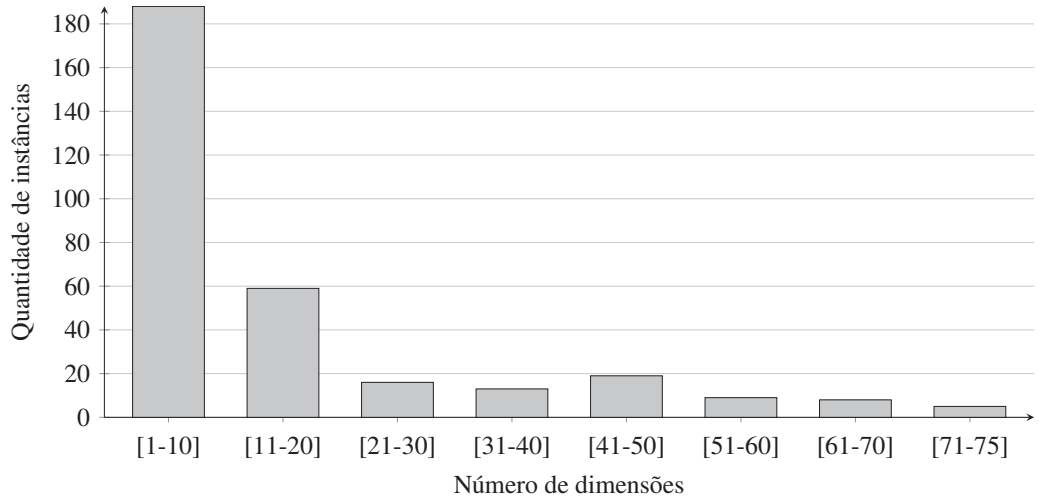
## 4.1 MATERIAIS E MÉTODOS

O primeiro ponto a ser definido é o conjunto de instâncias que serão abordadas. Grande parte das avaliações experimentais de otimizadores baseados em meta-heurísticas utilizam os *benchmarks* das competições do CEC (*Congress on Evolutionary Computation*). No entanto, suas instâncias são disponibilizadas somente como funções em código fonte a serem acessadas como simulações caixa-fechada. Dessa forma, eles não provêm a descrição algébrica necessária para exploração de informações da estrutura da instância, tal como é realizado na aplicação do contrator GAC.

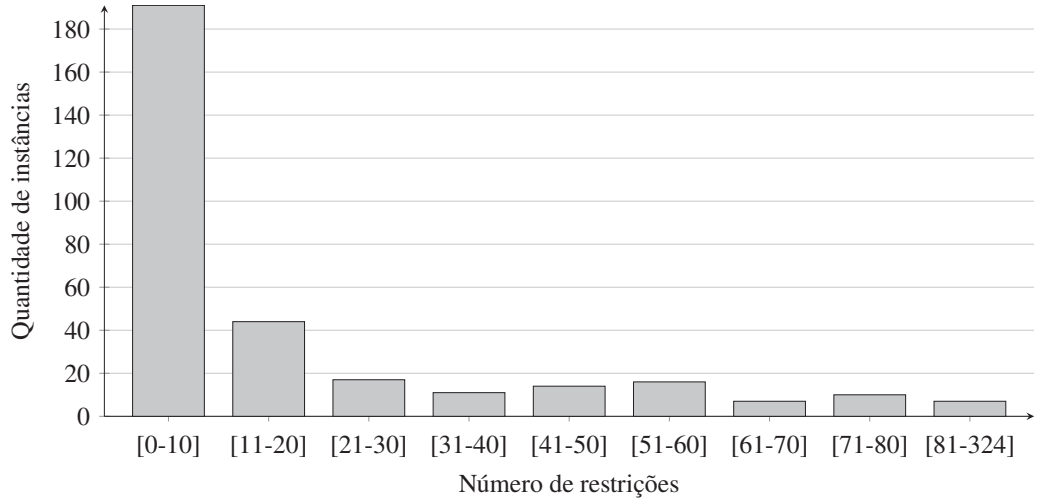
Durante boa parte do desenvolvimento desta tese e nas publicações relacionadas (Cassenote et al., 2019, 2021, 2024) foi utilizado o *benchmark* COCONUT (Shcherbina et al., 2003) devido ao fato de ele fornecer a descrição AMPL (*A Mathematical Programming Language*) necessária para a exploração da estrutura das instâncias no processo de consistência local. Contudo, com a fase de implementação e testes dos resolvedores propostos, detectou-se a necessidade de um *benchmark* mais robusto. A partir daí, foi adotada a MINLPLib[1] (uma biblioteca de instâncias de programação inteira-mista e não-linear contínua). A MINLPLib tem oferecido atualizações constantes, com a inclusão de novas instâncias, novas soluções e mais informações sobre cada instância, além de prover descrições algébricas e cobrir uma grande variedade de problemas do mundo real.

A partir daí, iniciou-se a seleção das instâncias da MINLPLib a serem utilizadas na avaliação experimental desta tese. Foram descartadas instâncias com variáveis inteiras, binárias, sem valor ótimo definido, sem descrição AMPL, de maximização e com operadores não suportados pelo núcleo multi-intervalar utilizado no processo de filtragem de domínio por consistência local. Com isso, foi obtido um subconjunto de 517 instâncias com até 900.023 variáveis e 1.016.476 restrições. Devido ao tempo de processamento demandado para o ajuste de parâmetros e o teste de diferentes heurísticas, optou-se por limitar a avaliação experimental a um subconjunto de 317 instâncias com até 75 variáveis e 324 restrições. A Figura 4.1 mostra a distribuição dessas instâncias em relação ao número de dimensões (variáveis) e de restrições.

[1]Disponível em `http://minlplib.org`, acessado em 24/06/2024.




(a) Distribuição das instâncias em relação ao número de dimensões.




(b) Distribuição das instâncias em relação ao número de restrições.

Figura 4.1: Distribuição do subconjunto de 317 instâncias do benchmark MINLPLib a ser utilizado na avaliação experimental.

É importante notar que a maior diversidade de instâncias abordadas nesta tese em relação às publicações anteriores se deve à extensão da implementação do núcleo multi-intervalar, realizada pelo grupo de pesquisa ao qual este trabalho está associado. Este núcleo é utilizado na etapa de filtragem de domínio por GAC (descrita na Seção 2.1.3 deste documento). Na versão original proposta por Derenievicz (2018), era possível abordar apenas instâncias com as operações $+$, $-$, $*$, $/$, $\wedge$ e $\sqrt{}$. Em Cassenote et al. (2021), essa implementação foi estendida para incluir operações adicionais, como *log*, *exp*, *sin*, *cos*, *tan*, *abs*, *sign*, *max* e *min*.

Outra contribuição do grupo de pesquisa associado a esta tese é a criação de um *parser* que converte instâncias no formato AMPL em duas representações distintas: uma modelagem algébrica em forma de rede de restrições ternárias e um código-fonte que permite a avaliação dos valores da função objetivo e das violações de restrições a partir das coordenadas das soluções candidatas. Vale notar que, nos otimizadores propostos, a modelagem algébrica das instâncias é acessada apenas durante a etapa de consistência local, enquanto todas as demais operações são realizadas sobre a codificação original da instância de modo caixa-fechada.

A transformação da modelagem original da instância em uma rede de restrições ternárias geralmente envolve a adição de uma variável auxiliar para cada operador presente na descrição

algébrica da instância. Dessa forma, são extraídas as restrições pertencentes ao conjunto Ω, cuja valoração é essencial para descartar regiões sub-ótimas e infactíveis do domínio da instância. Este processo é detalhado na Seção 2.1.3 desta tese e foi proposto por Derenievicz (2018); Derenievicz e Silva (2018).

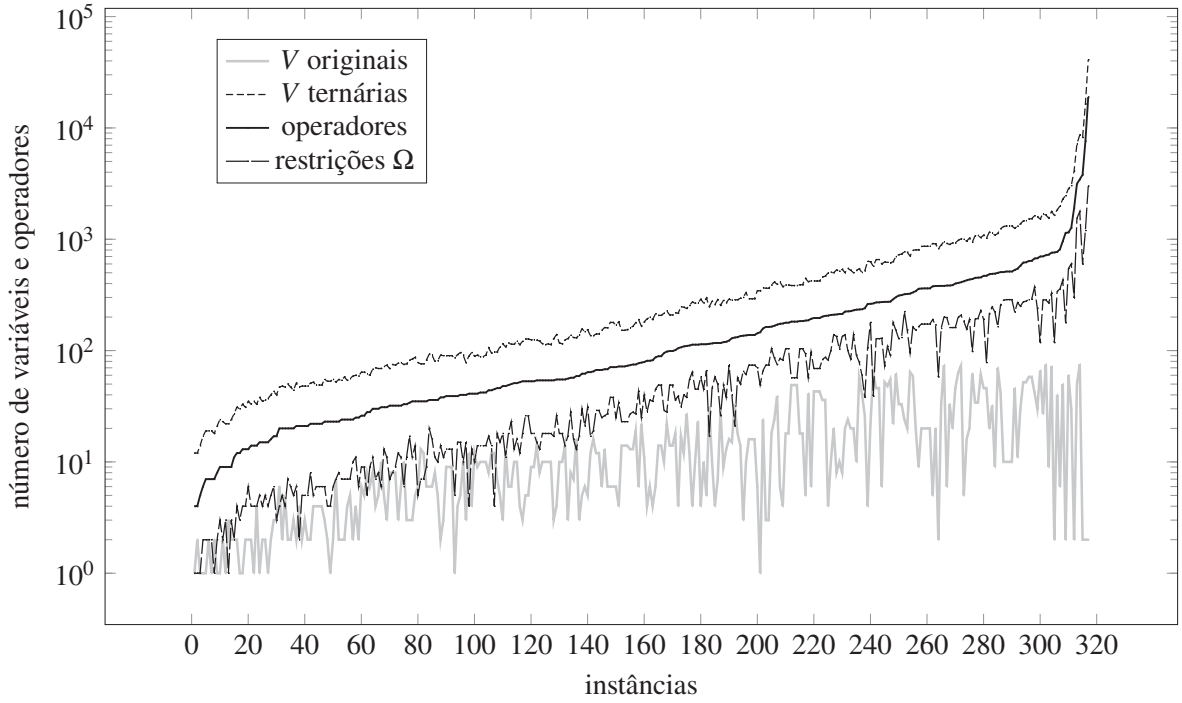



Figura 4.2: Número de variáveis na representação original, na representação ternária, Ω e operadores por instância.

A Figura 4.2 mostra o número de variáveis (e operadores) da modelagem original, da representação ternária e a quantidade de restrições de Ω para as 317 instâncias selecionadas da MINLPLib. Apesar de Ω apresentar uma redução considerável no número de restrições em relação à representação ternária, o espaço de busca explorado na etapa de poda possui um número maior de dimensões do que na modelagem original da instância. É importante notar que quanto maior a quantidade de operadores na formulação da instância, maior será a rede de restrições ternárias.

Toda a avaliação experimental associada a esta tese foi executada em um servidor com processadores Intel Xeon E5-4627v2 3.30GHz, 256GB de RAM, e Debian Linux 11.7 pertencente ao Centro de Computação Científica e Software Livre (C3SL)[2] da UFPR. Todos os otimizadores apresentados nesta tese foram implementados em linguagem C. Devido à natureza estocástica dos otimizadores, foram realizadas 15 execuções por instância em cada experimento.

Para garantir uma comparação justa, todos os otimizadores propostos utilizam a mesma implementação do módulo de filtragem de domínio por GAC, com o número máximo de passos $MaxSteps = 100$. Esse valor foi definido com base nos resultados experimentais apresentados em Cassenote et al. (2024), que traz uma análise entre diferentes configurações avaliadas sobre o otimizador HIDE (Seção 4.3). Além disso, o orçamento máximo de soluções candidatas avaliadas foi padronizado para todos os otimizadores como $MaxFEs = 10^7 \times D$, onde $D$ representa o número de dimensões da instância. Soluções cuja soma de violação de restrições seja menor que $10^{-8}$ são consideradas factíveis. Da mesma forma, instâncias com erro absoluto em relação à solução ótima menor que $10^{-4}$ são consideradas ótimas.

[2]Disponível em `https://www.c3sl.ufpr.br/`, acessado em 07/07/2024.

Para uma análise comparativa mais precisa entre diferentes configurações de parâmetros e diferentes otimizadores, tornou-se necessário estabelecer um método que considerasse mais informações do que simplesmente a quantidade de soluções ótimas e factíveis. A partir disso, foram adotados os critérios de avaliação utilizados na competição do CEC2020 em otimização não-convexa de problemas do mundo real com restrições (Kumar et al., 2020d)[3], que envolve os valores de função objetivo e de violação de restrições da melhor solução, da média e da mediana entre as 15 execuções. Todas as 317 instâncias são consideradas igualmente e a melhor, a média e a mediana das soluções possuem pesos de 30%, 50% e 20%, respectivamente.

Nas próximas seções, serão apresentados dois otimizadores propostos nesta tese. Primeiramente, o BBDE, uma meta-heurística que incorpora estratégias de otimizadores recentes para explorar pontos no espaço de busca de maneira eficiente. Em seguida, será introduzido o HIDE, uma meta-heurística baseada em intervalos que explora sub-regiões do espaço de busca.

## 4.2 BBDE: UMA EVOLUÇÃO DIFERENCIAL HÍBRIDA

Em Cassenote et al. (2021) foi introduzido o BBDE (do inglês, *Black-box Differential Evolution*), uma meta-heurística caixa-fechada que incorpora diversas estratégias de otimizadores recentes baseados em Evolução Diferencial (DE). Na análise experimental publicada, o BBDE superou o desempenho de diversos competidores do CEC2018, descritos na Seção 3.1 desta tese, sem utilizar nenhum tipo de técnica de consistência local ou informações da estrutura algébrica da instância. Dessa forma, acredita-se que o BBDE seja um resolvedor adequado para análise do comportamento de uma meta-heurística clássica diante das 317 instâncias selecionadas da MINLPLib.

A exploração do espaço de busca se dá por uma população de $NP$ soluções candidatas (ou indivíduos), assim como no DE clássico descrito na Seção 2.2.1 deste documento. O esquema em que a população atual é subdivida entre as subpopulações $A$ e $B$ de tamanho $NP/2$ proposto no IUDE (Trivedi et al., 2018) é aplicado. A população é mantida ordenada, o que permite que a subpopulação $A$ contenha as melhores soluções candidatas. A fim de promover a busca local em torno dessas regiões, é aplicado um conjunto de três estratégias de mutação a cada indivíduo de $A$. O melhor entre os três é comparado ao indivíduo atual na operação de seleção executada pelo método $\varepsilon$-*constrained* (Takahama e Sakai, 2010). Informações da subpopulação $A$ são utilizadas para determinar qual estratégia de mutação será aplicada a cada indivíduo da subpopulação $B$. Ao final de cada geração, todos os indivíduos são ordenados e divididos entre as duas subpopulações, que têm seus tamanhos linearmente reduzidos de acordo com o esquema proposto por Tanabe e Fukunaga (2014).

É sabido que as configurações adequadas dos parâmetros $F$ e $CR$ são dependentes da instância e podem mudar de acordo com a região do espaço de busca que está sendo percorrida. Para tornar esses parâmetros autoadaptativos, o BBDE emprega o esquema introduzido no SHADE (Tanabe e Fukunaga, 2013) que utiliza um par de memórias $\langle M_F, M_{CR}\rangle$, com 5 posições cada, para armazenar as configurações que obtiveram sucesso nas gerações recentes. A cada geração, uma das posições é circularmente atualizada com base nas diferenças de avaliação entre indivíduos experimentais e atuais.

No BBDE foram incorporadas heurísticas propostas por Brest et al. (2016) e Brest et al. (2017). As primeiras 4 posições do vetor de memórias são inicializadas com $\langle 0.8, 0.3\rangle$. A última posição é definida como $\langle 0.9, 0.9\rangle$ e permanece inalterada. A cada geração, uma das primeiras 4

[3]Os registros da competição e o algoritmo utilizado para classificação dos otimizadores se encontram disponíveis em `https://github.com/P-N-Suganthan/2020-RW-Constrained-Optimisation`, acessado em 09/07/2024.

posições é circularmente atualizada com base na distância entre as coordenadas dos indivíduos experimentais e atuais. Valores muito grandes de $F$ e muito pequenos de $CR$ não são permitidos nos primeiros estágios do processo de busca. Somente indivíduos experimentais da subpopulação $A$ que obtiveram sucesso são utilizados na adaptação de parâmetros, uma vez que somente nessa subpopulação as três estratégias de mutação são aplicadas a cada indivíduo.

As configurações de parâmetros do BBDE foram empiricamente definidas após uma série de experimentos preliminares. O operador DE/current-to-$p$best/1 utilizou $p = 25\%$. Os parâmetros do $\varepsilon$-*constrained* foram $\theta = 0.7$, $cp = 4$ e $T = 0.85 \times MaxFEs$. Foi utilizada a versão exponencial do operador de cruzamento.

Na análise experimental publicada em Cassenote et al. (2021), foi definido o orçamento de $MaxFEs = 2 \times 10^4 \times D$ para uma população inicial de tamanho $NP = 15 \times D$, em que $D$ é o número de dimensões da instância. Para tornar o desempenho do BBDE comparável com os otimizadores HIDE e HIBB propostos neste documento, foi estabelecido o orçamento $MaxFEs = 10^7 \times D$ para todos os otimizadores híbridos propostos. Sabendo que existe relação entre o orçamento e o tamanho adequado da população $NP$, é também necessário realizar o ajuste desse parâmetro. Como os experimentos conduzidos nesta tese utilizam um conjunto diferente de instâncias, foram testadas diversas configurações para $NP$, inclusive $7500 \times D$, que seria uma proporção direta em relação ao valor utilizado na publicação que introduz o BBDE (Cassenote et al., 2021).

A Tabela 4.1 traz uma análise detalhada sobre a quantidade de instâncias para as quais foram obtidas soluções factíveis e ótimas, além do tempo médio demandado pelas 15 execuções de cada experimento. O ajuste de $NP$ teve um impacto significativo no desempenho do BBDE, chegando a 287 instâncias factíveis com $NP = 25000 \times D$ e 154 soluções ótimas com $NP = 5000 \times D$. Conforme esperado, a maior proporção de soluções factíveis e ótimas se encontra nos grupos de instâncias com menor quantidade de dimensões. Adicionalmente, parece haver um acréscimo no tempo de execução conforme se aumenta o valor de $NP$, o que pode estar relacionado com a manipulação da estrutura de dados que armazena essas informações.

Tendo em vista que somente o número de instâncias com soluções factíveis e ótimas pode não ser suficiente para tomar uma decisão adequada em relação à melhor configuração de parâmetros, na Tabela 4.2 é realizada uma análise comparativa entre diferentes configurações de $NP$ de acordo com o método de classificação do CEC2020 (comentada na Seção 4.1), que considera melhor a configuração que obtém a menor pontuação. De acordo com a composição da melhor, da média e da mediana entre as 15 execuções, a configuração $NP = 25000 \times D$ apresentou o melhor desempenho e será adotada para o BBDE no restante deste documento.

Ao longo dos capítulos anteriores, muito se falou sobre a possibilidade de meta-heurísticas ficarem estagnadas em pontos específicos do espaço de busca devido a características da instância abordada. Nesse tipo de situação, uma parte considerável do orçamento é gasta sem que haja mudanças na melhor solução encontrada. Tendo em vista que na maioria dos casos a solução ótima não é informada ao otimizador, também podem ocorrer situações em que o ótimo é rapidamente encontrado e o orçamento segue sendo consumido até o fim. Nesse caso, o BBDE apresentou uma média de 89,79% do orçamento consumido até a melhor solução obtida, com 232 das 317 instâncias (73,18%) com média acima de 95%. Isso sugere que os resultados poderiam ser ainda melhores caso o orçamento disponível fosse maior.

Apesar de a média de orçamento consumido até a melhor solução conhecida indicar que o BBDE consegue evitar a estagnação em pontos sub-ótimos, não é uma tarefa trivial estimar a melhoria das soluções das 15 execuções de cada uma das 317 instâncias durante todo o processo de busca. Uma prática comum no desenvolvimento de métodos de meta-heurísticas é a utilização de esquemas de reinício (Öztop et al., 2022; Hellwig e Beyer, 2020b), em que periodicamente

Tabela 4.1: Quantidade de instâncias com soluções factíveis, ótimas e tempo médio de execução obtidos pelo BBDE com diferentes configurações de *NP*.

| | | $NP = 5000 \times D$ | | | $NP = 7500 \times D$ | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 160 | 124 | 35,7 | 165 | 124 | 37,3 |
| 11 a 20 | 59 | 49 | 21 | 148,0 | 50 | 20 | 151,6 |
| 21 a 30 | 16 | 15 | 4 | 339,9 | 15 | 4 | 340,1 |
| 31 a 40 | 13 | 12 | 3 | 453,7 | 12 | 3 | 454,5 |
| 41 a 50 | 19 | 16 | 2 | 878,5 | 14 | 0 | 876,4 |
| 51 a 60 | 9 | 8 | 0 | 946,8 | 9 | 0 | 942,2 |
| 61 a 70 | 8 | 5 | 0 | 1220,1 | 5 | 0 | 1204,6 |
| 71 a 75 | 5 | 4 | 0 | 2140,4 | 4 | 0 | 2144,1 |
| total | 317 | 269 | 154 | 228,6 | 274 | 151 | 229,7 |
| | | $NP = 10000 \times D$ | | | $NP = 25000 \times D$ | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 167 | 127 | 36,4 | 179 | 133 | 40,8 |
| 11 a 20 | 59 | 51 | 18 | 149,5 | 52 | 11 | 159,5 |
| 21 a 30 | 16 | 15 | 4 | 343,1 | 13 | 4 | 355,1 |
| 31 a 40 | 13 | 12 | 3 | 464,6 | 12 | 2 | 480,0 |
| 41 a 50 | 19 | 14 | 0 | 896,9 | 14 | 0 | 916,2 |
| 51 a 60 | 9 | 9 | 1 | 975,0 | 9 | 1 | 1004,2 |
| 61 a 70 | 8 | 6 | 0 | 1240,5 | 5 | 0 | 1262,5 |
| 71 a 75 | 5 | 3 | 0 | 2211,7 | 3 | 0 | 2243,8 |
| total | 317 | 277 | 153 | 233,4 | 287 | 151 | 242,1 |
| | | $NP = 30000 \times D$ | | | $NP = 40000 \times D$ | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 178 | 133 | 41,1 | 176 | 134 | 42,4 |
| 11 a 20 | 59 | 51 | 12 | 161,8 | 52 | 10 | 164,8 |
| 21 a 30 | 16 | 13 | 4 | 359,0 | 13 | 4 | 362,6 |
| 31 a 40 | 13 | 12 | 2 | 483,6 | 12 | 0 | 482,7 |
| 41 a 50 | 19 | 14 | 0 | 916,8 | 13 | 0 | 921,7 |
| 51 a 60 | 9 | 8 | 1 | 1001,9 | 8 | 1 | 1002,9 |
| 61 a 70 | 8 | 5 | 0 | 1263,6 | 5 | 0 | 1276,4 |
| 71 a 75 | 5 | 3 | 0 | 2263,6 | 3 | 0 | 2226,8 |
| total | 317 | 284 | 152 | 243,0 | 282 | 149 | 245,0 |

Tabela 4.2: Classificação do BBDE com diferentes configurações de *NP* de acordo com o método do CEC2020.

| *NP* | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| **$25000 \times D$** | **0,1960** | **0,1990** | **0,1854** | **0,1954** | **1º** |
| $7500 \times D$ | 0,1962 | 0,2187 | 0,1915 | 0,2065 | 2º |
| $10000 \times D$ | 0,1972 | 0,2160 | 0,2005 | 0,2072 | 3º |
| $30000 \times D$ | 0,2264 | 0,2178 | 0,2120 | 0,2192 | 4º |
| $5000 \times D$ | 0,1984 | 0,2319 | 0,2189 | 0,2192 | 5º |
| $40000 \times D$ | 0,3487 | 0,3482 | 0,3565 | 0,3500 | 6º |

parte da população é aleatoriamente redistribuída a fim de evitar a convergência prematura em pontos sub-ótimos.

Dessa forma, foi criada uma versão do BBDE com reinícios em que, a cada $10000 \times D$ soluções candidatas avaliadas, quase toda a população é aleatoriamente redistribuída pelo espaço de busca. O tamanho da população foi redefinido para $NP = 10 \times D$. Apenas a melhor solução candidata encontrada até o momento é preservada a cada reinício, o que permite a ampliação da exploração sem abandonar regiões promissoras já descobertas. As demais heurísticas e configurações de parâmetros do BBDE foram mantidas nessa versão.

Tabela 4.3: Quantidade de instâncias com soluções factíveis, ótimas e tempo médio de execução obtidos pelo BBDE e pelo BBDE com reinícios.

| | | BBDE | | | BBDE com reinícios | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 179 | 133 | 40,8 | 179 | 140 | 26,2 |
| 11 a 20 | 59 | 52 | 11 | 159,5 | 53 | 25 | 95,8 |
| 21 a 30 | 16 | 13 | 4 | 355,1 | 13 | 4 | 243,2 |
| 31 a 40 | 13 | 12 | 2 | 480,0 | 12 | 0 | 305,7 |
| 41 a 50 | 19 | 14 | 0 | 916,2 | 13 | 0 | 670,5 |
| 51 a 60 | 9 | 9 | 1 | 1004,2 | 6 | 1 | 695,3 |
| 61 a 70 | 8 | 5 | 0 | 1262,5 | 5 | 0 | 943,2 |
| 71 a 75 | 5 | 3 | 0 | 2243,8 | 3 | 0 | 1858,4 |
| total | 317 | 287 | 151 | 242,1 | 284 | 170 | 171,2 |

A Tabela 4.3 traz uma análise da quantidade de instâncias com soluções factíveis, ótimas e do tempo médio das 15 execuções do BBDE e do BBDE com reinícios. A mudança em relação à quantidade de instâncias para as quais foram encontradas soluções factíveis foi pequena. No entanto, o número de soluções ótimas aumentou de forma considerável, especialmente no grupo de instâncias de 11 a 20 dimensões. Adicionalmente, houve uma redução expressiva no tempo de execução com a adoção do esquema de reinícios. Acredita-se que isso tenha relação com a manipulação da estrutura de dados que armazena a população de soluções candidatas, que passa do tamanho $NP = 25000 \times D$ para $NP = 10 \times D$.

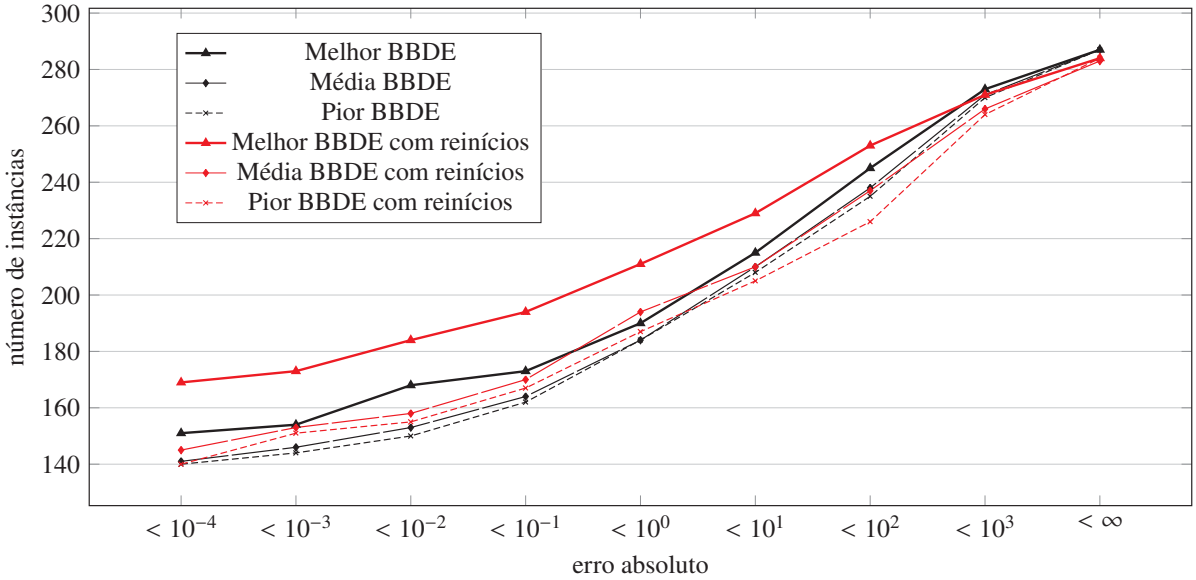



Figura 4.3: Número acumulado de instâncias com soluções factíveis obtidas por BBDE e BBDE com reinícios dados alguns limites de erro absoluto.

A melhoria na qualidade das soluções fica evidente também na Figura 4.3, que exibe o número acumulado de instâncias com soluções factíveis dados alguns limites de erro absoluto (diferença absoluta entre o valor da função objetivo da solução encontrada e da solução ótima). As linhas representam a melhor, a média e a pior das 15 execuções do BBDE e do BBDE com reinícios.

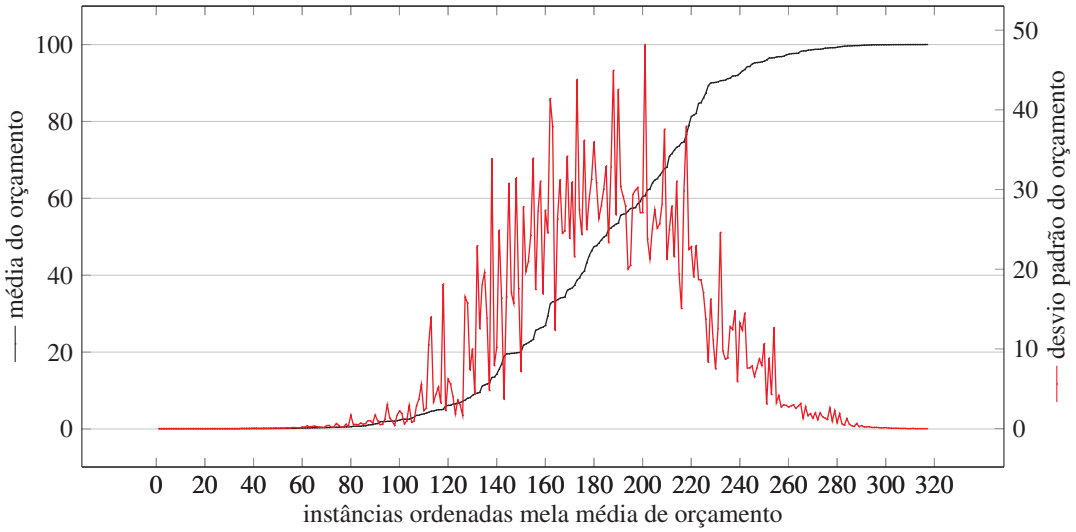



Figura 4.4: Média e desvio padrão da porcentagem de orçamento gasto nas 15 execuções de cada instância até a melhor solução encontrada pelo BBDE com reinícios.

A Figura 4.4 mostra a média e o desvio padrão da porcentagem de orçamento consumido pelas 15 execuções do BBDE com reinícios até a última atualização da melhor solução conhecida. Tendo em vista que não parece haver relação entre o número de dimensões da instância e os dados mostrados na figura, optou-se pela ordenação das instâncias pela média de orçamento para facilitar a visualização dos dados. Apenas 72 das 317 instâncias (22,71%) obtiveram uma média de orçamento gasto acima de 95%, contra as 232 (73,18%) observadas no BBDE. Combinando essas informações com o aumento de 19 instâncias com soluções ótimas, acredita-se que a redução da média de orçamento gasto seja reflexo de uma convergência mais rápida em direção ao ponto ótimo e sua vizinhança. No entanto, o desvio padrão elevado em algumas instâncias sugere uma grande variação nos resultados obtidos em diferentes execuções.

Por fim, com o intuito de avaliar uma hibridização entre as duas meta-heurísticas propostas e o contrator GAC (descrito na Seção 2.1.3 deste documento), foram implementadas versões do BBDE e do BBDE com reinícios associadas à uma etapa de poda. Mais precisamente, a cada atualização da melhor solução conhecida, testa-se se ela é factível. Em caso positivo, ela é fornecida ao contrator GAC para que seu valor de função objetivo seja utilizado para descarte de soluções sub-ótimas e infactíveis do espaço de busca conforme descrito no Algoritmo 2 da Seção 2.1.3 deste documento. A única configuração de parâmetro a ser ajustada para aplicação do contrator GAC é a quantidade máxima de passos utilizados no processo de poda, definida como $MaxSteps = 100$ com base nos resultados experimentais publicados em Cassenote et al. (2024).

A Tabela 4.4 mostra a quantidade de instâncias com soluções factíveis, ótimas e o tempo médio das 15 execuções de cada instância obtidos pelos otimizadores BBDE, BBDE com reinícios e por suas respectivas versões com o contrator GAC. Em relação ao BBDE, a adição da etapa de poda parece não ter gerado um impacto muito significativo em relação ao número de instâncias para as quais foram encontradas soluções factíveis e ótimas, enquanto no BBDE com

Tabela 4.4: Quantidade de instâncias com soluções factíveis, ótimas e tempo médio de execução obtidos pelo BBDE, pelo BBDE com reinícios e por suas versões com o contrator GAC.

| | | BBDE | | | BBDE com reinícios | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 179 | 133 | 40,8 | 179 | 140 | 26,2 |
| 11 a 20 | 59 | 52 | 11 | 159,5 | 53 | 25 | 95,8 |
| 21 a 30 | 16 | 13 | 4 | 355,1 | 13 | 4 | 243,2 |
| 31 a 40 | 13 | 12 | 2 | 480,0 | 12 | 0 | 305,7 |
| 41 a 50 | 19 | 14 | 0 | 916,2 | 13 | 0 | 670,5 |
| 51 a 60 | 9 | 9 | 1 | 1004,2 | 6 | 1 | 695,3 |
| 61 a 70 | 8 | 5 | 0 | 1262,5 | 5 | 0 | 943,2 |
| 71 a 75 | 5 | 3 | 0 | 2243,8 | 3 | 0 | 1858,4 |
| total | 317 | 287 | 151 | 242,1 | 284 | 170 | 171,2 |
| | | BBDE + GAC | | | BBDE com reinícios + GAC | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 180 | 135 | 42,8 | 183 | 143 | 26,0 |
| 11 a 20 | 59 | 52 | 10 | 160,8 | 53 | 32 | 95,7 |
| 21 a 30 | 16 | 13 | 4 | 354,2 | 13 | 4 | 244,1 |
| 31 a 40 | 13 | 12 | 2 | 471,6 | 12 | 0 | 304,7 |
| 41 a 50 | 19 | 14 | 0 | 901,3 | 12 | 0 | 675,0 |
| 51 a 60 | 9 | 9 | 1 | 964,9 | 7 | 1 | 688,1 |
| 61 a 70 | 8 | 5 | 0 | 1250,9 | 5 | 0 | 961,9 |
| 71 a 75 | 5 | 2 | 0 | 2153,5 | 3 | 0 | 1868,6 |
| total | 317 | 287 | 152 | 239,5 | 288 | 180 | 171,8 |

reinícios se observa uma pequena melhoria nos dois grupos com menor dimensionalidade. Em ambos os casos, também houve uma pequena redução no tempo médio de execução.

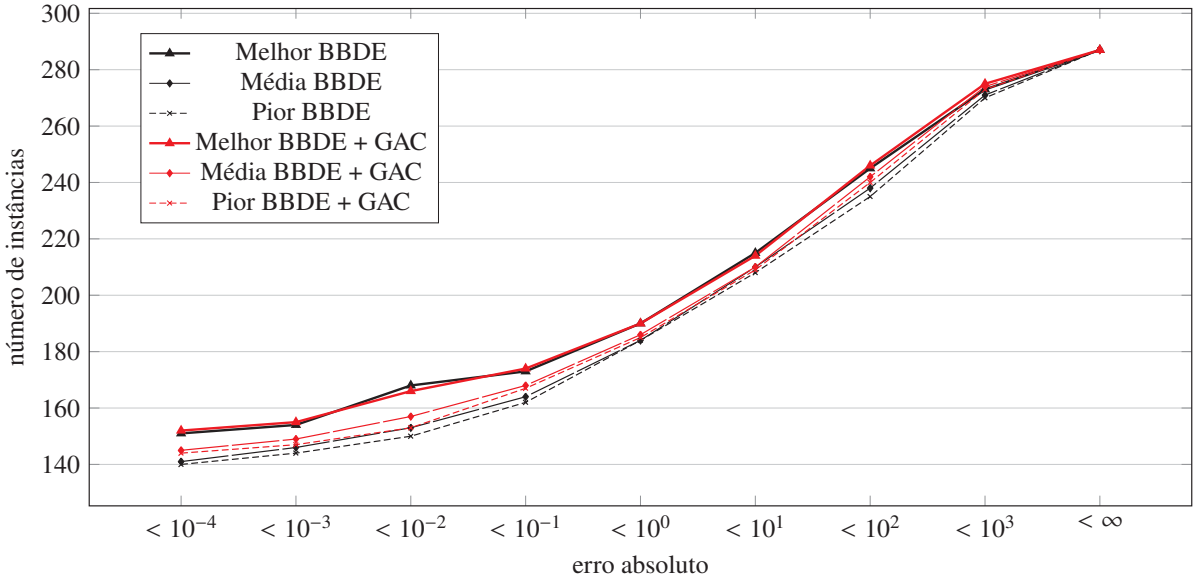



Figura 4.5: Número acumulado de instâncias com soluções factíveis obtidas por BBDE e BBDE + GAC dados alguns limites de erro absoluto.

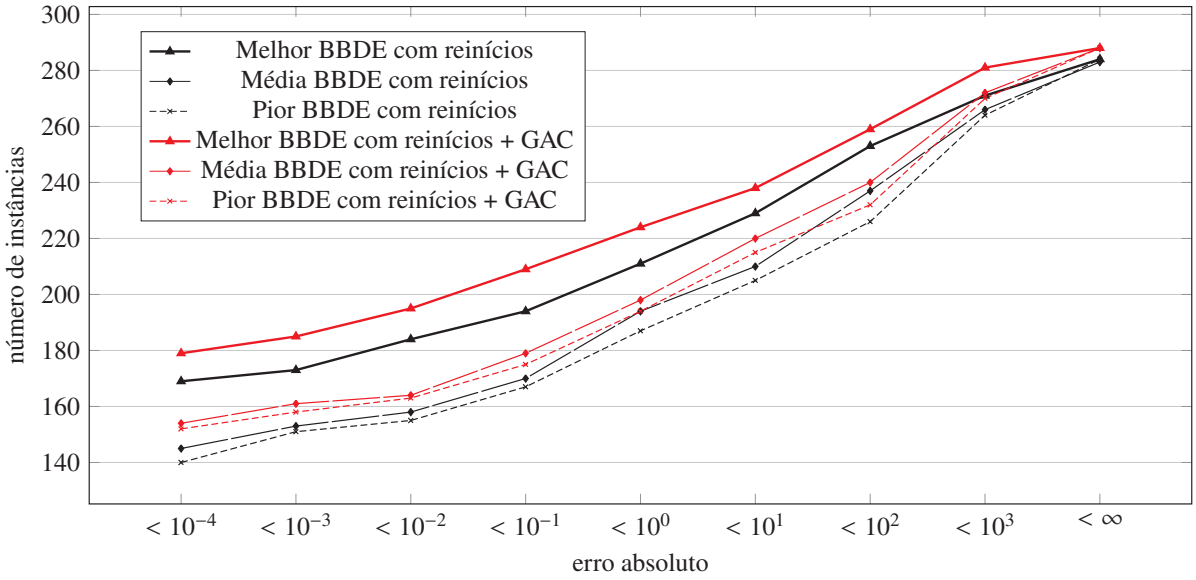



Figura 4.6: Número acumulado de instâncias com soluções factíveis obtidas por BBDE com reinícios e BBDE com reinícios + GAC dados alguns limites de erro absoluto.

Tabela 4.5: Classificação do BBDE, do BBDE com reinícios e de suas versões com o contrator GAC de acordo com o método do CEC2020.

| otimizador | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| **BBDE com reinícios + GAC** | **0,1601** | 0,2848 | **0,2553** | **0,2415** | **1º** |
| BBDE + GAC | 0,3175 | **0,2602** | 0,2672 | 0,2788 | 2º |
| BBDE com reinícios | 0,2128 | 0,3403 | 0,2955 | 0,2931 | 3º |
| BBDE | 0,3285 | 0,2906 | 0,2956 | 0,3029 | 4º |

A Tabela 4.5 e as Figuras 4.5 e 4.6 trazem uma comparação detalhada entre a melhor, a média e a mediana das 15 execuções de cada otimizador. Com esses dados, torna-se mais evidente a melhoria na qualidade das soluções, especialmente de média e mediana, o que indica que ambos os otimizadores foram superados por suas versões com aplicação da etapa de poda pelo contrator GAC. Ademais, ambas as versões de BBDE com reinícios foram superiores aos resultados obtidos por BBDE + GAC, o que indica que a estratégia de reinício possui melhor desempenho diante das instâncias abordadas.

Outro aspecto importante a ser considerado é a taxa de contração média obtida por GAC sobre os domínios das instâncias. No caso do BBDE + GAC houve uma contração média de 0,014% contra os 4,94% obtidos pelo BBDE com reinícios + GAC. Acredita-se que essa diferença nas porcentagens de contração se deva à maior quantidade de soluções factíveis encontradas pela versão com reinícios, o que contribui para o descarte de regiões sub-ótimas do espaço de busca.

Diante dos resultados alcançados na análise experimental, acredita-se que a incorporação de heurísticas provenientes de diversos resolvedores recentes, juntamente com a aplicação da etapa de contração por GAC, tornaram o desempenho do BBDE comparável aos otimizadores HIDE e HIBB que serão apresentados na Seção 4.3 e no Capítulo 5, respectivamente.

## 4.3 HIDE: UMA EVOLUÇÃO DIFERENCIAL INTERVALAR HÍBRIDA

Em Cassenote (2019) e Cassenote et al. (2019) foi publicado o InDE (*Interval Differential Evolution*), a primeira versão de uma Evolução Diferencial (DE) baseada em intervalos para

abordagem de problemas de otimização numérica com restrições. Posteriormente, em Cassenote et al. (2021) foi publicado o I2DE (*Improved Interval Differential Evolution*), uma evolução do InDE que reformula as estratégias de mutação intervalares e adiciona uma série de estratégias de meta-heurísticas recentes adaptadas para o contexto intervalar. Os otimizadores InDE e I2DE são detalhados na Seção 3.2. Por fim, em Cassenote et al. (2024) foi apresentado o HIDE (*Hybrid Interval Differential Evolution*) como uma versão aprimorada do InDE e do I2DE na qual é adicionado o BBDE (Seção 4.2) sem os esquemas de reinício e de consistência local. Esses otimizadores exploram o espaço de busca gerando caixas de maneira estocástica, de forma que qualquer ponto dentro de uma caixa é uma solução candidata.

No HIDE, O processo de busca se inicia com um número predefinido de caixas geradas pela bissecção recursiva do espaço de busca, o que garante sua cobertura total. Inspirado por esquemas utilizados em algoritmos de B&B, o HIDE repetidamente aplica três processos: poda, busca local e exploração. A execução termina quando um número predefinido de soluções candidatas avaliadas é atingido.

A etapa de poda envolve a aplicação do contrator GAC sobre uma caixa que representa um subdomínio da instância a fim de descartar regiões sub-ótimas e infactíveis. É importante notar que essa etapa é aplicada sobre a representação ternária da instância, enquanto os demais estágios do processo de busca são aplicados sobre a modelagem original codificada como uma função caixa-fechada.

Em seguida, uma etapa de busca local é executada na caixa pelo BBDE sem reinícios e sem consistência local descrito na Seção 4.2. O melhor ponto encontrado dentro da caixa é adotado como seu representante, assim como seus valores de função objetivo e de violação de restrições. Com isso, é possível definir quais caixas são mais promissoras e devem ser melhor exploradas. Além disso, os pontos factíveis encontrados pelo BBDE são utilizados para redução do limite superior da função objetivo da melhor solução conhecida, o que tende a acelerar o processo de poda executado por GAC.

A fim de explorar o domínio da instância de forma mais ampla, é utilizada uma versão de DE que opera sobre uma população de caixas, diferente de meta-heurísticas comuns que operam sobre pontos do espaço de busca. Esse DE substitui os operadores de mutação e cruzamento clássicos (definidos na Seção 2.2.1) por suas versões intervalares. Cabe salientar que esses operadores intervalares foram publicados em Cassenote et al. (2021) e são uma contribuição desta tese.

Uma vez que um intervalo $X = [\underline{x},\overline{x}]$ pode ser definido por seu comprimento $\omega(X) = \overline{x}-\underline{x}$ e ponto médio $\mu(X) = (\underline{x}+\overline{x})/2$, operadores de mutação podem ser aplicados ao ponto médio e estendidos ao comprimento dos intervalos. A versão intervalar do operador DE/rand/1 (Equação 2.2 da Seção 2.2.1) combina as caixas $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$ para gerar a caixa candidata $\mathtt{v}_i$. A $j$-ésima dimensão de $\mathtt{v}_i$ é definida por:

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

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

Os outros dois operadores de mutação descritos na Seção 2.2.1 são definidos de maneira similar. A versão intervalar de DE/current-to-rand/1 (Equação 2.3) combina a caixa $\mathtt{x}_i$, da população atual com outras três caixas aleatoriamente selecionadas $\mathtt{r}_1$, $\mathtt{r}_2$ e $\mathtt{r}_3$. A nova caixa é definida por:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + w \cdot \left(\mu(\mathtt{r}_{1j}) - \mu(\mathtt{x}_{ij})\right) + F \cdot \left(\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j})\right), \tag{4.2}$$

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

em que $w$ é um número aleatório entre 0 e 1. Essa estratégia não é combinada com o operador de cruzamento.

Por fim, a versão intervalar de DE/current-to-$p$best/1 (Equação 2.4) utiliza a caixa $\mathtt{x}_i$ e as caixas aleatoriamente selecionadas $\mathtt{r}_1$ e $\mathtt{r}_2$, enquanto $\mathtt{r}_p$ é selecionado entre as $p$ melhores caixas da população:

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

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

Da mesma forma que no BBDE, os valores dos parâmetros $F$ e $CR$ são automaticamente definidos de acordo com as heurísticas propostas por Brest et al. (2016, 2017) e Tanabe e Fukunaga (2014). O operador de cruzamento é uma versão intervalar daquele definido na Equação 2.6. O esquema de subdivisão da população entre $A$ e $B$ de tamanho $NP/2$ utilizado no BBDE é mantido, assim como a aplicação adaptativa de três estratégias de mutação (Equações 4.1 a 4.3) e de redução linear do tamanho da população ao longo da execução.

Após a criação de cada nova caixa, o processo de contração por GAC é aplicado. Caso o melhor ponto da caixa anterior esteja contido na nova caixa, ele é incluído como um dos pontos inicias da operação de busca local executada pelo BBDE. Os valores da função objetivo e da violação de restrições obtidos pela busca local são utilizados para seleção das caixas mais promissoras pelo método $\varepsilon$-*constrained* (Seção 2.2.2).

A avaliação experimental realizada inclui um orçamento geral de $MaxFEs = 10^7 \times D$, em que $D$ é o número de dimensões da instância. O tamanho da população de caixas iniciais foi empiricamente definido como $NP = 5 \times D$. A cada busca local executada pelo BBDE são avaliados $2 \times 10^3 \times D$ pontos candidatos a representantes da caixa, que são mantidos em uma população de tamanho inicial $5 \times D$. Os demais parâmetros seguem os mesmos ajustes apresentados na Seção 4.2 para o BBDE.

Os primeiros aspectos a serem analisados são a qualidade das soluções obtidas pelo HIDE e o impacto da inclusão do contrator GAC como um nível intermediário entre a manipulação de caixas intervalares e a etapa de busca local. Tendo em vista que informações estruturais da instância são exploradas somente durante a etapa de poda, pode-se dizer que, sem o contrator GAC, o HIDE é um otimizador meta-heurístico caixa-fechada.

A Tabela 4.6 mostra o número de instâncias com soluções factíveis e ótimas, além do tempo médio de execução obtidos pelo HIDE e por sua versão com o contrator GAC. Em relação ao número de soluções factíveis, houve um aumento de 23 instâncias distribuídas nas faixas de 1 a 60 dimensões. Já em relação à quantidade de soluções ótimas, houve um acréscimo de 34 instâncias distribuídas nas faixas de 1 a 30 dimensões. Em relação ao orçamento médio gasto até a melhor solução encontrada, tem-se 38,15% no HIDE e 45,97% no HIDE + GAC.

Na Figura 4.7 é exibido o número acumulado de instâncias com soluções factíveis dados alguns limites de erro absoluto. As linhas representam a melhor, a média e a pior das 15 execuções do HIDE e do HIDE + GAC. Dada a melhoria observada na qualidade das soluções, acredita-se que esses valores indiquem que a versão com GAC é mais robusta ao escapar de regiões sub-ótimas e infactíveis do espaço de busca, o que faz com que as soluções candidatas continuem evoluindo por mais tempo.

A Figura 4.8 traz alguns dados relacionados à adição de GAC no processo de busca. As instâncias mostradas na figura estão ordenadas pelo número de dimensões e, aquelas com a

Tabela 4.6: Quantidade de instâncias com soluções factíveis e ótimas e tempo médio de execução obtidos pelo HIDE com e sem a aplicação de filtragem de domínio pelo contrator GAC.

| | | HIDE | | | HIDE + GAC | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 180 | 135 | 26,3 | 186 | 158 | 27,1 |
| 11 a 20 | 59 | 47 | 21 | 95,7 | 54 | 32 | 95,5 |
| 21 a 30 | 16 | 12 | 3 | 241,8 | 13 | 4 | 245,0 |
| 31 a 40 | 13 | 8 | 0 | 305,5 | 11 | 0 | 304,6 |
| 41 a 50 | 19 | 5 | 0 | 669,8 | 8 | 0 | 674,1 |
| 51 a 60 | 9 | 2 | 1 | 692,6 | 5 | 1 | 681,2 |
| 61 a 70 | 8 | 2 | 0 | 995,2 | 2 | 0 | 975,9 |
| 71 a 75 | 5 | 1 | 0 | 2237,3 | 1 | 0 | 2207,4 |
| total | 317 | 257 | 161 | 178,4 | 280 | 195 | 177,9 |

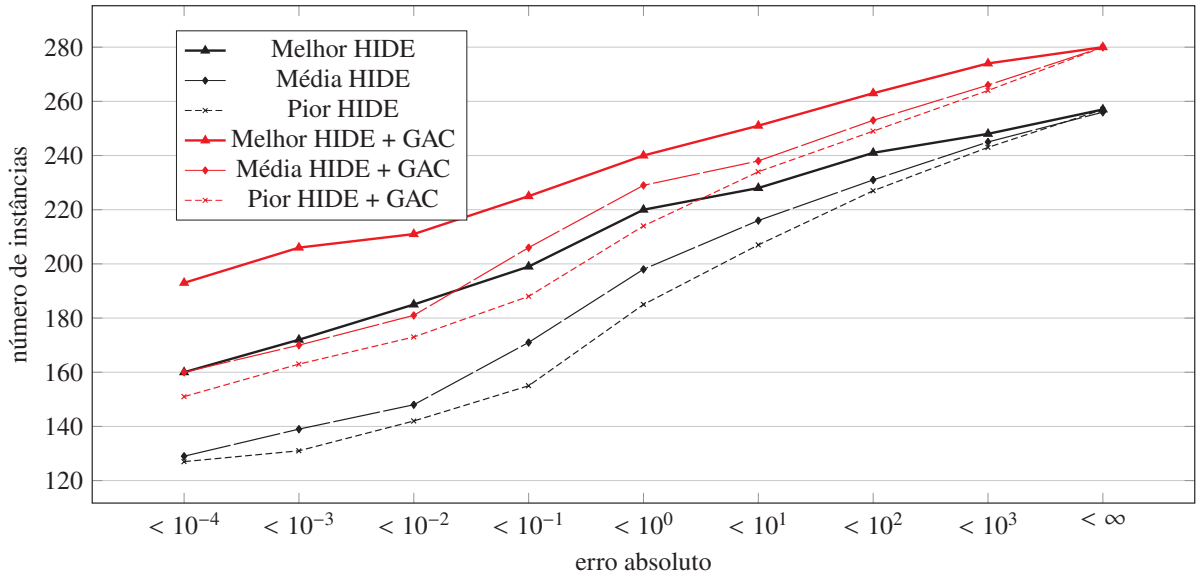



Figura 4.7: Número acumulado de instâncias com soluções factíveis obtidas por HIDE e HIDE + GAC dados alguns limites de erro absoluto.

mesma dimensionalidade, classificadas pela contração média geral. Ao longo da execução do HIDE + GAC, foi obtida uma média de 8,50% de caixas descartadas do processo de busca por serem consideradas inconsistentes durante a aplicação do contrator GAC. Analisando somente as caixas que não foram descartadas, foi obtida uma contração média por GAC de 5,25%. Se forem consideradas também as caixas descartadas, a contração média geral sobe para 12,88%. Essa poda do espaço de busca é útil para evitar o desperdício de recursos computacionais em regiões pouco promissoras. No entanto, cabe salientar que devido à sobreposição das caixas durante o processo estocástico, essas porcentagens superestimam a parcela do domínio da instância que foi descartado por ser considerado inconsistente.

A melhoria no desempenho do HIDE e os valores de contração média por GAC indicam que o HIDE + GAC é um otimizador híbrido com desempenho promissor. Contudo, sabe-se que não existem garantias de que as caixas resultantes das operações intervalares de mutação e cruzamento reduzam durante o processo de busca, visto que a contração por GAC está diretamente relacionada à modelagem da instância. Pela observação da Figura 4.8 fica clara a variação na taxa de contração média obtida por GAC em cada instância. Na prática, quanto maiores forem

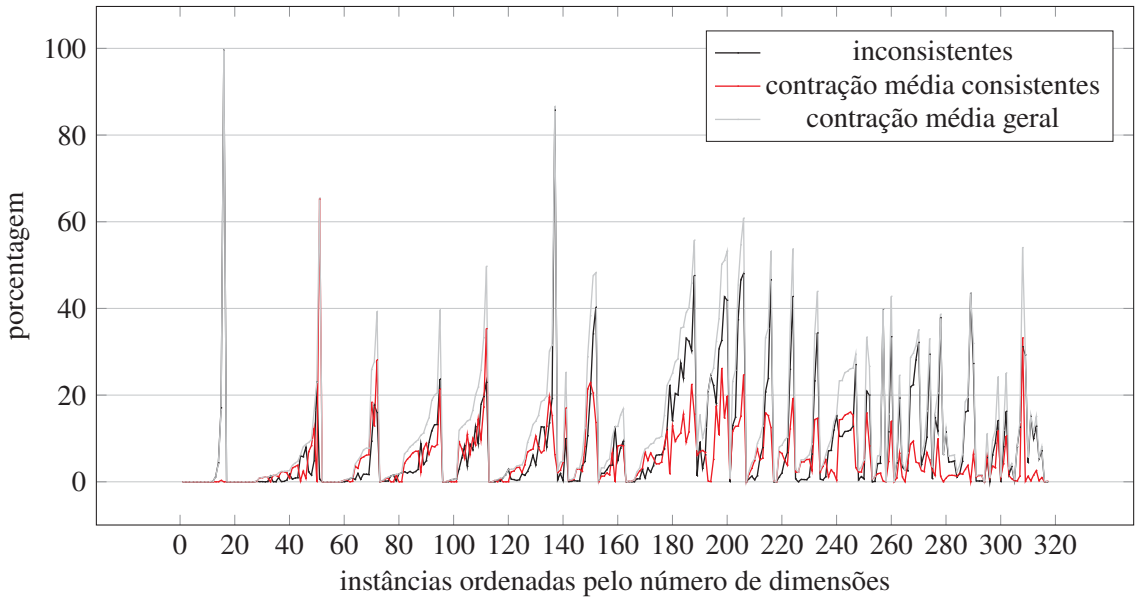



Figura 4.8: Porcentagem de caixas inconsistentes e taxas de contração obtidas por HIDE + GAC em cada instância.

as caixas mantidas na população, maior a probabilidade de haver sobreposição entre elas e, consequentemente, de haver desperdício de recursos ao explorar diversas vezes regiões que já foram anteriormente avaliadas como sub-ótimas.

Visando a redução do tamanho das caixas geradas pelas operações intervalares de mutação e cruzamento, foi implementado um contrator de natureza estocástica. A *$\varepsilon$-contração* (EPSCONT) é gerada a partir da menor sub-caixa que contenha todas as soluções $\varepsilon$-factíveis encontradas em uma caixa ao longo da execução da etapa de busca local, sendo o valor $\varepsilon$ uma tolerância utilizada para comparação entre soluções candidatas e ajustada conforme o método *$\varepsilon$-constrained* definido na Seção 2.2.2 deste documento.

Tabela 4.7: Quantidade de instâncias com soluções factíveis e ótimas e tempo médio de execução obtidos por HIDE + GAC e por HIDE + GAC + EPSCONT.

| | | HIDE + GAC | | | HIDE + GAC + EPSCONT | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 186 | 158 | 27,1 | 187 | 156 | 27,1 |
| 11 a 20 | 59 | 54 | 32 | 95,5 | 56 | 24 | 95,8 |
| 21 a 30 | 16 | 13 | 4 | 245,0 | 13 | 3 | 245,4 |
| 31 a 40 | 13 | 11 | 0 | 304,6 | 7 | 0 | 304,8 |
| 41 a 50 | 19 | 8 | 0 | 674,1 | 8 | 0 | 672,2 |
| 51 a 60 | 9 | 5 | 1 | 681,2 | 3 | 1 | 682,9 |
| 61 a 70 | 8 | 2 | 0 | 975,9 | 2 | 0 | 968,8 |
| 71 a 75 | 5 | 1 | 0 | 2207,4 | 1 | 0 | 2212,1 |
| total | 317 | 280 | 195 | 177,9 | 277 | 184 | 177,8 |

A Tabela 4.7 mostra a quantidade de instâncias para as quais foram obtidas soluções factíveis, ótimas e o tempo médio de execução do HIDE + GAC (dados replicados da Tabela 4.6 para melhor visualização) e do HIDE + GAC + EPSCONT. Apesar de o tempo médio de execução não ter apresentado uma grande variação, é possível observar um declínio considerável na qualidade das soluções encontradas (que se confirma pela observação da Figura 4.9), com a perda

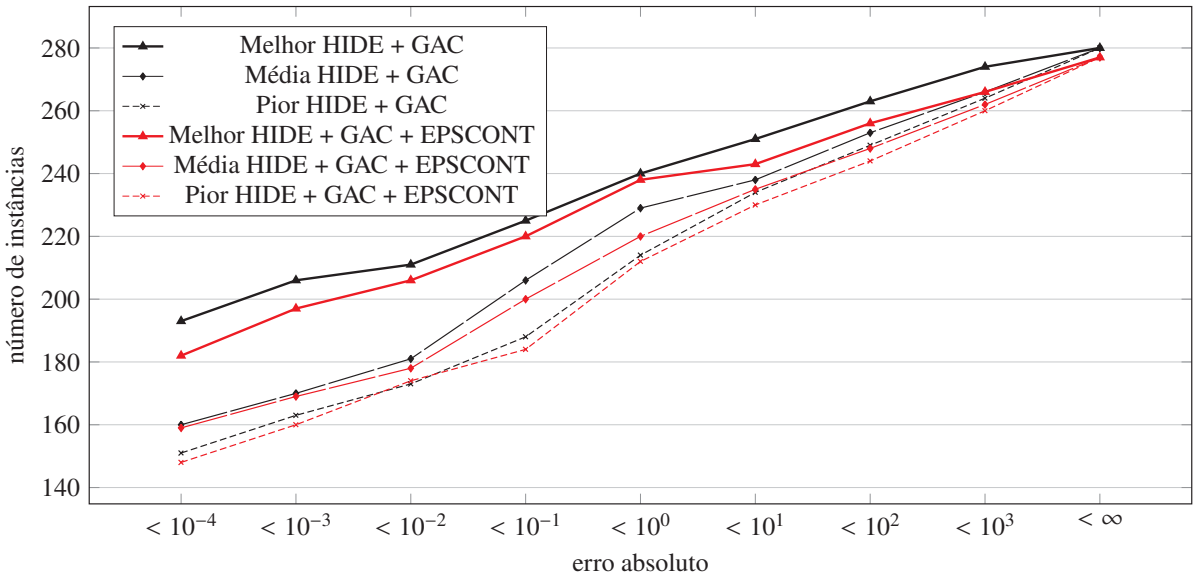



Figura 4.9: Número acumulado de instâncias com soluções factíveis obtidas por HIDE + GAC e HIDE + GAC + EPSCONT dados alguns limites de erro absoluto.

de 3 instâncias com soluções factíveis e 11 instâncias com soluções ótimas. O orçamento médio gasto até a melhor solução encontrada é muito próximo ao HIDE + GAC, com 46,70% contra os 45,97% da versão anterior.

Acerca dos dados relacionados à execução do contrator GAC, observou-se que a porcentagem média de caixas consideradas inconsistentes aumentou, com 9,66% contra os 8,50% anteriores. As porcentagens médias de contração em caixas consistentes e de contração geral também apresentaram acréscimos, passando de 5,25% para 5,58% e de 12,88% para 14,21%, respectivamente. Já a porcentagem média de contração das caixas pela aplicação de EPSCONT ficou em 2,15%.

Tabela 4.8: Classificação de diferentes versões de HIDE e de BBDE com reinícios + GAC de acordo com o método do CEC2020.

| otimizador | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| **HIDE + GAC** | **0,0947** | **0,1327** | **0,1139** | **0,1176** | **1º** |
| HIDE + GAC + EPSCONT | 0,1331 | 0,1697 | 0,1471 | 0,1542 | 2º |
| BBDE com reinícios + GAC | 0,2152 | 0,2522 | 0,2453 | 0,2397 | 3º |
| HIDE | 0,2919 | 0,3539 | 0,3242 | 0,3294 | 4º |

Na Tabela 4.8 são consideradas a melhor, a média e a mediana das soluções encontradas pelas diferentes versões de HIDE e pelo BBDE com reinícios + GAC (a melhor versão de BBDE apresentada na Seção 4.2). Além do evidente ganho em termos de qualidade da solução pela adição do contrator GAC, o HIDE + GAC também se destaca como uma meta-heurística baseada em intervalos com desempenho superior ao BBDE com reinícios + GAC, uma versão clássica que opera somente sobre pontos do espaço de busca.

A pesar de apresentar resultados superiores aos de uma meta-heurística não-intervalar, a sobreposição das caixas geradas pelas operações de mutação intervalares durante o processo de busca do HIDE é uma questão a ser analisada. Por um lado, ter caixas sobrepostas aumenta a probabilidade de se encontrar boas soluções, tendo em vista que a natureza estocástica de meta-heurísticas não garante cobertura total do espaço de busca. No entanto, essa sobreposição também pode causar o desperdício de recursos computacionais na medida em que regiões já descartadas

por serem consideradas pouco promissoras são novamente exploradas no processamento de outras caixas.

O contrator estocástico EPSCONT foi proposto com o intuito de reduzir essa sobreposição. Contudo, os experimentos realizados mostram que a porcentagem de contração média foi pequena e que a qualidade das soluções encontradas se deteriorou significativamente. Observou-se que nos primeiros estágios do processo de busca a contração por EPSCONT é maior, tendo em vista que as caixas exploradas são grandes e a quantidade de pontos $\varepsilon$-factíveis descobertos dentro delas é pequeno. Contudo, conforme o processo de busca avança, a porcentagem de contração média reduz, visto que boa parte das regiões infactíveis já foram descartadas pelo contrator GAC e pelo próprio processo evolutivo da meta-heurística. Com isso, acredita-se que a deterioração nos resultados pela adição de EPSCONT esteja relacionada ao descarte de regiões promissoras que ainda não haviam sido devidamente exploradas nos primeiros estágios do processo de busca. Essa é a maior desvantagem de EPSCONT em relação à GAC: ao se contrair uma caixa não há garantias de que a solução ótima foi mantida.

## 4.4 CONSIDERAÇÕES

Este capítulo apresentou uma análise detalhada de duas meta-heurísticas híbridas propostas no contexto desta tese, BBDE e HIDE, além de um conjunto de heurísticas e estratégias desenvolvidas para tornar seus desempenhos mais robustos e eficientes.

Mais especificamente, foi descrito o processo de seleção das 317 instâncias utilizadas na avaliação experimental desta tese. Este conjunto de instâncias permitiu uma análise aprofundada dos otimizadores propostos e da extensão do núcleo multi-intervalar para suportar operações matemáticas adicionais, ampliando significativamente as possibilidades de aplicação dos otimizadores propostos. O método de avaliação adotado seguiu os padrões da competição CEC2020, considerando não apenas a quantidade de soluções ótimas e factíveis, mas também o valor da função objetivo e da violação de restrições da melhor solução, da média e da mediana entre 15 execuções.

O BBDE, originalmente proposto em Cassenote et al. (2021), é uma meta-heurística baseada em DE que opera exclusivamente sobre pontos do espaço de busca. Nesta tese, foi integrado a uma estratégia de reinícios periódicos, contribuindo para a melhoria da qualidade de suas soluções. A hibridização com o contrator GAC para poda de regiões sub-ótimas e infactíveis do espaço de busca também resultou em melhorias significativas, tornando o BBDE com reinícios + GAC competitivo em relação aos outros otimizadores híbridos propostos nesta tese.

Em seguida, foi apresentado o HIDE, uma hibridização em três eixos: o BBDE, uma versão de DE que opera sobre intervalos e o contrator GAC. A análise realizada indica que a inclusão da etapa de contração melhora significativamente a qualidade das soluções, destacando o HIDE + GAC como uma meta-heurística baseada em intervalos com desempenho superior ao BBDE com reinícios + GAC. Entretanto, embora a sobreposição das caixas gerada pelas operações de mutação intervalares no HIDE aumente a probabilidade de encontrar boas soluções, pode desperdiçar recursos computacionais ao explorar repetidamente regiões pouco promissoras. O contrator estocástico EPSCONT foi introduzido com o intuito de mitigar essa sobreposição, mas os experimentos indicaram que ele não foi muito eficaz na contração e piorou a qualidade das soluções.

Considerando as desvantagens relacionadas à sobreposição de caixas e a ineficácia de EPSCONT ao atenuar esse problema, faz-se necessária uma abordagem que explore o espaço de busca de maneira mais sistemática e organizada. No próximo capítulo, será apresentado o

HIBB, uma abordagem híbrida que integra técnicas de consistência e busca local a um método de *Branch & Bound* (B&B).

# 5 HIBB: UM MÉTODO DE *BRANCH & BOUND* INTERVALAR HÍBRIDO

Uma das principais contribuições desta tese é o HIBB: um método de *Branch & Bound* intervalar híbrido que integra técnicas de consistência local, comumente utilizadas em métodos de otimização exatos, e uma busca local estocástica baseada em meta-heurísticas.

Assim como na maior parte dos métodos de *Branch & Bound* (B&B), o HIBB é composto por três etapas repetidamente executadas:

- Poda (*bound*): consiste na aplicação do contrator GAC (Seção 2.1.3) para descartar soluções sub-ótimas do espaço de busca;
- Busca local: baseada no BBDE (Seção 4.2) sem a estratégia de reinícios, é utilizada para avaliar a qualidade das caixas e fornecer soluções factíveis que possam ser utilizadas como limite superior para a solução ótima;
- Ramificação (*branch*): bissecta uma caixa para dar continuidade ao processo de busca.

Nesse esquema, a busca é guiada pelas caixas (subespaços) geradas pelo método de B&B, exploradas de forma a evitar sobreposições. Na prática, o esquema adotado previne que uma região do espaço de busca seja coberta por mais de uma caixa simultaneamente, ao contrário do que ocorre na exploração observada no HIDE (Seção 4.3).

**Algoritmo 3** HIBB: um método de *Branch & Bound* intervalar híbrido

**Entrada:** $\mathbf{X}_0$: caixa inicial, *MaxFEs*: orçamento de soluções candidatas a serem avaliadas
**Saída:** $\tilde{\mathbf{x}}$: melhor solução conhecida, $\tilde{f}$: avaliação da melhor solução conhecida

```
 1: (x̃, f̃) ← (∅, +∞)
 2: Q ← X_0
 3: enquanto (Q ≠ ∅) e (nfes < MaxFEs) faça
 4:     Extrai uma caixa X da fila Q
 5:     Bissecta X em X' e X''
 6:     para i em {X' e X''} faça
 7:         Aplica o contrator GAC sobre i para poda de soluções sub-ótimas
 8:         se i não for considerada inconsistente então
 9:             Executa a busca local BBDE para avaliar a caixa i
10:             Atualiza a melhor solução conhecida (x̃, f̃)
11:             Insere i na fila Q
12:         senão
13:             Elimina i da fila Q
14:         fim se
15:     fim para
16: fim enquanto
```

Para a execução do otimizador HIBB, detalhada no Algoritmo 3, são fornecidos o domínio inicial da instância $\mathbf{X}_0$ e o orçamento *MaxFEs* de soluções candidatas a serem avaliadas. Ao final da execução, é obtida a melhor solução encontrada $\tilde{\mathbf{x}}$ e sua avaliação $\tilde{f}$, composta pelos valores da função objetivo e da violação de restrições.

O processo de busca inicia-se pela extração de uma caixa $\mathbf{X}$ da fila $Q$, que inicialmente contém somente o domínio da instância. Em seguida, a caixa é bissectada (linha 5) e o contrator GAC (linha 7) é aplicado sobre cada caixa resultante, a fim de descartar soluções sub-ótimas

e infactíveis de seus domínios. Caso as caixas não sejam descartadas, o BBDE (linha 9) é executado em busca do melhor ponto contido nas caixas. Se esse melhor ponto for factível e seu valor de função objetivo for menor que $\tilde{f}$, a melhor solução conhecida é atualizada (linha 10). Por fim, as caixas resultantes são adicionadas à fila $Q$ (linha 11).

O HIBB implementado possui dois critérios de parada: a fila se tornar vazia ou o orçamento de soluções candidatas (pontos) a serem avaliadas pelo BBDE ser completamente consumido. Sem o segundo critério, o HIBB se torna um otimizador exato. Isso ocorre porque a meta-heurística usada como busca local impacta apenas a ordenação da fila de caixas a serem exploradas, e o contrator GAC descarta somente valorações que certamente não constituem a solução ótima da instância, salvo por eventuais erros de representação numérica. No entanto, é importante atentar aos recursos de memória e tempo de processamento exigidos pela execução completa do método, especialmente em instâncias com muitas dimensões.

Algumas heurísticas foram adotadas para determinar a ordem de exploração dos subespaços do domínio da instância. A primeira refere-se à dimensão a ser bissectada, definida como aquela cujo intervalo atual possui o maior comprimento relativo ao seu domínio original. Outra heurística refere-se à escolha da caixa a ser extraída da fila $Q$. Considerando a quantidade de caixas a serem mantidas em memória, utilizou-se uma estrutura de dados baseada em *heap* para tornar sua ordenação eficiente. Além disso, o critério de ordenação da fila é definido por um esquema chamado *ordenação por Qualidade do Ponto representante da Caixa* (QPC), em que:

1. Uma caixa representada por um ponto factível é sempre preferida em relação a uma caixa representada por um ponto infactível;

2. Entre duas caixas cujos pontos representantes são factíveis, escolhe-se aquela com menor valor de função objetivo;

3. Entre duas caixas representadas por pontos infactíveis, prioriza-se aquela com menor valor de violação de restrições.

Esse critério de ordenação prioriza caixas cujos pontos representantes possuem qualidade superior, utilizando uma estratégia de busca melhor-primeiro (*best-first*), o que tende a direcionar os recursos computacionais para a exploração da região mais promissora do espaço de busca. No entanto, um problema inerente a esse esquema é a possibilidade de que todo o orçamento seja consumido no processo de refinar a busca em torno da melhor caixa inicialmente encontrada, que não necessariamente contém uma solução ótima para a instância. Para promover a exploração de outras regiões, optou-se por abandonar a busca em caixas cuja maior dimensão seja menor do que 1% do espaço de busca original da instância.

Na Tabela 5.1, nota-se que o HIBB obteve um aumento de 20 soluções factíveis em relação ao HIDE + GAC. Esse ganho em factibilidade é observado inclusive em grupos de instâncias com maior dimensionalidade, indicando que evitar a sobreposição das caixas exploradas pode contribuir para uma exploração mais ampla do espaço de busca. No entanto, a quantidade de soluções ótimas não apresentou o mesmo comportamento, o que pode indicar que o HIDE + GAC apresenta uma convergência mais rápida.

Os resultados obtidos pelos dois resolvedores são mais detalhadamente comparados de acordo com o método do CEC2020 (Tabela 5.2). A maior quantidade de instâncias com soluções factíveis favorece o HIBB em relação à melhor dentre as 15 execuções, mas o HIDE + GAC obteve melhor desempenho sobre a média e a mediana das soluções.

Com a análise dos resultados obtidos pelo HIBB e pelo HIDE + GAC, acredita-se que a estratégia de busca melhor-primeiro (*best-first*) adotada na escolha da próxima caixa a ser

Tabela 5.1: Quantidade de instâncias com soluções factíveis e ótimas e tempo médio de execução obtidos por HIDE + GAC e por HIBB.

| | | HIDE + GAC | | | HIBB | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 186 | 158 | 27,1 | 187 | 152 | 26,5 |
| 11 a 20 | 59 | 54 | 32 | 95,5 | 56 | 36 | 94,0 |
| 21 a 30 | 16 | 13 | 4 | 245,0 | 13 | 5 | 242,7 |
| 31 a 40 | 13 | 11 | 0 | 304,6 | 12 | 0 | 303,2 |
| 41 a 50 | 19 | 8 | 0 | 674,1 | 14 | 0 | 665,4 |
| 51 a 60 | 9 | 5 | 1 | 681,2 | 9 | 1 | 673,3 |
| 61 a 70 | 8 | 2 | 0 | 975,9 | 5 | 0 | 954,1 |
| 71 a 75 | 5 | 1 | 0 | 2207,4 | 4 | 0 | 1836,8 |
| total | 317 | 280 | 195 | 177,9 | 300 | 194 | 169,9 |

Tabela 5.2: Classificação do HIDE + GAC e do HIBB de acordo com o método do CEC2020.

| otimizador | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| **HIDE + GAC** | 0,2145 | **0,2555** | 0,2303 | **0,2382** | **1º** |
| HIBB | **0,1735** | 0,3155 | **0,2461** | 0,2590 | 2º |

explorada pode ser excessivamente elitista, considerando o orçamento definido, o que inviabiliza uma exploração mais ampla do espaço de busca. Outra possibilidade é que a estratégia de abandonar o processo de busca em caixas menores que 1% do tamanho do domínio original não esteja adequadamente parametrizada, sendo dependente da instância abordada. Uma alternativa para reduzir o elitismo dessa estratégia é adotar a estratégia *MaxDist*, proposta por Vanaret et al. (2015), na qual a fila $Q$ prioriza as caixas que estejam localizadas o mais longe possível da melhor solução atual no espaço de busca. Essa estratégia foi experimentalmente avaliada e apresentou resultados significativamente inferiores em relação à estratégia melhor-primeiro.

A partir dessas conjecturas, na seção a seguir são propostas algumas heurísticas que influenciam na forma com que o espaço de busca é explorado e, consequentemente, na velocidade de convergência do HIBB.

## 5.1 HEURÍSTICAS PARA EXPLORAÇÃO DO ESPAÇO DE BUSCA

A fim de mitigar o elitismo inerente à estratégia de ordenação da fila $Q$ melhor-primeiro, propõe-se um esquema que mantém *NB* frentes de busca simultâneas. Essa abordagem é inspirada por práticas observadas em meta-heurísticas populacionais, como BBDE e HIDE, que exploram uma população de *NP* soluções candidatas ao mesmo tempo, e pelos algoritmos de busca em feixe (*beam search*), que utilizam múltiplas frentes de busca. Essa estratégia visa diversificar a exploração, reduzindo a probabilidade de convergência prematura para soluções sub-ótimas e visando distribuir os recursos computacionais de maneira mais equilibrada.

No contexto do HIBB, a ordenação da fila de prioridade é atualizada periodicamente a cada *NB* ciclos. Este intervalo de atualização permite que caixas menos promissoras também sejam exploradas. Inicialmente, foi adotada uma estratégia onde o valor de *NB* é definido de maneira proporcional ao número de dimensões da instância, visando uma exploração abrangente do espaço de busca. À medida em que a execução avança, *NB* é linearmente reduzido, direcionando os recursos computacionais para regiões do espaço de busca melhor avaliadas.

Tabela 5.3: Quantidade de instâncias com soluções factíveis e ótimas e tempo médio de execução obtidos por HIBB com múltiplas frentes de busca e ordenação da fila de prioridade pela qualidade da caixa (QPC).

| | | QPC com *NB* inicial = $0{,}5 \times D$ | | | QPC com *NB* inicial = $D$ | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 187 | 154 | 26,7 | 187 | 155 | 26,5 |
| 11 a 20 | 59 | 57 | 34 | 94,8 | 56 | 36 | 94,9 |
| 21 a 30 | 16 | 13 | 6 | 245,2 | 13 | 6 | 244,6 |
| 31 a 40 | 13 | 12 | 0 | 305,1 | 12 | 0 | 304,7 |
| 41 a 50 | 19 | 15 | 0 | 669,7 | 14 | 0 | 668,5 |
| 51 a 60 | 9 | 8 | 1 | 678,9 | 8 | 1 | 679,2 |
| 61 a 70 | 8 | 4 | 0 | 964,8 | 3 | 0 | 962,6 |
| 71 a 75 | 5 | 2 | 0 | 1828,5 | 3 | 0 | 1817,9 |
| total | 317 | 298 | 195 | 171,0 | 296 | 198 | 170,6 |
| | | QPC com *NB* inicial = $1{,}5 \times D$ | | | QPC com *NB* inicial = $2 \times D$ | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 187 | 154 | 26,5 | 187 | 155 | 26,4 |
| 11 a 20 | 59 | 56 | 35 | 94,7 | 56 | 32 | 94,7 |
| 21 a 30 | 16 | 13 | 5 | 244,7 | 13 | 5 | 244,6 |
| 31 a 40 | 13 | 12 | 0 | 304,1 | 12 | 0 | 304,1 |
| 41 a 50 | 19 | 15 | 0 | 669,8 | 15 | 0 | 669,1 |
| 51 a 60 | 9 | 7 | 1 | 679,8 | 8 | 1 | 678,2 |
| 61 a 70 | 8 | 3 | 0 | 962,9 | 3 | 0 | 960,9 |
| 71 a 75 | 5 | 2 | 0 | 1824,0 | 2 | 0 | 1826,9 |
| total | 317 | 295 | 195 | 170,7 | 296 | 193 | 170,5 |

Tabela 5.4: Classificação de diferentes versões do HIBB com ordenação pela qualidade da caixa (QPC) de acordo com o método do CEC2020.

| otimizador | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| ***NB* inicial =** $0{,}5 \times D$ | **0,1276** | **0,1925** | **0,1512** | **0,1648** | **1º** |
| *NB* inicial = $1{,}5 \times D$ | 0,1451 | 0,1777 | 0,1760 | 0,1676 | 2º |
| *NB* inicial = $D$ | 0,1673 | 0,1770 | 0,1536 | 0,1694 | 3º |
| *NB* inicial = $2 \times D$ | 0,1937 | 0,1872 | 0,1860 | 0,1889 | 4º |
| *NB* = 1 | 0,1696 | 0,3712 | 0,2529 | 0,2871 | 5º |

A Tabela 5.3 demonstra a variação na quantidade de instâncias para as quais o HIBB encontrou soluções factíveis e ótimas, com *NB* inicial = $0{,}5 \times D$ se destacando com a maior quantidade de soluções factíveis e *NB* inicial = $D$ com a maior quantidade de soluções ótimas. O tempo médio de execução teve pouca variação entre as configurações de parâmetros testadas. A análise da classificação na Tabela 5.4 indica que a adoção de múltiplas frentes de busca aprimora significativamente a qualidade das soluções obtidas, especialmente em relação à média e à mediana das 15 execuções. A primeira versão do HIBB, cujos resultados foram apresentados na Tabela 5.1, é listada como *NB* = 1. Todas as configurações com *NB* dinâmico superaram aquela com *NB* = 1. De acordo com o método do CEC2020, os melhores resultados foram obtidos por *NB* inicial = $0{,}5 \times D$ devido à sua maior quantidade de instâncias com soluções factíveis.

Além das múltiplas frentes de busca, também foi proposto um novo esquema para a ordenação da fila de caixas $Q$, chamado de *ordenação por Estagnação e Qualidade do Ponto representante da Caixa* (EQPC). Conforme o nome sugere, além da qualidade da solução, nesse

esquema considera-se o número de ciclos em que uma região do espaço de busca permanece estagnada. Especificamente, quando uma caixa é bissectada e a busca local é aplicada, se as caixas-filhas $\mathbf{X}'$ e $\mathbf{X}''$ não contiverem um ponto representante de qualidade superior ao da caixa-mãe $\mathbf{X}$, conta-se um ciclo de estagnação para essas caixas. Caso contrário, a contagem de ciclos de estagnação da caixa é zerada. Na ordenação da fila $Q$, adota-se uma abordagem em duas etapas: inicialmente, são priorizadas as caixas com o menor número de ciclos de estagnação; em seguida, entre as caixas com o mesmo número de ciclos de estagnação, são preferidas aquelas com melhor qualidade.

A combinação da estratégia de ordenação em duas etapas na fila de prioridade com a manutenção de múltiplas frentes de busca torna dispensável o esquema de abortar a exploração de caixas muito pequenas, como aquelas menores que 1% do domínio original. As duas heurísticas propostas permitem que, mesmo que uma frente de busca esteja focada em uma caixa pequena, outras possam explorar caixas de diferentes tamanhos e regiões do espaço de busca. Isso reduz a dependência de heurísticas que consideram o tamanho das caixas e a necessidade de ajustar parâmetros que podem variar conforme a instância em questão.

A Tabela 5.5 apresenta uma análise da quantidade de instâncias com soluções factíveis e ótimas, bem como o tempo médio de execução do HIBB utilizando a estratégia de ordenação por qualidade do ponto da caixa (QPC, com $NB = 1$) e de ordenação por estagnação + qualidade do ponto da caixa (EQPC) com diversas configurações de *NB* inicial. A maior quantidade de instâncias com soluções factíveis foi obtida por $NB = 1$, enquanto a maior quantidade de soluções ótimas foi apresentada por *NB* inicial = $D$.

A análise pelo método do CEC2020 (Tabela 5.6) indica um comportamento semelhante ao observado na ordenação por qualidade do ponto da caixa (QPC – Tabela 5.4), em que $NB = 1$ apresenta desempenho razoável em relação à melhor das 15 execuções, mas se deteriora significativamente quando são analisados os valores de média e mediana. Isso sugere um comportamento mais vulnerável aos elementos estocásticos do HIBB, que acabam sendo decisivos para a qualidade da solução, especialmente quando se trata de uma execução com definição de orçamento a ser consumido.

Essas impressões são corroboradas pela análise apresentada na Tabela 5.7, na qual é realizada uma classificação geral com todas as heurísticas propostas nesta seção. Todas as configurações com *NB* inicial definido em relação ao número de dimensões da instância apresentam resultados superiores aos obtidos pela versão com uma única frente de busca fixa. Esses resultados reforçam a ideia de que manter múltiplas frentes de busca no HIBB contribui para uma melhor alocação dos recursos computacionais disponíveis e permite uma exploração mais ampla do espaço de busca.

Além disso, os resultados obtidos por ambos os métodos de ordenação da fila de caixas a serem exploradas foram similares nos testes realizados. Ao garantir que sejam priorizadas caixas não-estagnadas, o método EQPC proporciona uma distribuição dinâmica dos recursos computacionais, permitindo uma exploração efetiva do espaço de busca de maneira semelhante à abordagem QPC, mesmo sem descartar caixas pequenas. Essa capacidade de adaptação pode compensar a ausência da heurística de tamanho, resultando em desempenhos comparáveis entre as duas estratégias de ordenação.

Na próxima seção, são analisados os impactos das heurísticas adicionadas ao HIBB sobre a forma com que o espaço de busca é explorado.

Tabela 5.5: Quantidade de instâncias com soluções factíveis e ótimas e tempo médio de execução obtidos por HIBB com múltiplas frentes de busca *NB* e ordenação da fila de prioridade por qualidade da caixa (QPC) e por estagnação + qualidade da caixa (EQPC).

| | | QPC com $NB = 1$ | | | EQPC com $NB = 1$ | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 187 | 152 | 26,5 | 188 | 151 | 26,6 |
| 11 a 20 | 59 | 56 | 36 | 94,0 | 56 | 36 | 93,9 |
| 21 a 30 | 16 | 13 | 5 | 242,7 | 13 | 7 | 243,6 |
| 31 a 40 | 13 | 12 | 0 | 303,2 | 12 | 0 | 303,3 |
| 41 a 50 | 19 | 14 | 0 | 665,4 | 15 | 0 | 763,7 |
| 51 a 60 | 9 | 9 | 1 | 673,3 | 9 | 1 | 673,0 |
| 61 a 70 | 8 | 5 | 0 | 954,1 | 5 | 0 | 950,6 |
| 71 a 75 | 5 | 4 | 0 | 1836,8 | 4 | 0 | 1852,2 |
| total | 317 | 300 | 194 | 169,9 | 302 | 195 | 176,1 |
| | | EQPC com $NB$ inicial $= 0{,}5 \times D$ | | | EQPC com $NB$ inicial $= D$ | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 187 | 153 | 26,5 | 187 | 155 | 26,6 |
| 11 a 20 | 59 | 57 | 35 | 93,9 | 56 | 37 | 94,9 |
| 21 a 30 | 16 | 13 | 6 | 243,2 | 13 | 6 | 243,9 |
| 31 a 40 | 13 | 12 | 0 | 302,7 | 12 | 0 | 304,3 |
| 41 a 50 | 19 | 15 | 0 | 667,0 | 14 | 0 | 670,9 |
| 51 a 60 | 9 | 8 | 1 | 675,2 | 8 | 1 | 676,3 |
| 61 a 70 | 8 | 4 | 0 | 958,2 | 3 | 0 | 958,4 |
| 71 a 75 | 5 | 2 | 0 | 1817,1 | 3 | 0 | 1819,7 |
| total | 317 | 298 | 195 | 169,8 | 296 | 199 | 170,6 |
| | | EQPC com $NB$ inicial $= 1{,}5 \times D$ | | | EQPC com $NB$ inicial $= 2 \times D$ | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 187 | 154 | 26,6 | 187 | 154 | 26,6 |
| 11 a 20 | 59 | 56 | 35 | 94,4 | 56 | 33 | 95,0 |
| 21 a 30 | 16 | 13 | 5 | 243,5 | 13 | 5 | 244,0 |
| 31 a 40 | 13 | 12 | 0 | 302,2 | 12 | 0 | 303,8 |
| 41 a 50 | 19 | 15 | 0 | 669,6 | 15 | 0 | 669,9 |
| 51 a 60 | 9 | 7 | 1 | 675,7 | 8 | 1 | 675,4 |
| 61 a 70 | 8 | 3 | 0 | 959,6 | 3 | 0 | 959,9 |
| 71 a 75 | 5 | 2 | 0 | 1823,5 | 2 | 0 | 1832,6 |
| total | 317 | 295 | 195 | 170,3 | 296 | 193 | 170,7 |

Tabela 5.6: Classificação de diferentes versões do HIBB com ordenação por estagnação + qualidade da caixa (EQPC) de acordo com o método do CEC2020.

| otimizador | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| ***NB* inicial =** $0{,}5 \times D$ | **0,1376** | **0,1888** | **0,1660** | **0,1689** | **1º** |
| *NB* inicial $= 1{,}5 \times D$ | 0,1528 | 0,1776 | 0,1745 | 0,1696 | 2º |
| *NB* inicial $= D$ | 0,1646 | 0,1851 | 0,1613 | 0,1742 | 3º |
| *NB* inicial $= 2 \times D$ | 0,2072 | 0,1900 | 0,1877 | 0,1947 | 4º |
| $NB = 1$ | 0,1597 | 0,3747 | 0,2547 | 0,2862 | 5º |

Tabela 5.7: Classificação de diferentes versões do HIBB com ordenação por qualidade da caixa (QPC) e por estagnação + qualidade da caixa (EQPC) de acordo com o método do CEC2020.

| Versão de HIBB | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| **QPC com *NB* inicial =** $0{,}5 \times D$ | **0,1270** | **0,1927** | **0,1499** | **0,1644** | **1º** |
| QPC com *NB* inicial = $D$ | 0,1620 | 0,1748 | 0,1515 | 0,1663 | 2º |
| EQPC com *NB* inicial = $0{,}5 \times D$ | 0,1261 | 0,1970 | 0,1624 | 0,1688 | 3º |
| EQPC com *NB* inicial = $1{,}5 \times D$ | 0,1435 | 0,1834 | 0,1712 | 0,1690 | 4º |
| QPC com *NB* inicial = $1{,}5 \times D$ | 0,1436 | 0,1835 | 0,1731 | 0,1694 | 5º |
| EQPC com *NB* inicial = $D$ | 0,1579 | 0,1857 | 0,1546 | 0,1711 | 6º |
| QPC com *NB* inicial = $2 \times D$ | 0,1925 | 0,1920 | 0,1833 | 0,1904 | 7º |
| EQPC com *NB* inicial = $2 \times D$ | 0,1989 | 0,1964 | 0,1870 | 0,1953 | 8º |
| QPC com $NB = 1$ | 0,1669 | 0,3680 | 0,2457 | 0,2832 | 9º |
| EQPC com $NB = 1$ | 0,1571 | 0,3807 | 0,2508 | 0,2877 | 10º |

## 5.2 ANÁLISE DA EXPLORAÇÃO DO ESPAÇO DE BUSCA

A partir da análise da Tabela 5.7, definiu-se o número inicial de frentes de busca do HIBB como $NB = 0{,}5 \times D$ para ambos os critérios de ordenação QPC (ordenação por Qualidade do Ponto representante da Caixa) e EQPC (ordenação por Estagnação e Qualidade do Ponto representante da Caixa). Com isso, é necessário analisar o impacto das heurísticas implementadas sobre a exploração do espaço de busca. A Tabela 5.8 apresenta a média e o desvio padrão (indicado entre parênteses) de alguns dados de execução que serão discutidos nos parágrafos a seguir.

Tabela 5.8: Dados médios das 15 execuções do HIBB com ordenação por qualidade da caixa (QPC) e do HIBB com ordenação por estagnação + qualidade da caixa (EQPC).

| | HIBB com QPC | HIBB com EQPC |
|---|---|---|
| Orçamento total consumido | 95,96% (19,00%) | 98,68% (11,02%) |
| Número de caixas exploradas | 960,48 (189,98) | 987,73 (110,18) |
| Tamanho final da fila $Q$ | 405,66 (127,40) | 468,71 (77,06) |
| Profundidade máxima explorada | 47,74 (30,29) | 73,33 (51,09) |
| Profundidade da melhor solução | 21,97 (27,14) | 23,45 (29,74) |
| Orçamento consumido até a melhor solução | 27,84% (32,62%) | 27,40% (33,07%) |

O primeiro aspecto a ser analisado é o consumo do orçamento de soluções candidatas avaliadas, definido como $MaxFes = 10^7 \times D$. A Tabela 5.8 mostra que a estratégia de ordenação QPC obteve um consumo de orçamento menor que aquele apresentado por EQPC, com média de 95,96% e 98,68%, respectivamente. Esse dado é corroborado pelo número de caixas exploradas, que também foi menor para HIBB com QPC. Uma análise mais detalhada revela que, das 317 instâncias totais, 16 instâncias na ordenação QPC não consumiram todo o orçamento máximo estabelecido, com 14 dessas ficando abaixo de 35%. No caso da EQPC, apenas 5 instâncias não consumiram completamente o orçamento estabelecido, todas ficando abaixo de 30%.

Em ambas as implementações do método HIBB, a única forma de o orçamento não ser totalmente consumido é quando não há mais caixas a serem exploradas na fila, ou seja, ela se torna vazia. O consumo menor de orçamento pela estratégia de ordenação QPC deve-se ao fato de que caixas menores que 1% do domínio original são abandonadas. Como resultado, o HIBB com ordenação QPC tende a explorar menos caixas, consumindo menos orçamento de soluções candidatas avaliadas em comparação à estratégia EQPC, que segue a exploração mesmo quando as caixas se tornam muito pequenas.

O abandono de caixas durante a execução do HIBB com QPC pode ser melhor compreendido através da Figura 5.1, que ilustra a porcentagem média de caixas abandonadas nas 15 execuções de cada instância, resultando em uma média geral de 6,53%. As instâncias estão organizadas primeiramente pelo número de dimensões e, dentro de cada grupo com a mesma dimensionalidade, são classificadas de acordo com a porcentagem de caixas abandonadas. É possível observar uma variação considerável na porcentagem de caixas abandonadas entre instâncias com números de dimensões similares. Por exemplo, a instância 112, com 5 dimensões, apresenta uma média de 66,66% de caixas abandonadas porque somente três caixas foram exploradas ao longo da busca, enquanto a instância 113, com 6 dimensões, não teve nenhuma caixa abandonada. Embora haja essa variação entre instâncias de tamanhos semelhantes, nota-se que a média de caixas abandonadas diminui até a instância 251, que possui 22 dimensões. Isso sugere que, com o orçamento definido e as heurísticas adotadas, em real as caixas exploradas em instâncias com mais de 22 dimensões não alcançaram o tamanho mínimo de 1% do espaço de busca original.

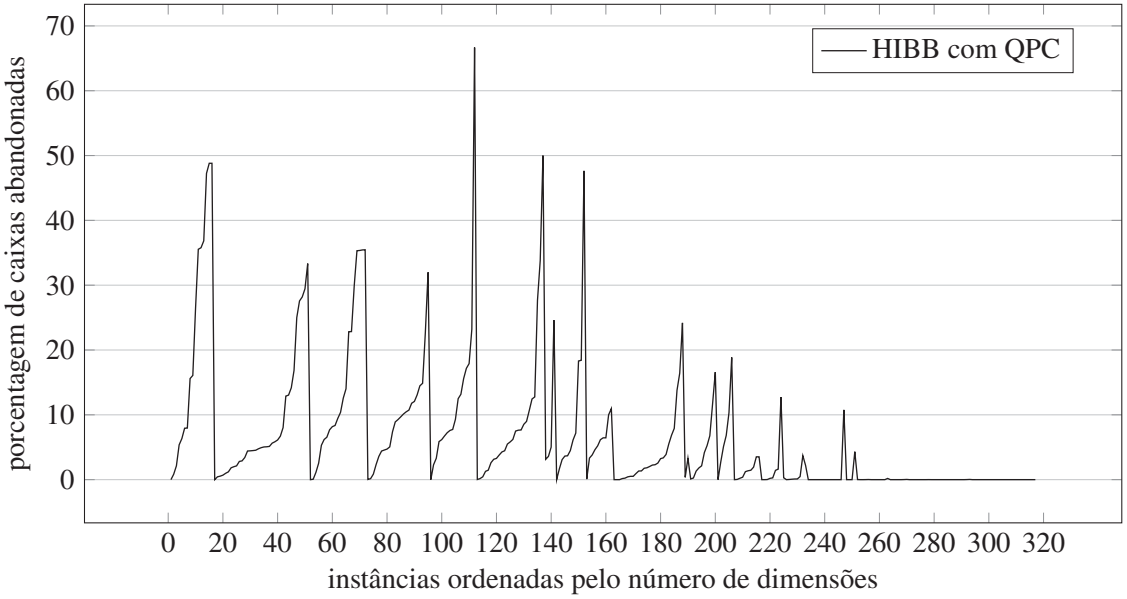



Figura 5.1: Porcentagem média de caixas abandonadas nas 15 execuções de HIBB com QPC por atingirem um tamanho menor que 1% do espaço de busca original.

Considerando que 94,95% e 98,42% das instâncias com ordenação QPC e EQPC, respectivamente, ainda possuíam caixas a serem exploradas na fila $Q$ quando o orçamento foi totalmente consumido, outro aspecto interessante a ser analisado é a quantidade de caixas restantes na fila. A Figura 5.2 emprega o mesmo critério de ordenação das instâncias que a Figura 5.1. O tamanho médio final da fila é de 405,66 para a ordenação QPC e 468,71 para EQPC. Essa variação está relacionada às caixas descartadas pelo HIBB com QPC devido ao seu tamanho reduzido. Essa tendência é clara até a instância 251, que tem 22 dimensões. Após essa instância, os tamanhos médios das filas de ambos os critérios de ordenação convergem. Além disso, é importante notar que o abandono de caixas pequenas não é o único fator que reduz o espaço de busca explorado. A etapa de consistência local, que será analisada mais adiante nesta seção, também influencia significativamente esse processo.

Outro aspecto importante a ser analisado é a profundidade atingida pelas estratégias de ordenação na árvore de busca do B&B. Os dados da Tabela 5.8 mostram que o HIBB com EQPC tem uma profundidade média explorada nas 15 execuções significativamente maior (73,33) do que o HIBB com QPC (47,74). Essa diferença sugere que a estratégia EQPC permite uma exploração

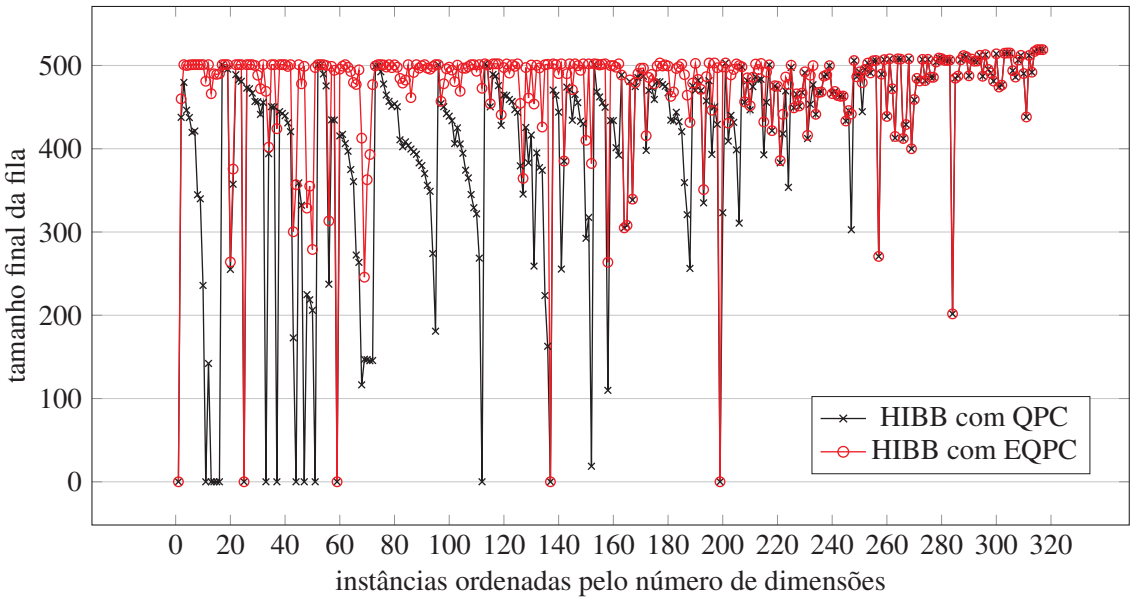



Figura 5.2: Tamanho médio da fila de caixas por instância ao final das 15 execuções de HIBB com QPC e de HIBB com EQPC.

mais profunda do espaço de busca, o que pode ser atribuído não somente à estratégia de abandono de caixas pequenas em QPC, mas também à capacidade do EQPC de evitar a estagnação em regiões menos promissoras. A média de profundidade atingida por instância, observada na parte superior da Figura 5.3, corrobora essa ideia, mostrando uma grande diferença de profundidade entre as estratégias de ordenação, especialmente em instâncias com menor dimensionalidade. No entanto, conforme o número de dimensões aumenta e o abandono de caixas pequenas diminui, a profundidade atingida por ambas as estratégias de ordenação se torna similar.

A Tabela 5.8 também revela que a profundidade média da melhor solução é ligeiramente maior para EQPC (23,45) em comparação com QPC (21,97). Essa similaridade é observada na parte inferior da Figura 5.3, que mostra que a profundidade da melhor solução é bastante próxima entre as duas estratégias na maioria das instâncias. Além disso, o orçamento médio consumido até a melhor solução é similar para ambas as estratégias, com QPC consumindo 27,84% e EQPC 27,40%. Esses dados indicam que, embora a estratégia EQPC explore o espaço de busca de maneira mais profunda, a qualidade e a profundidade das soluções encontradas por ambas as estratégias são comparáveis, conforme demonstrado na Tabela 5.7. Isso sugere que a ordenação QPC também é eficaz em encontrar boas soluções, mesmo com uma exploração menos profunda.

Um dos componentes essenciais do método HIBB é a utilização do contrator GAC (Seção 2.1.3) para podar valorações sub-ótimas e infactíveis do espaço de busca. Essa etapa de poda torna o processo de otimização mais eficiente, permitindo que métodos exatos encontrem soluções para instâncias com um número crescente de dimensões em um tempo razoável. Portanto, é necessário analisar os efeitos do contrator GAC nas diferentes instâncias abordadas pelo HIBB e comparar seus indicadores com os dos otimizadores híbridos meta-heurísticos propostos nesta tese.

A Tabela 5.9 apresenta a porcentagem média de caixas inconsistentes, a média de contração das caixas consistentes e a média de contração geral (caixas consistentes + inconsistentes) obtidas pelas melhores versões dos otimizadores híbridos propostos nesta tese: BBDE com reinícios + GAC, HIDE + GAC e HIBB com as estratégias QPC e EQPC. Ao analisar a porcentagem média de caixas inconsistentes, são observados valores bastante próximos entre HIBB com QPC (5,35%) e HIBB com EQPC (5,41%), apesar de a ordenação por EQPC ter

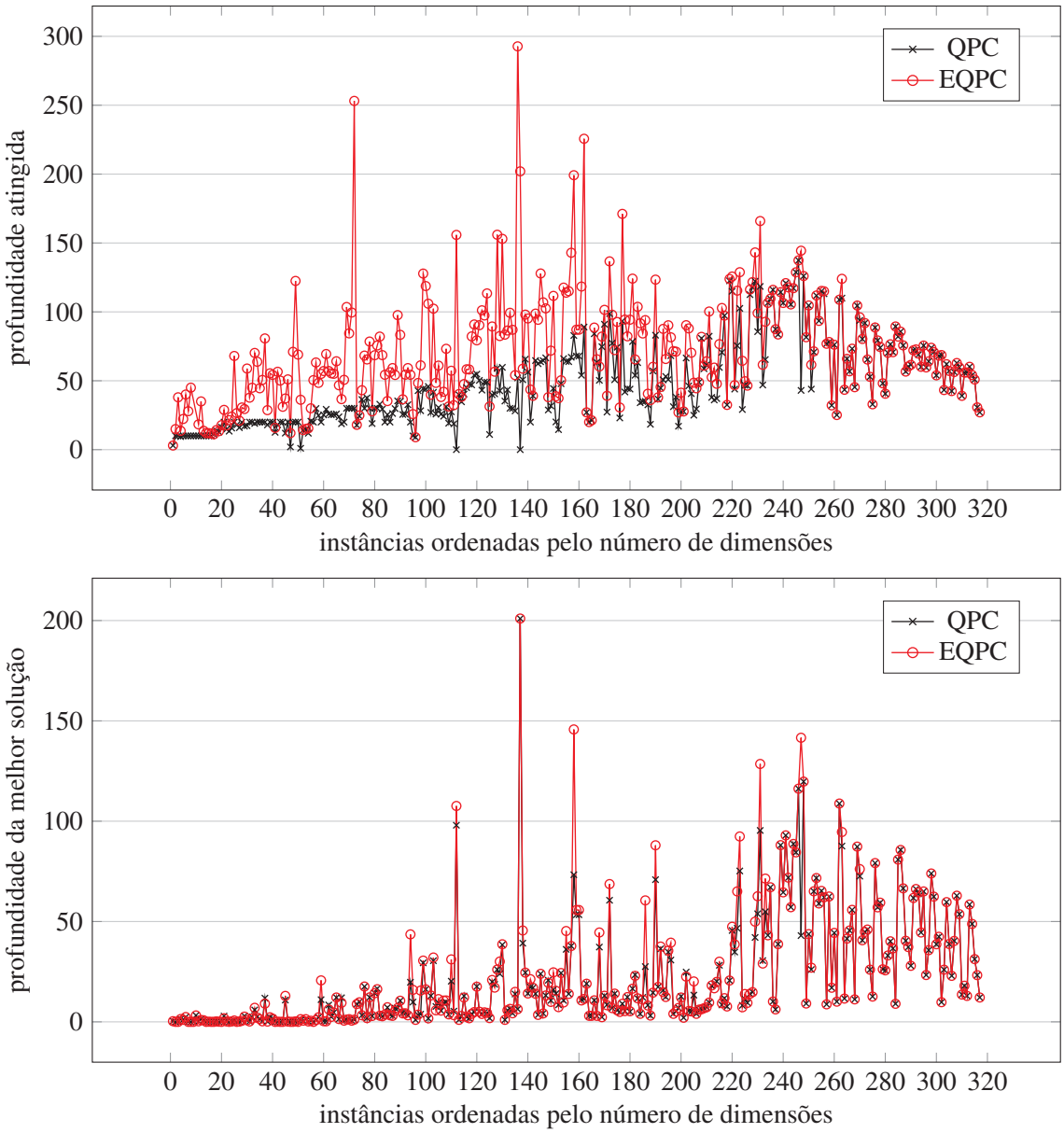



Figura 5.3: Profundidade média explorada e profundidade média da melhor solução encontrada nas 15 execuções de cada instância para HIBB com QPC e HIBB com EQPC.

Tabela 5.9: Média de caixas inconsistentes, contração média de caixas consistentes e contração média geral obtidas nas 15 execuções dos otimizadores híbridos propostos.

| | inconsistentes | contração média consistenstes | contração média geral |
|---|---|---|---|
| BBDE com reinícios + GAC | 0% | 4,94% | 4,94% |
| HIDE + GAC | 8,50% | 5,25% | 12,88% |
| HIBB com QPC | 5,35% | 6,86% | 11,66% |
| HIBB com EQPC | 5,41% | 6,24% | 11,27% |

explorado uma quantidade maior de caixas em relação a QPC (987,73 e 960,48, respectivamente, conforme demonstrado na Tabela 5.8). Essa porcentagem de inconsistências é significativamente menor que a apresentada por HIDE + GAC (8,50%), sugerindo que a exploração sem sobreposição de caixas, baseada em B&B executada pelo HIBB, é mais eficaz em gerar caixas consistentes

do que a exploração estocástica executada pelo HIDE. O BBDE com reinícios + GAC não gera caixas inconsistentes porque seu espaço de busca não é segmentado, ou seja, a exploração é realizada sobre uma única caixa.

A média de contração das caixas consistentes mostra que o HIBB com QPC (6,86%) apresentou uma ligeira superioridade em relação ao EQPC (6,24%) e uma diferença mais significativa quando comparado ao BBDE (4,94%) e ao HIDE (5,25%). Em termos de contração média geral, o HIBB com QPC (11,66%) e o HIBB com EQPC (11,27%) mostraram desempenhos muito próximos, superando significativamente o BBDE (4,94%) e ficando um pouco atrás do HIDE (12,88%). Isso indica que as versões do HIBB apresentam capacidades de contração semelhantes nos experimentos realizados.

É importante notar que, apesar de o HIDE ter apresentado uma contração geral superior ao HIBB e ao BBDE, esse valor pode estar superestimado pela forma com que as caixas são geradas estocasticamente e se sobrepõem durante a exploração do espaço de busca. Por outro lado, valorações descartadas do BBDE e do HIBB durante o processo de consistência local são definitivamente eliminados dos domínios das variáveis, o que representa uma redução real no espaço de busca da instância.

Corroborando essas informações, a Figura 5.4 traz uma análise da porcentagem média de caixas descartadas por serem consideradas inteiramente inconsistentes e da porcentagem de contração média geral nas 15 execuções do HIBB com QPC e do HIBB com EQPC. As instâncias são ordenadas pelo número de dimensões e, aquelas com a mesma dimensionalidade, classificadas pela contração média geral. No gráfico superior, as maiores diferenças entre QPC e EQPC se concentram nas instâncias com menor dimensionalidade. A partir da instância 200 (12 dimensões), a porcentagem de caixas inconsistentes se torna similar entre ambas as abordagens. No gráfico inferior, nota-se um comportamento semelhante, com QPC se destacando em algumas instâncias com menor dimensionalidade e as porcentagens se tornando semelhantes conforme aumenta o número de dimensões. Isso se reflete na comparação da Tabela 5.9, em que existe uma pequena diferença de contração média geral entre QPC e EQPC.

A análise apresentada nesta seção auxilia na compreensão de como as estratégias de exploração em múltiplas frentes de busca e as ordenações por QPC e EQPC impactam o processo de busca. Observou-se que a estratégia de abortar a busca em caixas pequenas permite ao HIBB com QPC consumir menos orçamento e explorar menos caixas, embora atinja menor profundidade na árvore de busca. Já o HIBB com EQPC atinge maior profundidade, o que pode levar a melhores soluções. Na próxima seção, será realizada uma análise comparativa da qualidade das soluções obtidas pelos otimizadores híbridos propostos, aprofundando a avaliação de seu desempenho geral.

## 5.3 ANÁLISE COMPARATIVA ENTRE OTIMIZADORES

Os três métodos híbridos de otimização propostos nesta tese – BBDE com reinícios + GAC, HIDE + GAC, HIBB com QPC ou EQPC – foram apresentados individualmente, com suas estratégias e heurísticas detalhadas e seus principais parâmetros discutidos em suas seções específicas. O BBDE com reinícios + GAC combina Evolução Diferencial (DE) com reinícios periódicos e técnicas de consistência local (Seção 4.2). O HIDE + GAC integra Evolução Diferencial Intervalar com uma busca local e técnicas de consistência (Seção 4.3). O HIBB com QPC aplica *Branch& Bound* intervalar com ordenação por Qualidade do Ponto representante da Caixa, enquanto o HIBB com EQPC utiliza ordenação por Estagnação e Qualidade do Ponto representante da Caixa. Nesta seção, os resultados desses métodos são comparados a fim de avaliar o desempenho de cada abordagem.

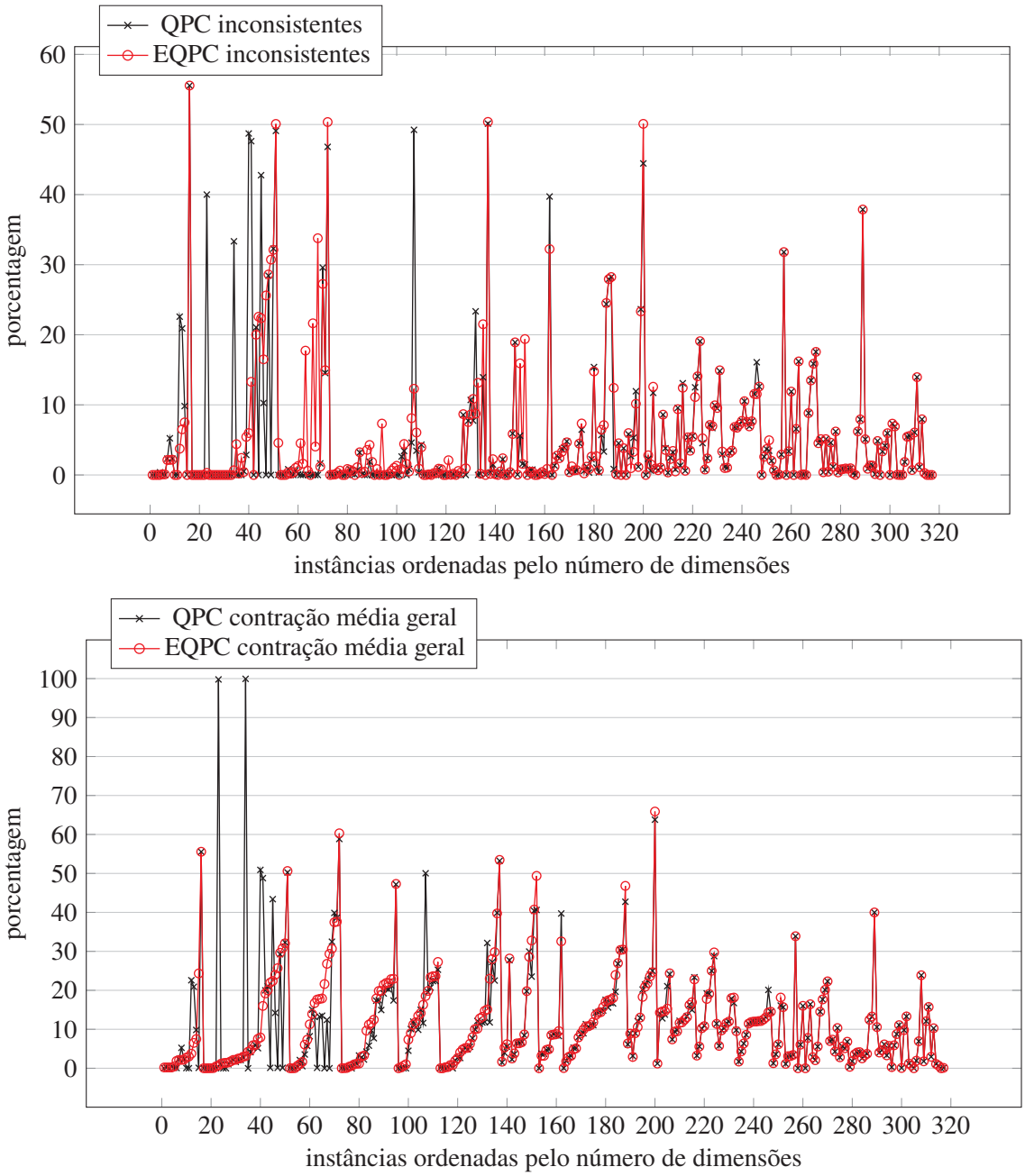



Figura 5.4: Média de caixas inconsistentes e contração média geral obtidas nas 15 execuções do HIBB com QPC e do HIBB com EQPC.

A Tabela 5.10 apresenta a quantidade de instâncias para as quais foram encontradas soluções factíveis e ótimas, assim como o tempo médio de execução de cada um dos otimizadores híbridos propostos. Em termos de factibilidade, as duas variações do método HIBB demonstram uma superioridade significativa, especialmente em relação ao HIDE + GAC. No que diz respeito à quantidade de instâncias com soluções ótimas, há um empate de 195 instâncias entre ambas as versões do HIBB e o HIDE + GAC, enquanto o BBDE atinge apenas 180 instâncias. Destaca-se também a similaridade dos resultados obtidos pelo HIBB com QPC e o HIBB com EQPC. Além disso, a tabela revela que os tempos médios de execução dos otimizadores híbridos propostos são bastante próximos, variando de 169,8 segundos (HIBB com EQPC) a 177,9 segundos (HIDE + GAC).

Tabela 5.10: Quantidade de instâncias com soluções factíveis e ótimas e tempo médio de execução obtidos pelos métodos híbridos propostos.

| | | BBDE com reinícios + GAC | | | HIDE + GAC | | |
|---|---|---|---|---|---|---|---|
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 183 | 143 | 26,0 | 186 | 158 | 27,1 |
| 11 a 20 | 59 | 53 | 32 | 95,7 | 54 | 32 | 95,5 |
| 21 a 30 | 16 | 13 | 4 | 244,1 | 13 | 4 | 245,0 |
| 31 a 40 | 13 | 12 | 0 | 304,7 | 11 | 0 | 304,6 |
| 41 a 50 | 19 | 12 | 0 | 675,0 | 8 | 0 | 674,1 |
| 51 a 60 | 9 | 7 | 1 | 688,1 | 5 | 1 | 681,2 |
| 61 a 70 | 8 | 5 | 0 | 961,9 | 2 | 0 | 975,9 |
| 71 a 75 | 5 | 3 | 0 | 1868,6 | 1 | 0 | 2207,4 |
| total | 317 | 288 | 180 | 171,8 | 280 | 195 | 177,9 |
| | | QPC com $NB$ inicial = $0{,}5 \times D$ | | | EQPC com $NB$ inicial = $0{,}5 \times D$ | | |
| dimensões | número de instâncias | factíveis | ótimas | tempo médio (s) | factíveis | ótimas | tempo médio (s) |
| 1 a 10 | 188 | 187 | 154 | 26,7 | 187 | 153 | 26,5 |
| 11 a 20 | 59 | 57 | 34 | 94,8 | 57 | 35 | 93,9 |
| 21 a 30 | 16 | 13 | 6 | 245,2 | 13 | 6 | 243,2 |
| 31 a 40 | 13 | 12 | 0 | 305,1 | 12 | 0 | 302,7 |
| 41 a 50 | 19 | 15 | 0 | 669,7 | 15 | 0 | 667,0 |
| 51 a 60 | 9 | 8 | 1 | 678,9 | 8 | 1 | 675,2 |
| 61 a 70 | 8 | 4 | 0 | 964,8 | 4 | 0 | 958,2 |
| 71 a 75 | 5 | 2 | 0 | 1828,5 | 2 | 0 | 1817,1 |
| total | 317 | 298 | 195 | 171,0 | 298 | 195 | 169,8 |

Tabela 5.11: Classificação dos otimizadores híbridos propostos de acordo com o método do CEC2020.

| otimizador | melhor | média | mediana | total | classificação |
|---|---|---|---|---|---|
| **HIBB com QPC** | **0,1188** | **0,1531** | **0,1299** | **0,1382** | **1º** |
| HIBB com EQPC | 0,1218 | 0,1615 | 0,1344 | 0,1442 | 2º |
| HIDE + GAC | 0,1734 | 0,2567 | 0,2364 | 0,2277 | 3º |
| BBDE com reinícios + GAC | 0,2661 | 0,3438 | 0,3173 | 0,3152 | 4º |

Comparando os resultados de forma mais detalhada, a Tabela 5.11 revela que o método HIBB com QPC obteve o melhor desempenho geral de acordo com o método do CEC2020, apresentando melhores resultados sobre a melhor, a média e a mediana das 15 execuções. O método HIBB com EQPC também apresentou um desempenho sólido, com pontuações ligeiramente inferiores às da abordagem com QPC na média e na mediana das soluções. A proximidade dos resultados entre QPC e EQPC sugere que ambas as estratégias possuem desempenhos muito semelhantes nos experimentos realizados. As pontuações obtidas pelo HIDE + GAC indicam um desempenho significativamente inferior em relação aos métodos HIBB. Por fim, o BBDE com reinícios + GAC apresentou o pior desempenho entre os quatro métodos analisados. Isso sugere que, embora as meta-heurísticas híbridas propostas sejam melhores que suas versões originais, não são tão eficazes quanto as abordagens baseadas em B&B diante das instâncias e parâmetros testados.

Essa análise é complementada pela Figura 5.5, que apresenta o número acumulado de instâncias com soluções factíveis obtidas por cada método, considerando diferentes limites de erro absoluto e destacando as melhores, médias, medianas e piores soluções das 15 execuções

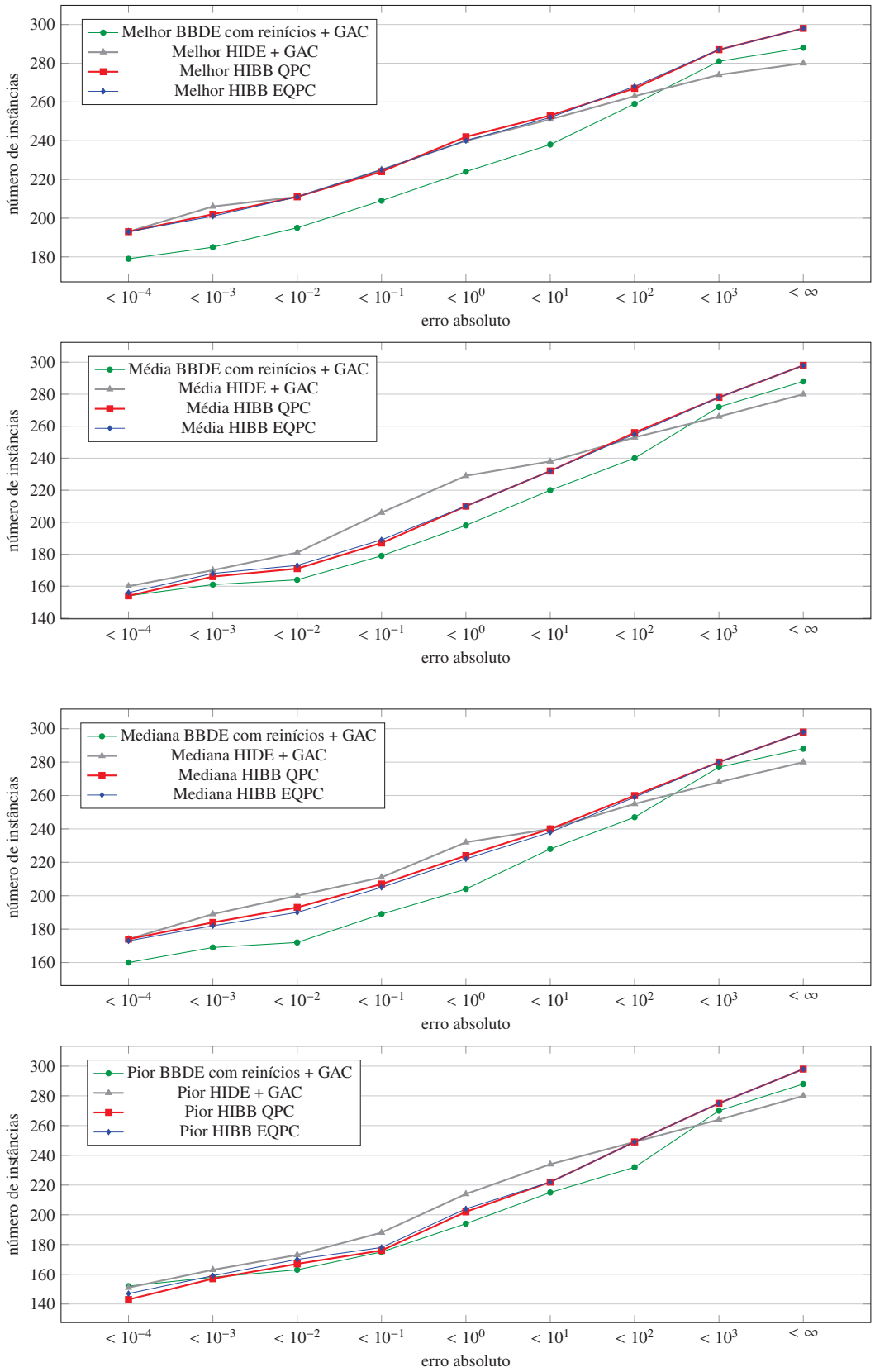



Figura 5.5: Número acumulado de instâncias com soluções factíveis obtidas nas 15 execuções do BBDE com reinícios + GAC, HIDE + GAC, HIBB com QPC e HIBB com EQPC, dados alguns limites de erro absoluto.

realizadas. É possível observar que ambas as versões de HIBB apresentam desempenhos muito

próximos entre si e superiores em relação ao BBDE com reinícios + GAC e ao HIDE + GAC, o que corrobora a classificação da tabela anterior. O HIDE + GAC também apresenta desempenho competitivo em termos de erro absoluto, ficando muito próximo ou acima das versões de HIBB em alguns limites de erro. No entanto, o número de soluções factíveis encontradas é menor, como evidenciado no limite $< \infty$, o que contribui para a sua classificação inferior pelo método do CEC2020. Já o BBDE com reinícios + GAC mostra-se competitivo em termos de factibilidade, mas os erros absolutos das soluções encontradas são, em geral, maiores do que os observados nas demais abordagens. Isso indica que, apesar de o BBDE beneficiar-se de técnicas de consistência local, ele não atinge o mesmo nível de eficácia dos métodos HIBB.

## 5.4 CONSIDERAÇÕES

Neste capítulo, propôs-se o HIBB como um método híbrido de otimização que integra técnicas de consistência local, geralmente utilizadas em métodos exatos, com uma busca local estocástica baseada em meta-heurísticas. Este método é composto por três etapas: aplicação do contrator GAC para poda de soluções sub-ótimas, uma busca local baseada em BBDE para avaliar a qualidade das caixas e a bissecção dessas caixas para continuidade da busca.

Os primeiros testes utilizaram uma única frente de busca implementada na forma de uma estratégia melhor-primeiro (*best-first*). A fim de melhorar a qualidade das soluções obtidas, foi proposta a utilização de múltiplas frentes de busca (*NB*), iniciando a execução com $NB = 0{,}5 \times D$, em que $D$ é o número de dimensões da instância. O valor de *NB* diminui ao longo da execução para gradualmente focar a busca nas regiões mais promissoras. Essa abordagem visa alocar os recursos computacionais de maneira a permitir que múltiplas regiões do espaço de busca sejam exploradas simultaneamente, especialmente nos estágios iniciais da busca. Além disso, foram propostas duas estratégias de ordenação para a fila de prioridade: a primeira considera a qualidade do ponto representante da caixa (QPC), enquanto a segunda avalia a estagnação do processo de busca na caixa e, em seguida, a qualidade de seu ponto representante (EQPC).

A avaliação experimental mostrou que as heurísticas propostas trazem vantagens significativas para a exploração do espaço de busca. Especificamente, observou-se que o HIBB com QPC consome menos orçamento, explora menos caixas e atinge menor profundidade na árvore de busca comparado ao HIBB com EQPC. No entanto, isso não parece impactar significativamente o orçamento consumido até a melhor solução encontrada. É importante destacar que, apesar de obterem resultados muito próximos, a estratégia QPC possui um parâmetro adicional relacionado à definição do tamanho mínimo a ser atingido pelas caixas antes de serem abandonadas. Definir esse parâmetro não é trivial e pode depender de cada instância específica.

A análise comparativa entre os métodos híbridos propostos mostrou que o HIBB com QPC e o HIBB com EQPC superam o BBDE com reinícios + GAC e o HIDE + GAC em termos de desempenho geral e eficiência na exploração do espaço de busca. O HIBB com QPC obteve os melhores resultados e um dos melhores tempos médios de execução, enquanto o HIBB com EQPC apresentou desempenho muito próximo. Embora o HIDE + GAC tenha mostrado bons resultados em termos de erro absoluto, a menor quantidade de soluções factíveis encontradas prejudicou seu desempenho. Apesar de o BBDE com reinícios + GAC ser mais robusto em relação à sua versão original, apresentou resultados inferiores em relação às demais abordagens.

De modo geral, as abordagens híbridas apresentaram resultados significativamente superiores às versões originais, explorando de maneira eficaz as vantagens individuais de métodos exatos e meta-heurísticas. Adicionalmente, a estratégia do HIBB de explorar o espaço de busca sem sobreposições de caixas, característica dos métodos de B&B, demonstrou melhores resultados em comparação à exploração estocástica com sobreposição de caixas utilizada pelo

HIDE e à exploração direta de pontos do espaço de busca empregada pelo BBDE. Esta abordagem sistemática do HIBB contribuiu para uma exploração mais eficiente e direcionada do espaço de busca, resultando em melhores soluções.

# 6 CONCLUSÃO

Na Seção 1.2, foi definido como objetivo desta tese desenvolver e avaliar métodos híbridos de otimização que combinam técnicas de otimização exata e meta-heurísticas, a fim de melhorar o desempenho de abordagens clássicas em problemas de otimização numérica global com restrições. Nos Capítulos 4 e 5, foram detalhados os três otimizadores híbridos propostos: BBDE, HIDE e HIBB. Além disso, foram avaliadas diferentes estratégias de exploração do espaço de busca e maneiras de potencializar o desempenho desses otimizadores híbridos.

Mais especificamente, o BBDE (Seção 4.2) consiste em uma versão da meta-heurística Evolução Diferencial (DE) que explora de maneira estocástica pontos específicos dentro do espaço de busca. A versão básica do BBDE, publicada em Cassenote et al. (2021), integra múltiplas heurísticas e estratégias, tendo alcançado desempenho superior a diversos otimizadores recentes da literatura. Para evitar a estagnação durante a busca, neste trabalho foi implementado um esquema de reinícios periódicos. Além disso, o BBDE foi combinado com técnicas de consistência local, em que o valor da função objetivo de cada solução factível encontrada é fornecido ao contrator GAC para filtragem do espaço de busca da instância. Essa combinação contribuiu para melhorar significativamente o desempenho do BBDE original (Cassenote et al., 2021), tornando o BBDE com reinícios + GAC competitivo em relação aos outros otimizadores híbridos propostos nesta tese.

O HIDE (Seção 4.3) é uma hibridização que combina uma versão aprimorada de DE Intervalar, técnicas de consistência local e uma busca local baseada no BBDE. Este método gera caixas estocasticamente no espaço de busca, sendo cada ponto dentro de uma caixa considerado uma solução candidata. O contrator GAC é aplicado para podar regiões sub-ótimas e infactíveis, enquanto o BBDE executa uma busca local dentro dessas caixas e seleciona o melhor ponto encontrado como seu representante. A análise experimental realizada indica que a inclusão da etapa de contração melhora significativamente a qualidade das soluções, destacando o HIDE como uma meta-heurística baseada em intervalos com desempenho superior ao BBDE.

Por fim, o HIBB (Capítulo 5) foi introduzido como um método híbrido de *Branch & Bound* (B&B) intervalar composto por três etapas: aplicação do contrator GAC para poda de soluções sub-ótimas, uma busca local baseada em BBDE e a bissecção das caixas para continuidade da busca. A fim de melhorar a qualidade das soluções, foi proposta a utilização de múltiplas frentes de exploração (*NB*), que reduzem ao longo da execução para focar nas regiões mais promissoras. Além disso, foram propostas duas estratégias de ordenação para a fila de prioridade do B&B: uma baseada na qualidade do ponto representante da caixa (QPC), e outra que considera a estagnação do processo de busca na caixa antes da qualidade do ponto representente (EQPC). É importante destacar que a estratégia QPC possui um parâmetro adicional relacionado à definição do tamanho mínimo a ser atingido pelas caixas antes de serem abandonadas. Definir esse parâmetro não é uma tarefa trivial e pode depender de cada instância específica. A avaliação experimental mostrou resultados bastante similares para as estratégias QPC e EQPC, o que indica que ambas as heurísticas propostas trazem vantagens significativas para a exploração do espaço de busca.

Uma contribuição secundária significativa do grupo de pesquisa ao qual esta tese está vinculada envolve o desenvolvimento de duas ferramentas (Seção 4.1) que possibilitaram a implementação e avaliação experimental dos otimizadores híbridos propostos. Uma dessas ferramentas é um *parser* que converte instâncias no formato AMPL (*A Mathematical Programming Language*) para uma rede de restrições ternárias utilizada no processo de consistência local, e para

um código-fonte em linguagem C que avalia a função objetivo e as violações de restrições a partir das valorações das soluções candidatas. Além disso, o grupo realizou uma extensão substancial do núcleo multi-intervalar também utilizado na etapa de consistência local dos otimizadores híbridos propostos.

A análise comparativa das melhores versões dos métodos híbridos propostos mostrou que tanto o HIBB com QPC quanto o HIBB com EQPC superam o BBDE com reinícios + GAC e o HIDE + GAC em termos de qualidade das soluções e eficiência na exploração do espaço de busca. O HIBB com QPC obteve os melhores resultados e um dos menores tempos médios de execução, com desempenho muito próximo ao do HIBB com EQPC. Embora o HIDE + GAC tenha mostrado bons resultados em termos de erro absoluto, obteve uma menor quantidade de soluções factíveis. O BBDE com reinícios + GAC, apesar de superior em relação a sua versão original, apresentou resultados inferiores aos demais métodos híbridos.

Em geral, as abordagens híbridas obtiveram desempenhos significativamente superiores às suas versões originais, beneficiando-se de características complementares de métodos exatos e meta-heurísticas. Ademais, a estratégia do HIBB de explorar o espaço de busca por meio de uma segmentação típica de métodos de B&B demonstrou melhores resultados em comparação à exploração estocástica com sobreposição de caixas utilizada pelo HIDE e à exploração direta de pontos do espaço de busca empregada pelo BBDE. Desta forma, conclui-se que o objetivo desta tese foi atingido.

## 6.1 LIMITAÇÕES E TRABALHOS FUTUROS

Uma das principais limitações no desenvolvimento desta tese refere-se à escalabilidade dos métodos híbridos propostos. Embora tenham demonstrado desempenho promissor em termos de tempo de execução e qualidade das soluções, ainda não foram avaliados em problemas com um número elevado de dimensões. Acredita-se que, nesses casos, a implementação de outros métodos de consistência local e de busca local possa contribuir para a eficiência do processo de busca e para um melhor direcionamento dos recursos computacionais disponíveis. Outra possibilidade é a utilização de técnicas de paralelismo para aumentar a escalabilidade e robustez dos otimizadores. Essas melhorias podem tornar os métodos propostos comparáveis a resolvedores do estado-da-arte, tais como BARON (Sahinidis, 1996), Couenne (Belotti et al., 2009) e GlobSol (Kearfott, 2013).

Os métodos híbridos propostos utilizam as soluções factíveis encontradas pela busca local para poda do limite superior da solução ótima durante o processo de consistência local. Um trabalho futuro importante é adicionar um passo de avaliação das soluções factíveis encontradas por meio de aritmética intervalar para garantir que não haja descarte inadequado de valores devido a erros de representação numérica. Implementar essa verificação adicional poderá melhorar a qualidade dos resultados.

Ainda que tenham sido aplicadas diversas heurísticas e estratégias adaptativas para ajustar as configurações de parâmetros dos otimizadores propostos, alguns deles ainda impactam consideravelmente a qualidade das soluções e o tempo de execução, como o número de frentes de busca (*NB*) no HIBB e o tamanho mínimo que as caixas devem atingir antes de serem abandonadas na estratégia de ordenação QPC. Trabalhos futuros incluem a análise e implementação de abordagens automatizadas que possam minimizar a necessidade de ajuste desses parâmetros.

Para melhor compreender as limitações e potencialidades dos métodos híbridos propostos, é necessário realizar uma análise de desempenho por tipos de problemas. Isso envolve aplicar os métodos a uma maior diversidade de problemas, incluindo diferentes tipos de funções objetivo e

restrições, a fim de avaliar a generalização dos métodos. Realizar essa análise permitirá identificar em quais problemas os métodos híbridos são mais eficazes e onde devem ser melhorados, fornecendo uma visão mais detalhada sobre seu desempenho e potenciais aplicações.

# REFERÊNCIAS


Akhmedova, S. e Stanovov, V. (2020). Self-tuning co-operation of biology-inspired and evolutionary algorithms for real-world single objective constrained optimization. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–6. IEEE.

Alliot, J.-M., Durand, N., Gianazza, D. e Gotteland, J.-B. (2012). Finding and proving the optimum: Cooperative stochastic and deterministic search. Em *ECAI 2012, 20th European Conference on Artificial Intelligence*, páginas 55–60.

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

Araya, I., Trombettoni, G. e Neveu, B. (2010). Exploiting monotonicity in interval constraint propagation. Em *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Araya, I., Trombettoni, G. e Neveu, B. (2012). A contractor based on convex interval taylor. Em *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, páginas 1–16. Springer.

Awad, N. H., Ali, M. Z., Liang, J. J., Qu, B. Y. e Suganthan, P. N. (November 2016). Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. Relatório técnico, Nanyang Technological University, Singapore.

Awad, N. H., Ali, M. Z. e Suganthan, P. N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems. Em *2017 IEEE Congress on Evolutionary Computation (CEC)*, páginas 372–379. IEEE.

Baltussen, T. M. J. T., Goutham, M., Menon, M., Garrow, S. G., Santillo, M. e Stockar, S. (2023). A parallel monte-carlo tree search-based metaheuristic for optimal fleet composition considering vehicle routing using branch & bound. Em *2023 IEEE Intelligent Vehicles Symposium (IV)*, páginas 1–6.

Bandaru, S. e Deb, K. (2016). Metaheuristic techniques. *Decision sciences*, páginas 693–750.

Bazaraa, M. S., Sherali, H. D. e Shetty, C. M. (2006). *Nonlinear programming: theory and algorithms*. John wiley & sons.

Belotti, P., Lee, J., Liberti, L., Margot, F. e Wächter, A. (2009). Branching and bounds tighteningtechniques for non-convex minlp. *Optimization Methods & Software*, 24(4-5):597–634.

Benhamou, F., Goualard, F., Granvilliers, L. e Puget, J.-F. (1999). Revising hull and box consistency. Em *Proceedings of the 15th International Conference on Logic Programming*, páginas 230–244. MIT press.

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

Bertsekas, D. (2016). *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific.

Bertsekas, D. P., Hager, W. e Mangasarian, O. (1999). Nonlinear programming. athena scientific belmont. *Massachusets, USA*.

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

Bhurjee, A. K. e Panda, G. (2012). Efficient solution of interval optimization problem. *Mathematical Methods of Operations Research*, 76(3):273–288.

Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L. e Abraham, A. (2020). Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, 90:103479.

Blum, C. e Raidl, G. R. (2016). *Hybrid Metaheuristics: Powerful Tools for Optimization*. Springer.

Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308.

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

Boyd, S. e Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brest, J., Maučec, M. S. e Bošković, B. (2016). il-shade: Improved l-shade algorithm for single objective real-parameter optimization. Em *2016 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1188–1195. IEEE.

Brest, J., Maučec, M. S. e Bošković, B. (2017). Single objective real-parameter optimization: Algorithm jso. Em *2017 IEEE congress on evolutionary computation (CEC)*, páginas 1311–1318. IEEE.

Bujok, P. e Kolenovsky, P. (2022). Eigen crossover in cooperative model of evolutionary algorithms applied to cec 2022 single objective numerical optimisation. Em *2022 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8.

Bunnag, D. (2014). Combining interval branch and bound and stochastic search. Em *Abstract and Applied Analysis*, volume 2014. Hindawi.

Cassenote, M. R. S. (2019). Evolução Diferencial Intervalar: uma abordagem baseada em decomposição estrutural de problemas de otimização global. Dissertação de Mestrado, Pós-Graduação em Informática - Universidade Federal do Paraná, Curitiba - PR.

Cassenote, M. R. S., Derenievicz, G. A. e Silva, F. (2019). Interval Differential Evolution using structural information of global optimization problems. Em *EPIA Portuguese Conference on Artificial Intelligence*, páginas 724–736. Springer.

Cassenote, M. R. S., Derenievicz, G. A. e Silva, F. (2021). I2DE: Improved interval differential evolution for numerical constrained global optimization. Em *Intelligent Systems. BRACIS 2021. Lecture Notes in Computer Science*, páginas 186–201, São Paulo - Brazil.

Cassenote, M. R. S., Derenievicz, G. A. e Silva, F. (2024). A Hybrid Approach Integrating Generalized Arc Consistency and Differential Evolution for Global Optimization. Em Dilkina, B., editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, páginas 190–207, Cham. Springer Nature Switzerland.

Chabert, G., Trombettoni, G. e Neveu, B. (2004). New light on arc consistency over continuous domains. Relatório Técnico RR-5365, INRIA.

Chakraborty, U. K. (2008). *Advances in differential evolution*, volume 143. Springer.

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.

Coello, C. A. C. (2022). Constraint-handling techniques used with evolutionary algorithms. Em *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, página 1310–1333, New York, NY, USA. Association for Computing Machinery.

Cohen, D. A. e Jeavons, P. G. (2017). The power of propagation: when gac is enough. *Constraints*, 22(1):3–23.

Cohoon, J. P., Hegde, S. U., Martin, W. N. e Richards, D. (1987). Punctuated equilibria: a parallel genetic algorithm. Em *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987.

Cotta, C. e Troya, J. M. (2003). Embedding branch and bound within evolutionary algorithms. *Applied Intelligence*, 18(2):137–153.

Cotta-Porras, C. (1998). A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Communications*, 11(3, 4):223–224.

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.

Derenievicz, G. A. (2018). *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.

El-Abd, M. e Kamel, M. (2005). A taxonomy of cooperative search algorithms. Em *International Workshop on Hybrid Metaheuristics*, páginas 32–41. Springer.

Faltings, B. (1998). Arc consistency for continuous variables. *Artificial Intelligence*, 65: 363–376.

Faltings, B. e Gelle, E. M. (1997). Local consistency for ternary numeric constraints. Em *15th International Joint Conference on Artificial Intelligence*, páginas 392–397.

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

Gallardo, J. E., Cotta, C. e Fernández, A. J. (2007). On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):77–83.

Gilli, M., Maringer, D. e Schumann, E. (2019). *Numerical Methods and Optimization in Finance*. Elsevier Science.

Gmys, J., Mezmaz, M., Melab, N. e Tuyttens, D. (2020). A computationally efficient branch-and-bound algorithm for the permutation flow-shop scheduling problem. *European Journal of Operational Research*, 284(3):814–833.

Grossmann, I. E. (2014). Challenges in the application of mathematical programming in the enterprise-wide optimization of process industries. *Theoretical Foundations of Chemical Engineering*, 48(5):555–573.

Guo, S.-M., Tsai, J. S.-H., Yang, C.-C. e Hsu, P.-H. (2015). A self-optimization approach for l-shade incorporated with eigenvector-based crossover and successful-parent-selecting framework on cec 2015 benchmark set. Em *2015 IEEE congress on evolutionary computation (CEC)*, páginas 1003–1010. IEEE.

Gupta, A. e Ray, S. (2010). Economic emission load dispatch using interval differential evolution algorithm. Em *4th International Workshop on reliable Engineering Computing (REC 2010)*. Citeseer.

Gurrola-Ramos, J., Hernàndez-Aguirre, A. e Dalmau-Cedeño, O. (2020). Colshade for real-world single-objective constrained optimization problems. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

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

Hansen, E. e Walster, G. W. (2004b). *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.

Hansen, N., Müller, S. D. e Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18.

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.

Hellwig, M. e Beyer, H.-G. (2020a). A modified matrix adaptation evolution strategy with restarts for constrained real-world problems. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

Hellwig, M. e Beyer, H.-G. (2020b). A modified matrix adaptation evolution strategy with restarts for constrained real-world problems. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8.

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

Hillier, F. e Lieberman, G. (2013). *Introdução à Pesquisa Operacional*. AMGH.

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

Hussain, K., Salleh, M. N. M., Cheng, S. e Shi, Y. (2019). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4):2191–2233.

Iorio, A. W. e Li, X. (2004). Solving rotated multi-objective optimization problems using Differential Evolution. Em *Australasian joint conference on artificial intelligence*, páginas 861–872. Springer.

Jaulin, L., Kieffer, M., Didrit, O. e Walter, E. (2001). Interval analysis. Em *Applied interval analysis*, páginas 11–43. Springer.

Kaveh, A. e Eslamlou, A. D. (2020). *Metaheuristic optimization algorithms in civil engineering: new applications*, volume 900. Springer Nature.

Kearfott, R. B. (1992). An interval branch and bound algorithm for bound constrained optimization problems. *J. Glob. Optim.*, 2(3):259–280.

Kearfott, R. B. (2013). *Rigorous global search: continuous problems*, volume 13. Springer Science & Business Media.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. Em *Proceedings of the IEEE International Conference on Neural Networks*, páginas 1942–1948. IEEE.

Kizilay, D., Tasgetiren, M. F., Oztop, H., Kandiller, L. e Suganthan, P. N. (2020). A differential evolution algorithm with q-learning for solving engineering design problems. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

Korte, B. H., Vygen, J., Korte, B. e Vygen, J. (2011). *Combinatorial optimization*, volume 1. Springer.

Kumar, A., Das, S. e Zelinka, I. (2020a). A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems. Em *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, páginas 11–12.

Kumar, A., Das, S. e Zelinka, I. (2020b). A self-adaptive spherical search algorithm for real-world constrained optimization problems. Em *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, páginas 13–14.

Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N. e Das, S. (2020c). Guidelines for real-world single-objective constrained optimisation competition. Relatório técnico, Technical report.

Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N. e Das, S. (2020d). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56:100693.

Lawler, E. L. e Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719.

Lee, C.-Y. e Yao, X. (2004). Evolutionary programming using mutations based on the lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8(1):1–13.

Liang, J., Qu, B., Suganthan, P. e Chen, Q. (2014). Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 29:625–640.

Liang, J., Qu, B., Suganthan, P. e Hernández-Díaz, A. G. (2013a). Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 201212(34):281–295.

Liang, J. J., Qu, B. Y. e Suganthan, P. N. (2013b). Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 635:490.

Mackworth, A. K. (1977a). Consistency in networks of relations. *Artificial intelligence*, 8(1):99–118.

Mackworth, A. K. (1977b). On reading sketch maps. Em *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, IJCAI 1977*, páginas 598–606, MIT, Cambridge, MA.

Malik, H., Iqbal, A., Joshi, P., Agrawal, S. e Bakhsh, F. I. (2021). *Metaheuristic and evolutionary computation: algorithms and applications*. Springer.

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.

Martins, J. e Ning, A. (2021). *Engineering Design Optimization*. Cambridge University Press.

Mezura-Montes, E. (2009). *Constraint-Handling in Evolutionary Optimization*, volume 9. Universidad Nacional de La Plata.

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

Mohamed, A. W., Hadi, A. A., Agrawal, P., Sallam, K. M. e Mohamed, A. K. (2021). Gaining-sharing knowledge based algorithm with adaptive parameters hybrid with imode algorithm for solving cec 2021 benchmark problems. Em *2021 IEEE Congress on Evolutionary Computation (CEC)*, páginas 841–848.

Mohamed, A. W., Hadi, A. A., Fattouh, A. M. e Jambi, K. M. (2017). Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems. Em *2017 IEEE Congress on evolutionary computation (CEC)*, páginas 145–152. IEEE.

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.

Moscato, P. (1999). Memetic algorithms: A short introduction. *New ideas in optimization*, páginas 219–234.

Mérel, P. e Howitt, R. (2014). Theory and application of positive mathematical programming in agriculture and the environment. *Annual Review of Resource Economics*, 6(1):451–470.

Nocedal, J. e Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2e edition.

Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., Suganthan, P. N., Coello Coello, C. A. e Herrera, F. (2021). A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm and Evolutionary Computation*, 64:100888.

Ozkan, O., Ermis, M. e Bekmezci, I. (2019). Reliable communication network design: The hybridisation of metaheuristics with the branch and bound method. *Journal of the Operational Research Society*, 71(5):784–799.

Papadimitriou, C. H. e Steiglitz, K. (2013). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

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. (2016). 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.

Puchinger, J. e Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. Em *International work-conference on the interplay between natural and artificial computation*, páginas 41–53. Springer.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. Em *International workshop on hybrid metaheuristics*, páginas 1–12. Springer.

Raidl, G. R., Puchinger, J. e Blum, C. (2019). Metaheuristic hybrids. Em *Handbook of metaheuristics*, páginas 385–417. Springer.

Sahinidis, N. V. (1996). Baron: A general purpose global optimization software package. *Journal of global optimization*, 8(2):201–205.

Sallam, K. M., Elsayed, S. M., Chakrabortty, R. K. e Ryan, M. J. (2020). Improved multi-operator differential evolution algorithm for solving unconstrained problems. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

Schrijver, A. et al. (2003). *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer.

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.

Sidebottom, G. e Havens, W. S. (1992). Hierarchical arc consistency for disjoint real intervals in constraint logic programming. *Computational Intelligence*, 8.

Sotiropoulos, D., Stavropoulos, E. e Vrahatis, M. (1997). A new hybrid genetic algorithm for global optimization. *Nonlinear Analysis: Theory, Methods & Applications*, 30(7):4529–4538.

Soto, R., Crawford, B., Galleguillos, C., Monfroy, E. e Paredes, F. (2013). A hybrid ac3-tabu search algorithm for solving sudoku puzzles. *Expert Systems with Applications*, 40(15):5817–5821.

Stanovov, V., Akhmedova, S. e Semenkin, E. (2018). Lshade algorithm with rank-based selective pressure strategy for solving cec 2017 benchmark problems. Em *2018 IEEE congress on evolutionary computation (CEC)*, páginas 1–8. IEEE.

Storn, R. e Price, K. (1995). Differrential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical report, International Computer Science Institute*, 11.

Sun, M. e Johnson, A. (2005). Interval branch and bound with local sampling for constrained global optimization. *Journal of Global Optimization*, 33:61–82.

Tahami, H. e Fakhravar, H. (2022). A literature review on combining heuristics and exact algorithms in combinatorial optimization. *European Journal of Information Technologies and Computer Science*, 2(2):6–12.

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

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 8(5):541–564.

Talbi, E.-G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research*, 240(1):171–215.

Talbi, E.-G. et al. (2013). *Hybrid metaheuristics*, volume 166. Springer.

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.

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., 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).

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.

Van Kampen, E. (2010). *Global optimization using interval analysis: interval optimization for aerospace applications*. Tese de doutorado, Aerospace Engineering, Zutphen, The Netherlands.

Vanaret, C. (2015). *Hybridization of interval methods and evolutionary algorithms for solving difficult optimization problems*. Tese de doutorado, l'Institut National Polytechnique de Toulouse, Toulouse - France.

Vanaret, C., Gotteland, J.-B., Durand, N. e Alliot, J.-M. (2013a). A fast and reliable hybrid algorithm for numerical nonlinear global optimization. Em *AAAI 2013, 27th AAAI Conference on Artificial Intelligence*.

Vanaret, C., Gotteland, J.-B., Durand, N. e Alliot, J.-M. (2013b). Preventing premature convergence and proving the optimality in evolutionary algorithms. Em *International Conference on Artificial Evolution (Evolution Artificielle)*, páginas 29–40. Springer.

Vanaret, C., Gotteland, J.-B., Durand, N. e Alliot, J.-M. (2015). Hybridization of interval cp and evolutionary algorithms for optimizing difficult problems. Em *International Conference on Principles and Practice of Constraint Programming*, páginas 446–462. Springer.

Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T. e Zamuda, A. (2020). Dish-xx solving cec2020 single objective bound constrained numerical optimization benchmark. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

Wen, X., Wu, G., Fan, M., Wang, R. e Suganthan, P. N. (2020). Voting-mechanism based ensemble constraint handling technique for real-world single-objective constrained optimization. Em *2020 IEEE Congress on Evolutionary Computation (CEC)*, páginas 1–8. IEEE.

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, National University of Defense Technology, and Kyungpook National University and Nanyang Technological University.

Yang, X.-S. (2010a). Firefly algorithm, levy flights and global optimization. Em *Research and development in intelligent systems XXVI*, páginas 209–218. Springer.

Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. Em *Nature inspired cooperative strategies for optimization (NICSO 2010)*, páginas 65–74. Springer.

Yang, X.-S., Chien, S. F. e Ting, T. O. (2015). Chapter 1 - bio-inspired computation and optimization: An overview. Em Yang, X.-S., Chien, S. F. e Ting, T. O., editores, *Bio-Inspired Computation in Telecommunications*, páginas 1–21. Morgan Kaufmann, Boston.

Yang, X.-S. e Deb, S. (2009). Cuckoo search via lévy flights. Em *2009 World congress on nature & biologically inspired computing (NaBIC)*, páginas 210–214. Ieee.

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, X. e Liu, S. (2007). A new interval-genetic algorithm. Em *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, páginas 193–197. IEEE.

Zhang, Y., Chen, G., Cheng, L., Wang, Q. e Li, Q. (2023). Methods to balance the exploration and exploitation in differential evolution from different scales: A survey. *Neurocomputing*, 561:126899.

Öztop, H., Tasgetiren, M. F., Kandiller, L. e Pan, Q.-K. (2022). Metaheuristics with restart and learning mechanisms for the no-idle flowshop scheduling problem with makespan criterion. *Computers & Operations Research*, 138:105616.