Universidade Federal de Ouro Preto
Escola de Minas
Engenharia de Controle e Automação

Harrison de Lana Araújo e Silva

Pesquisa Operacional Aplicada ao *One-Sided Crossing Minimization* (OSCM) em Grafos Bipartidos

Ouro Preto, 2026

Harrison de Lana Araújo e Silva

Pesquisa Operacional Aplicada ao *One-Sided Crossing Minimization* (OSCM) em Grafos Bipartidos

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Pablo Luiz Araújo Munhoz
Coorientador: Prof. Dr. Agnaldo José da Rocha Reis

Ouro Preto
2026

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Harrison de Lana Araújo e Silva

Pesquisa Operacional Aplicada ao *One-Sided Crossing Minimization* (OSCM) em Grafos Bipartidos

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Controle e Automação

Aprovada em 27 de julho de 2026

Membros da banca

Doutor - Pablo Luiz Araújo Munhoz - Orientador (Universidade Federal de Ouro Preto)
Doutor - Agnaldo José da Rocha Reis - Coorientador (Universidade Federal de Ouro Preto)
Doutor - André Luiz Carvalho Ottoni - Examinador (Universidade Federal de Ouro Preto)
Doutor - Paulo Marcos de Barros Monteiro - Examinador (Universidade Federal de Ouro Preto)

Pablo Luiz Araújo Munhoz, orientador do trabalho, aprovou a versão final e autorizou o seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 29/07/2026

Documento assinado eletronicamente por **Pablo Luiz Araujo Munhoz**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 29/07/2026, às 17:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Joao Carlos Vilela de Castro**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 30/07/2026, às 07:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1147764** e o código CRC **CFBF8979**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Agradeço, primeiramente, a Deus, por me conceder a oportunidade de cursar Engenharia, por me dar forças para superar os desafios encontrados ao longo dessa caminhada e por guiar meus passos durante toda essa jornada.

Agradeço também aos meus pais, Arthur e Genísia, cujo apoio incondicional foi fundamental para a realização deste trabalho e para a conclusão desta importante etapa da minha vida. Vocês sempre foram minha base, oferecendo amor, dedicação, incentivo e valores que contribuíram decisivamente para minha formação pessoal e acadêmica.

À minha companheira, Jéssica, expresso minha sincera gratidão pelo carinho, pela paciência, pela compreensão nos momentos mais desafiadores e pelo constante incentivo ao longo de toda a graduação. Seu companheirismo foi essencial para que eu mantivesse a motivação e a confiança durante essa trajetória.

Ao meu orientador, Pablo, e ao meu coorientador, Agnaldo, agradeço pela orientação, disponibilidade, confiança e pelos conhecimentos compartilhados durante o desenvolvimento deste trabalho. Suas contribuições foram fundamentais para sua realização e para a qualidade dos resultados alcançados.

Aos meus amigos, agradeço pela amizade, pelo apoio e pelos inúmeros momentos compartilhados ao longo dessa caminhada. Em especial, ao meu amigo Sélio Guilherme, pela disponibilidade e pelas contribuições, que foram de grande importância para o desenvolvimento deste Trabalho de Conclusão de Curso.

Por fim, agradeço aos professores, técnicos e demais profissionais da Universidade Federal de Ouro Preto (UFOP), pelo comprometimento, dedicação e pelos conhecimentos transmitidos ao longo da graduação, que foram essenciais para minha formação acadêmica, profissional e pessoal.

"Se fosse fácil achar o caminho das pedras, tantas pedras no caminho não seria ruim"

— Engenheiros do Hawai

Resumo

A minimização de cruzamentos de arestas constitui um dos principais critérios de qualidade em desenhos de grafos, influenciando diretamente sua legibilidade e interpretabilidade. Nesse contexto, o Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* - OSCM) consiste em determinar uma ordenação para os vértices da camada livre de um grafo bipartido, mantendo fixa a ordenação da camada adjacente, de modo a minimizar o número de cruzamentos entre as arestas. Trata-se de um problema NP-difícil, com aplicações em áreas como desenho hierárquico de grafos, visualização de software e projeto de circuitos integrados (*Very Large Scale Integration* - VLSI). Este trabalho apresenta o desenvolvimento de métodos heurísticos para a resolução do OSCM. A metodologia proposta combina heurísticas construtivas para geração de soluções iniciais, procedimentos de busca local para refinamento das soluções e uma metaheurística *Simulated Annealing* (SA), responsável por intensificar a exploração do espaço de busca. Foram implementadas as heurísticas do Baricentro e da Mediana, as estruturas de vizinhança *Swap*, *Insert* e *2-Opt*, além do método *Variable Neighborhood Descent* (VND). Para acelerar a avaliação da função objetivo, foi empregada uma matriz de cruzamentos, permitindo o cálculo incremental das variações provocadas pelos movimentos de vizinhança. Os parâmetros do SA foram calibrados por meio da ferramenta *iRace*. Os experimentos computacionais foram realizados utilizando instâncias do *Parameterized Algorithms and Computational Experiments Challenge* (PACE) *Challenge 2024*. Inicialmente, foram conduzidos testes em um conjunto de 20 instâncias para comparar o desempenho das heurísticas construtivas, das buscas locais e da metaheurística proposta. Em seguida, o SA foi avaliado em 196 instâncias do conjunto oficial da competição. Os resultados mostraram que a heurística da Mediana produziu soluções iniciais de melhor qualidade que a heurística do Baricentro e que a integração entre a solução inicial, os procedimentos de refinamento e o *Simulated Annealing* proporcionou melhorias expressivas na qualidade das soluções. A abordagem proposta reduziu o Desvio Médio Relativo (DMR) de 8,20% para 0,75% em relação às melhores soluções conhecidas do PACE *Challenge* 2024, alcançando um DMR de apenas 0,24% quando desconsiderada uma instância específica que não apresentou melhoria. Esses resultados evidenciam a eficácia da combinação entre heurísticas construtivas, busca local e metaheurística na resolução do OSCM, produzindo soluções satisfatórias para instâncias de diferentes características.

Palavras-chave: Problema de Minimização de Cruzamentos Unilaterais; Desenho de grafos; Heurísticas; Busca local; *Simulated Annealing*; PACE *Challenge* 2024.

Abstract

Edge crossing minimization is one of the main quality criteria in graph drawing, as it directly influences the readability and interpretability of graph visualizations. In this context, the One-Sided Crossing Minimization (OSCM) problem consists of determining an ordering of the vertices in the free layer of a bipartite graph while keeping the ordering of the adjacent layer fixed, with the objective of minimizing the number of edge crossings. This is an NP-hard problem with applications in areas such as hierarchical graph drawing, software visualization, and Very Large Scale Integration (VLSI) circuit design. This work presents the development of heuristic methods for solving the OSCM problem. The proposed methodology combines constructive heuristics for generating initial solutions, local search procedures for solution refinement, and a Simulated Annealing (SA) metaheuristic to intensify the exploration of the search space. The implemented approaches include the Barycenter and Median heuristics, the Swap, Insert, and 2-Opt neighborhood structures, as well as the Variable Neighborhood Descent (VND) method. To accelerate objective function evaluation, a crossing matrix was employed, enabling the incremental computation of the changes produced by neighborhood moves. The parameters of the SA algorithm were calibrated using the iRace automatic configuration tool. Computational experiments were conducted using benchmark instances from the Parameterized Algorithms and Computational Experiments Challenge (PACE) Challenge 2024. Initially, experiments were carried out on a set of 20 instances to compare the performance of the constructive heuristics, local search methods, and the proposed metaheuristic. Subsequently, the SA algorithm was evaluated on 196 instances from the official competition benchmark. The results showed that the Median heuristic generated higher-quality initial solutions than the Barycenter heuristic and that the integration of the initial solution, refinement procedures, and Simulated Annealing led to significant improvements in solution quality. The proposed approach reduced the Average Relative Gap (ARG) from 8.20% to 0.75% with respect to the best-known solutions from the PACE Challenge 2024, achieving an ARG of only 0.24% when excluding a single instance for which no improvement was obtained. These results demonstrate the effectiveness of combining constructive heuristics, local search, and metaheuristics for solving the OSCM problem, producing competitive solutions for instances with different characteristics.

Keywords: One-Sided Crossing Minimization Problem; Graph drawing; Heuristics; Local search; Simulated Annealing; PACE Challenge 2024.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ACO	*Ant Colony Optimization*
AG	Algoritmos Genéticos
BKS	*Best Known Solution*
CLP	Controlador Lógico Programável
CPS	*Cyber-Physical Systems*
DMR	Desvio Médio Relativo
FBD	*Function Block Diagram*
FPT	*Fixed-Parameter Tractable*
GRASP	*Greedy Randomized Adaptive Search Procedure*
IHM	Interface Homem-Máquina
ILS	*Iterated Local Search*
OSCM	*One-Sided Crossing Minimization*
PACE	*Parameterized Algorithms and Computational Experiments*
PCB	*Printed Circuit Board*
PI	Programação Inteira
PL	Programação Linear
PO	Pesquisa Operacional
RCL	*Restricted Candidate List*
SA	*Simulated Annealing*
UML	*Unified Modeling Language*
VLSI	*Very Large Scale Integration*
VND	*Variable Neighborhood Descent*
VNS	*Variable Neighborhood Search*

Lista de símbolos

α	Parâmetro que controla o grau de aleatoriedade na construção da solução
α_{SA}	Fator de resfriamento do algoritmo *Simulated Annealing*
$BC(y)$	Valor do baricentro associado ao vértice y
$b_{\max}$	Maior valor de baricentro entre os vértices candidatos
$b_{\min}$	Menor valor de baricentro entre os vértices candidatos
$C = (c_{ij})$	Matriz de cruzamentos pré-computada
c_{ij}	Número de cruzamentos quando i precede j na camada livre
Δ	Variação da função objetivo causada por um movimento
$f(\pi)$	Função objetivo correspondente ao número total de cruzamentos da permutação π
$f(\pi')$	Função objetivo correspondente ao número total de cruzamentos da permutação π'
f^*	Valor da melhor função objetivo encontrada
L	Limite de tempo de execução
L_{iter}	Número máximo de iterações em cada temperatura
L_{neigh}	Limite de tempo para exploração de cada vizinhança
L_{VND}	Limite de tempo total da execução do VND
$m(y)$	Valor da mediana associado ao vértice y
$N(\pi)$	Vizinhança da permutação π
P	Probabilidade de aceitação de uma solução não melhorante
$P(y)$	Sequência ordenada das posições dos vizinhos de v na camada fixa
$\Pi(Y)$	Conjunto de todas as permutações dos vértices da camada livre
π	Permutação dos vértices da camada livre
π'	Permutação obtida após a aplicação de uma busca local

π^*	Melhor permutação encontrada
$\pi(y)$	Posição do vértice $y \in Y$ na permutação
T	Temperatura atual do algoritmo *Simulated Annealing*
Tk	Temperatura atual do algoritmo *Simulated Annealing* na interação k
$T_{\max}$	Temperatura inicial
$T_{\min}$	Temperatura mínima utilizada como critério de parada
τ	Limiar utilizado para construir a Lista Restrita de Candidatos (RCL)
X	Conjunto de vértices da camada fixa
$x(x)$	Posição fixa do vértice $x \in X$
Y	Conjunto de vértices da camada livre

Sumário

1 Introdução

A Pesquisa Operacional (PO) constitui uma importante área do conhecimento dedicada ao apoio à tomada de decisão em problemas complexos (TAHA, 2017). Por meio da utilização de modelos matemáticos, métodos computacionais e técnicas de análise quantitativa, a PO busca identificar soluções eficientes para problemas encontrados em diversos setores, como indústria, logística, transportes, saúde, energia e tecnologia (TAHA, 2017).

As origens da Pesquisa Operacional remontam à Segunda Guerra Mundial, período em que equipes multidisciplinares de cientistas passaram a empregar métodos científicos para otimizar a utilização de recursos militares. Conforme descrito por Taha (2008), as primeiras aplicações formais da área ocorreram na Inglaterra, onde pesquisadores utilizaram técnicas quantitativas para apoiar decisões estratégicas relacionadas à alocação de recursos e ao emprego de sistemas de radar na defesa aérea britânica. Posteriormente, abordagens semelhantes foram adotadas pelos Estados Unidos, contribuindo para a otimização das operações navais durante a Batalha do Atlântico Norte (HILLIER; LIEBERMAN, 2013). Os resultados alcançados evidenciaram o potencial dos métodos analíticos na resolução de problemas complexos e impulsionaram o desenvolvimento da área.

Após o término da guerra, as técnicas desenvolvidas passaram a ser amplamente aplicadas em setores civis, contribuindo para avanços em áreas como planejamento da produção, gestão de estoques, transporte e alocação de recursos. Um marco importante nesse processo foi o desenvolvimento do Método Simplex por George Dantzig em 1947 (DANTZIG, 1963), consolidando a Programação Linear (PL) como uma das principais ferramentas da Pesquisa Operacional. Desde então, a evolução dos recursos computacionais e o surgimento de novas técnicas de otimização ampliaram significativamente o alcance e a relevância da área.

Na atualidade, a crescente complexidade dos sistemas organizacionais e a necessidade de tomadas de decisão rápidas e eficientes tornam a PO uma ferramenta estratégica em diversos setores (PETROPOULOS *et al.*, 2024). Suas aplicações abrangem desde o planejamento financeiro (FABOZZI *et al.*, 2007), a gestão logística e o gerenciamento da cadeia de suprimentos (SIMCHI-LEVI; KAMINSKY; SIMCHI-LEVI, 2008), até sistemas de transporte (CEDER, 2016), planejamento e controle da produção (PINEDO, 2016) e análise de dados por meio de técnicas de otimização e aprendizado de máquina (BERTSIMAS; DUNN, 2019). O avanço da capacidade computacional também ampliou significativamente seu campo de atuação, permitindo a integração de modelos matemáticos e técnicas de otimização a softwares especializados capazes de apoiar a análise, simulação e resolução de problemas

complexos.

Nesse cenário, a otimização combinatória ocupa posição de destaque dentro da Pesquisa Operacional (AARTS; LENSTRA, 2003). Essa área dedica-se ao estudo de problemas cuja quantidade de soluções viáveis cresce de forma combinatória, frequentemente exponencial, com o aumento do tamanho da instância, tornando inviável a obtenção de soluções ótimas por métodos exatos para problemas de grande porte (AARTS; LENSTRA, 2003). Em decorrência dessa elevada complexidade computacional, especialmente em problemas classificados como NP-difíceis, tornou-se necessário o desenvolvimento de métodos aproximativos, como heurísticas e metaheurísticas, capazes de produzir soluções de alta qualidade em tempos computacionais compatíveis com aplicações práticas.

Entre os problemas da otimização combinatória destaca-se o Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* - OSCM), introduzido por Peter Eades e Kozo Sugiyama, em 1990 (EADES; SUGIYAMA, 1990). O problema surge no contexto de grafos, especificamente nos grafos bipartidos. Nos grafos bipartidos, os vértices são divididos em dois conjuntos disjuntos (ou camadas disjuntas) X e Y, onde podem existir arestas que ligam um vértice de X com um vértice de Y, mas nunca existem arestas que ligam vértices do mesmo conjunto. O OSCM trata do desenho automático de grafos bipartidos em camadas e consiste em determinar uma ordenação para os vértices de uma camada livre, mantendo fixa a ordenação da outra camada, de forma a minimizar o número de cruzamentos entre as arestas do grafo.

O OSCM possui relevância tanto teórica quanto prática (DI BATTISTA *et al.*, 1999). O problema está diretamente relacionado ao desenho de grafos e possui aplicações em áreas como projeto de circuitos integrados (*Very Large Scale Integration* - VLSI), visualização hierárquica de grafos, engenharia de software e biologia computacional, nas quais a redução do número de cruzamentos é essencial para melhorar a legibilidade das representações e facilitar a interpretação das relações entre os elementos do sistema (LENGAUER, 1990).

Entretanto, o OSCM é classificado como um problema NP-difícil (JÜNGER; MUTZEL, 2002), o que inviabiliza a utilização de métodos exatos para instâncias de grande porte. Em virtude da complexidade computacional, diversas abordagens tem sido propostas na literatura para sua resolução. Entre os métodos destacam-se as heurísticas empregadas devido à sua eficiência na obtenção de soluções boa qualidade (EADES; SUGIYAMA, 1990).

Recentemente, o problema ganhou ainda mais relevância com sua inclusão no PACE (*Parameterized Algorithms and Computational Experiments*[1]) *Challenge*, competição internacional voltada à avaliação experimental de algoritmos para problemas computacionalmente difíceis. A edição de 2024 foi dedicada ao Problema de Minimização

[1] https://pacechallenge.org/2024/

de Cruzamentos Unilaterais, contemplando três categorias distintas: algoritmos exatos, algoritmos heurísticos e algoritmos parametrizados. Enquanto a trilha heurística priorizou a obtenção de soluções de alta qualidade em tempo limitado, as trilhas exata e parametrizada tiveram como objetivo a obtenção de soluções ótimas para instâncias com características estruturais específicas (KINDERMANN; KLUTE; TERZIADIS, 2024). Para a avaliação dos métodos propostos, o desafio disponibilizou conjuntos de instâncias públicas e privadas contendo 200 problemas em cada trilha, estabelecendo um *benchmark* padronizado que atualmente representa uma das principais referências para estudos experimentais envolvendo o OSCM.

1.1 Objetivos

1.1.1 Objetivo Geral

Investigar e avaliar o desempenho de heurísticas construtivas, métodos de busca local e da metaheurística *Simulated Annealing* aplicados ao *One-Sided Crossing Minimization* (OSCM), analisando sua capacidade de produzir soluções de alta qualidade para instâncias do *benchmark* oficial do PACE *Challenge* 2024.

1.1.2 Objetivos Específicos

- Realizar um levantamento bibliográfico sobre o problema;
- Implementar as heurísticas construtivas do Baricentro e da Mediana para geração de soluções iniciais;
- Desenvolver mecanismos de refinamento baseados nas vizinhanças *Swap*, *Insert* e *2-Opt*;
- Implementar a estratégia *Variable Neighborhood Descent* (VND);
- Implementar e calibrar automaticamente os parâmetros da metaheurística *Simulated Annealing* utilizando a ferramenta *iRace*;
- Comparar o desempenho das abordagens propostas por meio de métricas relacionadas à qualidade das soluções e ao tempo computacional;
- Avaliar os resultados obtidos em relação às melhores soluções conhecidas para o *benchmark* oficial do PACE *Challenge* 2024.

1.2 Organização e Estrutura do Trabalho

Além deste capítulo introdutório, o trabalho está organizado em outros cinco capítulos, que apresentam o referencial teórico, a caracterização do problema, a metodologia proposta, os resultados experimentais e as conclusões.

O Capítulo 2 apresenta o referencial teórico que fundamenta este trabalho. Inicialmente, são discutidos conceitos relacionados à Pesquisa Operacional, às heurísticas e metaheurísticas e ao desenho automático de grafos. Em seguida, é apresentado o Problema de Minimização de Cruzamentos Unilaterais, abordando os principais métodos propostos na literatura para sua resolução.

O Capítulo 3 apresenta os conceitos específicos relacionados ao Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* - OSCM). São descritos sua definição formal, a modelagem matemática, as principais aplicações e o *benchmark* oficial do *PACE Challenge* 2024, utilizado como base para a avaliação experimental.

O Capítulo 4 descreve a metodologia proposta e os algoritmos implementados neste trabalho. São apresentadas as representações computacionais utilizadas, a avaliação da função objetivo, as heurísticas construtivas do Baricentro e da Mediana, os métodos de busca local, a estratégia *Variable Neighborhood Descent* (VND), a metaheurística *Simulated Annealing* e o processo de calibração automática de parâmetros realizado por meio do pacote *iRace*.

O Capítulo 5 apresenta os experimentos computacionais e a análise dos resultados obtidos. Inicialmente, são avaliadas as heurísticas construtivas e os métodos de busca local. Em seguida, são apresentados os resultados obtidos com a metaheurística *Simulated Annealing*, incluindo comparações com soluções de referência e com os resultados do *benchmark* oficial do *PACE Challenge* 2024.

Por fim, o Capítulo 6 apresenta as conclusões desta pesquisa, os principais resultados obtidos ao longo do trabalho e as contribuições alcançadas. Além disso, são apresentadas perspectivas para trabalhos futuros, visando o aperfeiçoamento das abordagens propostas e a ampliação das investigações sobre o problema.

2 Revisão de Literatura

Para compreender os conceitos e métodos empregados na resolução do OSCM, este capítulo apresenta os fundamentos teóricos que sustentam a pesquisa. Inicialmente, são discutidos os principais conceitos relacionados à Pesquisa Operacional e à otimização combinatória. Em seguida, são abordados os fundamentos de heurísticas e metaheurísticas, técnicas amplamente utilizadas na resolução de problemas NP-difíceis e a Teoria dos Grafos. Por fim, são apresentados o Problema de Minimização de Cruzamentos Unilaterais e os aspectos relevantes do PACE *Challenge* 2024.

2.1 Pesquisa Operacional

A Pesquisa Operacional (PO) constitui uma área do conhecimento voltada ao apoio à tomada de decisão por meio da aplicação de métodos científicos, modelos matemáticos e técnicas computacionais. Seu principal objetivo é fornecer subsídios quantitativos para a análise e resolução de problemas complexos envolvendo a utilização eficiente de recursos limitados (TAHA, 2017).

Segundo Morse e Kimball (2003), a Pesquisa Operacional pode ser definida como "o uso do método científico para fornecer aos departamentos executivos uma base quantitativa para decisões relacionadas às operações sob seu controle"[1]. Essa definição evidencia o caráter aplicado da área, cuja finalidade é transformar problemas reais em modelos analíticos capazes de apoiar decisões mais eficientes.

De acordo com Taha (2017), a PO combina aspectos científicos e práticos. Seu caráter científico decorre da utilização de ferramentas matemáticas e estatísticas para modelagem e análise de problemas, enquanto sua dimensão prática está relacionada à capacidade do analista em representar adequadamente sistemas reais e interpretar os resultados obtidos. Dessa forma, a PO pode ser compreendida tanto como uma ciência quanto como uma arte de modelagem e tomada de decisão.

A evolução dos recursos computacionais ampliou significativamente o alcance da Pesquisa Operacional, permitindo o tratamento de problemas cada vez mais complexos. Atualmente, suas aplicações abrangem áreas como logística, transporte, planejamento da produção, gestão da cadeia de suprimentos, telecomunicações, energia, saúde, finanças e tecnologia da informação (HILLIER; LIEBERMAN, 2013). Em todos esses contextos, a PO busca identificar soluções que maximizem desempenho, minimizem custos ou equilibrem

[1] *"Operations research is a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control."*

múltiplos objetivos sujeitos a diferentes restrições.

Para atingir esses objetivos, a Pesquisa Operacional utiliza diversas técnicas de modelagem e otimização. Entre as principais destacam-se a Programação Linear, Programação Inteira, Programação Dinâmica, Teoria das Filas, Teoria dos Jogos, Simulação, métodos de otimização em redes e técnicas de otimização combinatória (TAHA, 2017). Muitas dessas abordagens possuem fundamentos matemáticos rigorosos e permitem a obtenção de soluções ótimas para determinados problemas.

Entretanto, diversos problemas relevantes apresentam elevada complexidade computacional, tornando inviável a utilização de métodos exatos em instâncias de grande porte. Nesses casos, tornam-se necessárias abordagens aproximadas, como heurísticas e metaheurísticas, capazes de produzir soluções de alta qualidade em tempos computacionais compatíveis. Entre os problemas que apresentam essa característica encontra-se o Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* – OSCM), objeto de estudo deste trabalho.

2.2 Etapas e Métodos de Resolução da Pesquisa Operacional

A PO fundamenta-se na aplicação de métodos científicos para apoiar a tomada de decisão em problemas complexos. Para atingir esse objetivo, a área utiliza um processo sistemático que transforma situações reais em modelos analíticos capazes de representar, analisar e otimizar sistemas sob diferentes restrições (HILLIER; LIEBERMAN, 2013).

De maneira geral, a aplicação da Pesquisa Operacional compreende um conjunto de etapas interdependentes que se inicia com a definição do problema e se estende até a implementação e o acompanhamento das soluções propostas. Embora diferentes autores apresentem pequenas variações quanto à organização dessas etapas, há consenso de que o processo possui caráter iterativo, permitindo que decisões tomadas em uma fase sejam revisadas e aperfeiçoadas à medida que novas informações são obtidas ou inconsistências são identificadas (RABENSCHLAG, 2005).

A Figura 1 ilustra esse processo de forma simplificada. Inicialmente, realiza-se a definição do problema e a construção do modelo matemático que representa o sistema em estudo. Em seguida, aplica-se um método de solução para obtenção dos resultados, os quais são submetidos a um processo de validação. Uma vez validado, o modelo pode ser implementado no sistema real e seus resultados passam a ser continuamente avaliados e monitorados. Caso sejam identificadas inconsistências ou oportunidades de melhoria, o processo retorna às etapas anteriores para ajustes, evidenciando o caráter cíclico da Pesquisa Operacional.

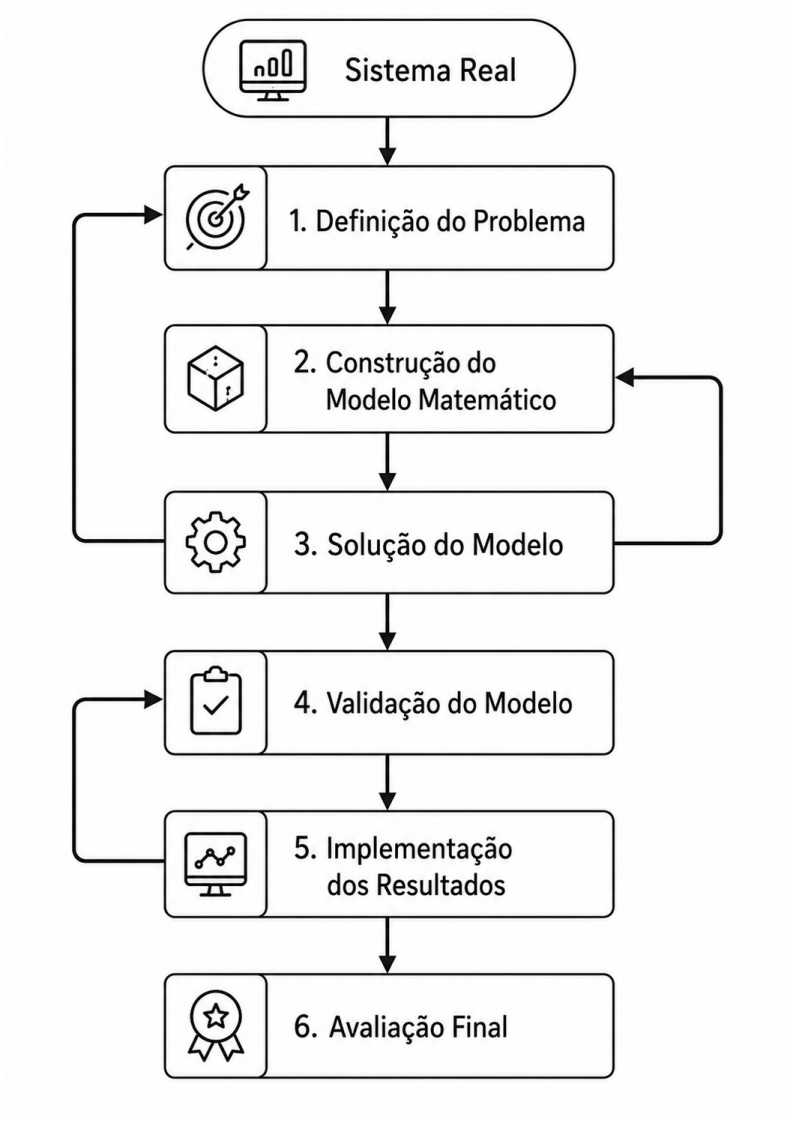

Figura 1 – Etapas de implementação da Pesquisa Operacional

Fonte: Elaborado pelo autor (2026), adaptado de (BELFIORE; FÁVERO, 2013)

Nas subseções seguintes, cada uma das etapas apresentadas na Figura 1 é descrita com maior detalhamento, destacando seus objetivos e sua importância no processo de resolução de problemas por meio da Pesquisa Operacional.

2.2.1 Definição do problema

A definição do problema constitui a etapa inicial e uma das mais importantes da Pesquisa Operacional. Nessa fase, busca-se compreender o sistema em estudo, identificar seus objetivos, restrições e variáveis relevantes, além de delimitar claramente o escopo da análise (MARINS, 2009). Segundo Lisboa (2002), uma adequada definição do problema deve contemplar três elementos fundamentais:

- Definição dos objetivos a serem alcançados;
- Identificação das alternativas de decisão disponíveis;
- Levantamento das restrições e condições que limitam o sistema.

Conforme destacam Hillier e Lieberman (2013), falhas nessa etapa podem comprometer todas as fases subsequentes, conduzindo a soluções inadequadas mesmo quando os modelos matemáticos são corretamente resolvidos.

2.2.2 Construção do modelo matemático

Após a definição do problema, desenvolve-se um modelo matemático capaz de representar os principais elementos do sistema real. Segundo Belfiore e Fávero (2013), um modelo matemático consiste em uma abstração formal da realidade, construída com o objetivo de analisar cenários, compreender o comportamento do sistema e apoiar o processo de tomada de decisão.

A construção desse modelo inicia-se pela identificação dos aspectos mais relevantes do problema, selecionando apenas as características essenciais para sua representação matemática. Como ilustrado na Figura 2, o modelo não representa toda a complexidade do mundo real, mas sim uma parcela desse sistema considerada relevante para o objetivo da análise. Dessa forma, busca-se simplificar a realidade sem comprometer a capacidade do modelo de representar adequadamente o problema estudado.

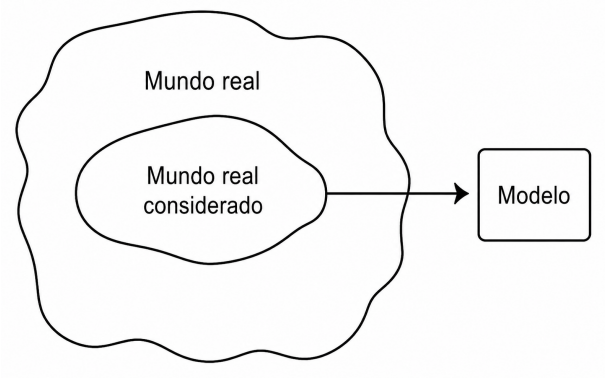

Figura 2 – Processo de abstração na construção de um modelo matemático

Fonte: Elaborado pelo autor (2026), adaptado de (GATTI JUNIOR, 2020)

Uma vez definido o escopo do sistema a ser representado, o modelo matemático é estruturado a partir de três componentes fundamentais (HILLIER; LIEBERMAN, 2013):

- Variáveis de decisão, que representam as escolhas ou ações sob controle do tomador de decisão;

- Função objetivo, responsável por quantificar o desempenho do sistema, expressando o critério a ser maximizado ou minimizado;
- Restrições, que definem as limitações operacionais, físicas ou lógicas do problema, delimitando o conjunto de soluções viáveis.

A modelagem exige um equilíbrio entre fidelidade à realidade e simplicidade matemática. Modelos excessivamente detalhados podem tornar-se difíceis de resolver, enquanto simplificações exageradas podem comprometer sua capacidade de representar adequadamente o sistema analisado (TAHA, 2017). Assim, o sucesso de um modelo depende da seleção criteriosa dos elementos que influenciam o problema, preservando apenas as características essenciais para a obtenção de soluções úteis e confiáveis.

2.2.3 Solução do modelo

Uma vez formulado o modelo, aplica-se um método de solução adequado às características do problema. Dependendo da estrutura matemática envolvida, podem ser utilizados algoritmos exatos ou métodos aproximados (TAHA, 2017). Entre os principais métodos empregados na Pesquisa Operacional destacam-se:

- Método *Simplex*, amplamente utilizado em problemas de Programação Linear (CHAVES, 2011);
- Algoritmos *Branch-and-Bound*, empregados na resolução de problemas de Programação Inteira (BELFIORE; FÁVERO, 2013);
- Heurísticas e metaheurísticas, utilizadas quando a elevada complexidade computacional inviabiliza a obtenção da solução ótima por métodos exatos (TAHA, 2017).

Atualmente, diversos software de otimização, como CPLEX[2], Gurobi[3], GLPK[4] e LINDO[5], integram diferentes técnicas de otimização e desempenham papel fundamental na resolução de modelos matemáticos encontrados em problemas reais (MARINS, 2009).

2.2.4 Validação do modelo

A validação consiste em verificar se o modelo desenvolvido representa adequadamente o sistema real e se seus resultados são coerentes com o comportamento observado (RABENSCHLAG, 2005). Segundo Taha (2017), a validação pode envolver diferentes procedimentos, entre eles:

[2] https://www.ibm.com/br-pt/products/ilog-cplex-optimization-studio
[3] https://www.gurobi.com/
[4] https://winglpk.sourceforge.net/
[5] https://www.lindo.com/

- Comparação dos resultados obtidos com dados históricos;
- Verificação da consistência lógica das relações modeladas;
- Análise do comportamento do modelo sob diferentes cenários.

Caso sejam identificadas inconsistências, o processo retorna às etapas anteriores para ajustes e refinamentos, evidenciando o caráter iterativo da Pesquisa Operacional.

2.2.5 Implementação dos resultados

Após a validação, as soluções propostas são aplicadas ao sistema real. Essa etapa envolve a transformação dos resultados analíticos em ações práticas, podendo exigir adaptações organizacionais, investimentos e mudanças operacionais (RABENSCHLAG, 2005). Conforme destacam Hillier e Lieberman (2013), mesmo soluções matematicamente ótimas possuem utilidade limitada caso não sejam implementáveis ou aceitas pelo ambiente organizacional.

2.2.6 Avaliação e monitoramento

A etapa final consiste em avaliar os resultados obtidos após a implementação da solução e verificar se os objetivos inicialmente estabelecidos foram alcançados. Além disso, busca-se identificar oportunidades de melhoria e eventuais necessidades de ajustes (RABENSCHLAG, 2005). Dessa forma, a Pesquisa Operacional deve ser compreendida como um processo contínuo de análise, implementação e aperfeiçoamento, permitindo que os modelos evoluam juntamente com as mudanças observadas no sistema estudado.

2.3 Heurísticas e Metaheurísticas

O crescimento da complexidade dos problemas de otimização enfrentados pela PO impulsionou o desenvolvimento de métodos alternativos aos algoritmos exatos tradicionais. Em muitos problemas de otimização combinatória, especialmente aqueles classificados como NP-difíceis, o tempo computacional necessário para garantir a obtenção da solução ótima cresce rapidamente com o tamanho da instância, tornando inviável sua aplicação em situações práticas. Nesse contexto, heurísticas e metaheurísticas surgem como abordagens capazes de produzir soluções de alta qualidade em tempos computacionais compatíveis (BELFIORE; FÁVERO, 2013).

O termo *heurística* deriva do verbo grego *heurísko*, que significa “encontrar” ou “descobrir” (BALIEIRO FILHO, 2017). De maneira geral, heurísticas podem ser definidas como procedimentos baseados em regras, critérios ou estratégias que orientam a busca por soluções satisfatórias para problemas complexos. Conforme destaca Ferreira (1997),

uma heurística corresponde a um conjunto de métodos voltados à descoberta e à resolução eficiente de problemas.

As heurísticas podem ser classificadas em diferentes categorias. Uma das classificações mais difundidas divide essas abordagens em heurísticas construtivas e heurísticas de melhoria. As heurísticas construtivas constroem uma solução gradativamente a partir de uma estrutura inicialmente vazia. A cada etapa, novos elementos são inseridos na solução de acordo com critérios previamente estabelecidos, até que uma solução completa e viável seja obtida. Em geral, apresentam baixo custo computacional e são amplamente utilizadas para gerar soluções iniciais para métodos mais sofisticados.

Por sua vez, as heurísticas de melhoria, também conhecidas como métodos de busca local, partem de uma solução inicial previamente construída e realizam modificações sucessivas em sua estrutura com o objetivo de melhorar seu valor de função objetivo. Essas modificações são realizadas por meio de movimentos definidos em uma vizinhança da solução corrente. O processo continua enquanto forem encontradas soluções melhores, encerrando-se quando nenhum movimento vizinho produz melhoria adicional (GLOVER; KOCHENBERGER, 2003). Embora frequentemente produzam resultados expressivos, as heurísticas de busca local apresentam uma limitação importante: a tendência de convergir para ótimos locais. Como consequência, a qualidade da solução final pode depender fortemente da solução inicial utilizada (ARROYO, 2002).

Com o objetivo de superar essa limitação, surgiram as metaheurísticas. Essas abordagens podem ser definidas como estratégias de otimização capazes de coordenar e orientar procedimentos heurísticos de forma mais eficiente. Em vez de serem projetadas para um problema específico, as metaheurísticas fornecem mecanismos genéricos de exploração e intensificação da busca, podendo ser adaptadas a diferentes problemas de otimização (SUCUPIRA, 2004). Segundo Talbi (2009), uma das principais características das metaheurísticas é sua capacidade de equilibrar dois objetivos complementares: a exploração de novas regiões do espaço de busca (*diversification*) e a intensificação da busca em regiões promissoras (*intensification*). Esse equilíbrio permite reduzir a probabilidade de aprisionamento em ótimos locais e aumentar as chances de obtenção de soluções melhores.

As metaheurísticas ainda podem ser classificadas em duas grandes categorias: metaheurísticas baseadas em trajetória e metaheurísticas baseadas em população. As metaheurísticas baseadas em trajetória operam sobre uma única solução que é modificada iterativamente ao longo do processo de busca. Nessa categoria destacam-se métodos como o *Simulated Annealing* e a Busca Tabu, que incorporam mecanismos específicos para escapar de ótimos locais e explorar novas regiões do espaço de busca (BLUM; ROLI, 2003). Já as metaheurísticas baseadas em população trabalham simultaneamente com um conjunto de soluções, explorando mecanismos inspirados em fenômenos naturais ou comportamentos coletivos. Exemplos incluem os Algoritmos Genéticos (AG) e os Algoritmos de Colônia

de Formigas (*Ant Colony Optimization* - ACO), amplamente utilizados em problemas de otimização combinatória (BLUM; ROLI, 2003).

A Figura 3 apresenta uma visão geral da classificação dos principais métodos empregados em problemas de otimização. Em um primeiro nível, esses métodos podem ser divididos em algoritmos exatos e heurísticos. Os algoritmos exatos, como o método *Simplex* e o *Branch-and-Bound*, são capazes de garantir a obtenção da solução ótima do problema. Entretanto, seu custo computacional pode crescer rapidamente à medida que o tamanho das instâncias aumenta, limitando sua aplicação em problemas de grande porte.

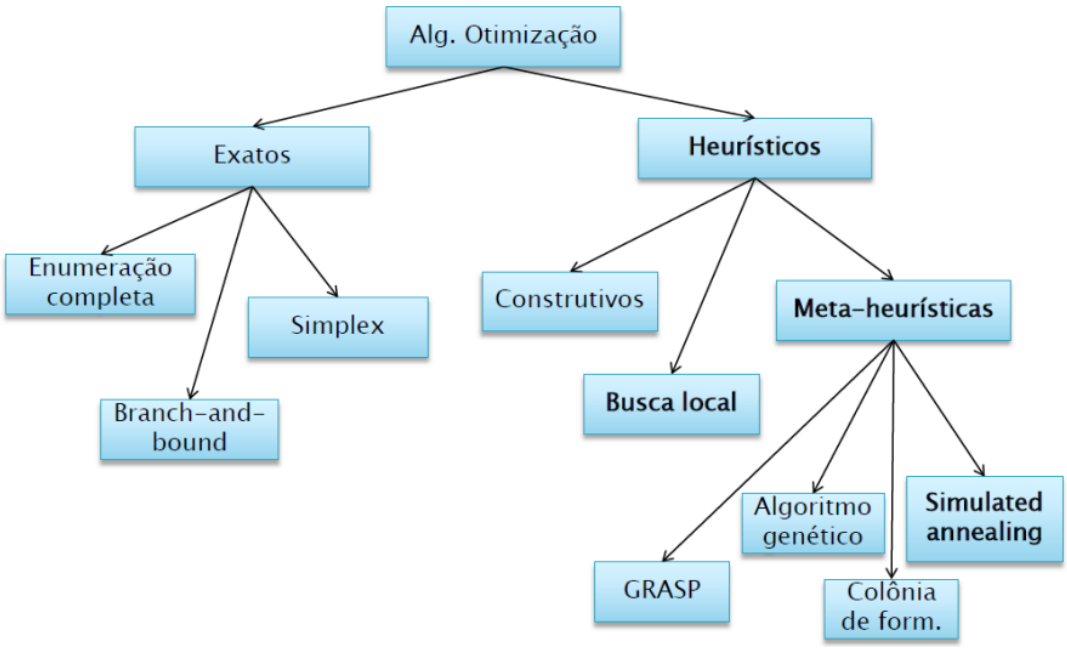

Figura 3 – Algoritmos de otimização

Fonte: (BELAN, 2024)

Por outro lado, os métodos heurísticos não garantem a obtenção da solução ótima. Seu principal objetivo é encontrar soluções viáveis e de boa qualidade com baixo esforço computacional. Essa característica torna essas abordagens especialmente atrativas em problemas de grande porte, nos quais a busca exaustiva ou a aplicação de algoritmos exatos se tornam impraticáveis (BUENO, 2009).

Conforme ilustrado na Figura 3, as metaheurísticas compreendem um conjunto amplo de estratégias de otimização, podendo ser por diferentes paradigmas, como o *Greedy Randomized Adaptive Search Procedure* (GRASP), os Algoritmos Genéticos, os Algoritmos de Colônia de Formigas (*Ant Colony Optimization* - ACO) e o *Simulated Annealing* (SA). Essas abordagens são mecanismos empregados para explorar o espaço de busca e intensificar a busca em regiões promissoras, características fundamentais para a obtenção de soluções de elevada qualidade em problemas de otimização combinatória.

Nas últimas décadas, heurísticas e metaheurísticas consolidaram-se como ferramentas fundamentais para a resolução de problemas complexos em áreas como engenharia, logística, telecomunicações, manufatura, redes de computadores e ciência de dados. Essa ampla utilização decorre da capacidade desses métodos de produzir soluções de alta qualidade para problemas caracterizados por espaços de busca muito extensos, nos quais a aplicação de métodos exatos se torna computacionalmente inviável. Em razão dessa versatilidade, as metaheurísticas permanecem como uma das principais linhas de pesquisa em otimização combinatória, sendo continuamente adaptadas e aperfeiçoadas para atender a novos desafios e aplicações (TALBI, 2009).

No contexto deste trabalho, as heurísticas construtivas são utilizadas para gerar soluções iniciais para o Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* - OSCM), enquanto as heurísticas de busca local são aplicadas para realizar o refinamento dessas soluções por meio de diferentes movimentos de vizinhança. Complementarmente, o *Simulated Annealing*, uma metaheurística baseada em trajetória, é empregado com o objetivo de ampliar a exploração do espaço de busca e reduzir a probabilidade de convergência prematura em ótimos locais. Dessa forma, as abordagens apresentadas nesta seção constituem a fundamentação metodológica dos métodos desenvolvidos neste trabalho.

2.4 Teoria dos Grafos

A Teoria dos Grafos constitui um ramo da matemática discreta dedicado ao estudo de estruturas formadas por vértices e arestas, sendo amplamente utilizada para modelar relações entre elementos em diversos contextos científicos e tecnológicos. Considera-se como marco inicial dessa área a solução do Problema das Pontes de *Konigsberg* por Leonhard Euler, em 1736 (EULER, 1736). Desde então, a Teoria dos Grafos passou a desempenhar papel fundamental na representação e análise de sistemas complexos, encontrando aplicações em áreas como ciência da computação, engenharia, biologia computacional, logística e redes de comunicação (BONDY; MURTY, 2008).

Um grafo é uma estrutura matemática utilizada para representar relações entre elementos de um conjunto (NICOLETTI, 2018). Formalmente, é representado por:

$$G = (V, E), \tag{1}$$

em que V representa o conjunto de vértices e E o conjunto de arestas que conectam pares de vértices (DIESTEL, 2017). Dependendo da natureza do problema modelado, as arestas podem possuir orientação, pesos ou outras características específicas.

A Figura 4 apresenta um exemplo de um grafo direcionado, constituído por um

conjunto de vértices interligados por arestas. Nesse tipo de representação, cada aresta possui um sentido, indicado por uma seta, representando uma relação orientada entre o vértice de origem e o vértice de destino. O vértice identificado pelo número 4 é utilizado para exemplificar um elemento do conjunto V, enquanto a aresta destacada ilustra uma relação pertencente ao conjunto E. Essa representação gráfica facilita a visualização da estrutura do grafo e a compreensão dos conceitos fundamentais empregados na modelagem de problemas computacionais, teoria dos grafos e otimização.

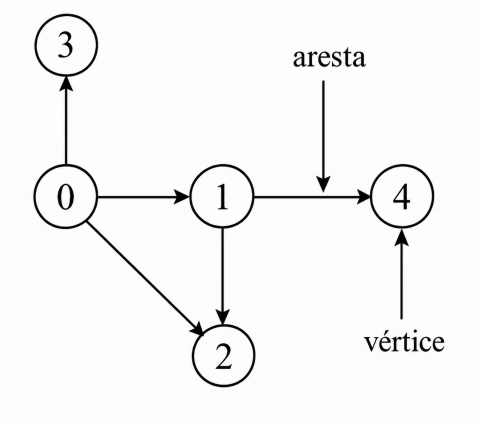

Figura 4 – Grafo direcionado

Fonte: Elaborado pelo autor (2026), adaptado de (LIMA, E. S. D., 2017)

Os grafos são amplamente e sua capacidade de representar relações de forma estruturada torna essa ferramenta particularmente importante para o desenvolvimento de algoritmos e métodos de análise computacional.

2.4.1 Grafos Bipartidos

Um dos tipos mais importantes de grafos para este trabalho é o grafo bipartido. Um grafo é classificado como bipartido quando seu conjunto de vértices pode ser particionado em dois subconjuntos disjuntos X e Y, de modo que todas as arestas conectem exclusivamente vértices pertencentes a conjuntos distintos (LIMA, G. C. G. D., 2017). Formalmente,

$$V = X \cup Y \tag{2}$$

$$X \cap Y = \emptyset \tag{3}$$

Além disso, para toda aresta $(x, y) \in E$, tem-se que $x \in X$ e $y \in Y$.

Os grafos bipartidos são frequentemente utilizados para representar relações entre dois conjuntos distintos de elementos, como tarefas e recursos, estudantes e disciplinas, usuários e produtos ou genes e proteínas. Devido à sua capacidade de modelar naturalmente relações entre conjuntos disjuntos, constituem uma importante classe de grafos com aplicações em problemas de otimização, escalonamento, sistemas de recomendação, bioinformática e ciência de dados (DIESTEL, 2017).

2.4.2 Desenho de Grafos em Camadas

O desenho automático de grafos (*Graph Drawing*) consiste no desenvolvimento de métodos para representar grafos de forma clara e organizada. Entre os diversos modelos existentes, destaca-se o desenho em camadas (*layered graph drawing*), amplamente empregado na visualização de grafos direcionados e bipartidos (DI BATTISTA *et al.*, 1999).

Nesse modelo, os vértices são distribuídos em níveis ou camadas paralelas, enquanto as arestas são representadas por segmentos que conectam vértices pertencentes a camadas distintas. O principal objetivo é produzir representações visualmente legíveis, reduzindo ambiguidades e facilitando a interpretação da estrutura do grafo.

Um dos critérios mais importantes para a qualidade de um desenho em camadas é a minimização do número de cruzamentos entre arestas. Estudos mostram que representações com menor quantidade de cruzamentos apresentam maior legibilidade e facilitar a compreensão das relações existentes no grafo (PURCHASE, 2002).

A Figura 5 ilustra um exemplo de grafo bipartido em camadas, evidenciando a ocorrência de cruzamentos entre arestas.

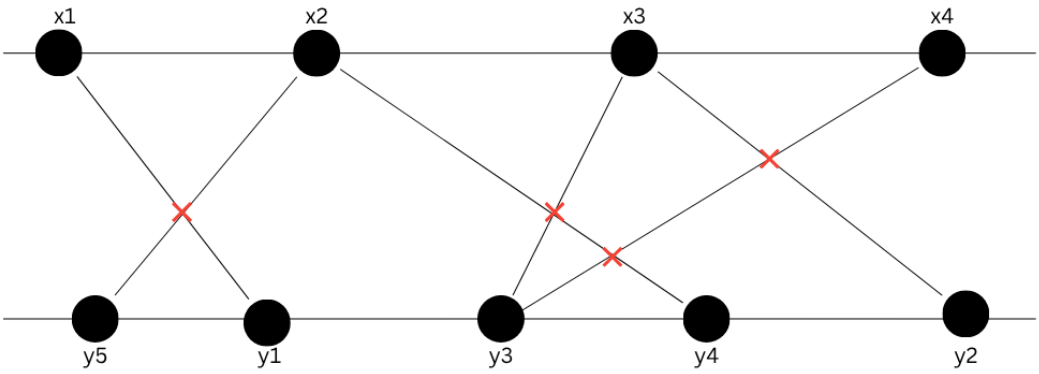

Figura 5 – Grafo bipartido em camadas

Fonte: Elaborado pelo autor (2026)

A minimização desses cruzamentos é um dos desafios do desenho de grafos e dá origem a diversos problemas de otimização combinatória, por exemplo o OSCM.

3 O *One-Sided Crossing Minimization* (OSCM)

O Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* - OSCM) é um problema da área de desenho de grafos, amplamente estudado no contexto da visualização de grafos bipartidos e grafos organizados em camadas (*layered graphs*). De modo geral, a minimização de cruzamentos entre arestas é considerada um dos principais critérios de qualidade em representações gráficas, uma vez que a presença excessiva de interseções compromete a legibilidade, a interpretação visual e a compreensão da estrutura representada (TAMASSIA, 2013).

A importância desse critério foi reconhecida desde os primeiros estudos sobre desenho automático de grafos. Trabalhos experimentais mostraram que o número de cruzamentos exerce influência mais significativa sobre a compreensão humana do que outros aspectos estéticos, como comprimento das arestas, uniformidade dos espaçamentos ou simetria do desenho (PURCHASE, 2002). Como consequência, a redução de cruzamentos tornou-se um dos objetivos centrais dos algoritmos modernos de visualização de grafos.

Problemas dessa natureza possuem aplicações em diversas áreas, incluindo projeto de circuitos integrados (*Very Large Scale Integration* - VLSI), visualização de software, bioinformática, engenharia de requisitos e desenho hierárquico de grafos direcionados (LENGAUER, 1990). Em particular, no modelo proposto por (SUGIYAMA; TAGAWA; TODA, 1981) para desenho de grafos direcionados acíclicos, a etapa de ordenação dos vértices em cada camada é responsável pela maior parte da redução de cruzamentos observada no layout final.

Nesse contexto, o OSCM surge como um subproblema fundamental. Seu objetivo consiste em determinar a melhor ordenação dos vértices de uma camada, mantendo fixa a ordenação da camada adjacente, de modo a minimizar o número de cruzamentos entre as arestas. Devido à sua relevância prática e à sua elevada complexidade computacional, o problema tornou-se objeto de intensa investigação na literatura, sendo frequentemente utilizado como componente básico de algoritmos mais gerais para minimização de cruzamentos em grafos multicamadas (EADES; SUGIYAMA, 1990).

3.1 Definição Formal do Problema

O OSCM é definido sobre um grafo bipartido G, em que o conjunto de vértices Y é particionado em dois subconjuntos disjuntos X e Y, conforme estabelecido pelas

equações 2 e 3, sendo $E \subseteq X \times Y$ o conjunto de arestas que conectam vértices pertencentes às duas partições.

No problema, assume-se que os vértices da camada X possuem uma ordenação linear fixa, enquanto os vértices da camada Y podem ser reordenados livremente. O modelo geométrico adotado consiste em posicionar os vértices de X e Y sobre duas retas paralelas distintas, respeitando suas respectivas ordenações, e representar cada aresta por meio de um segmento de reta ligando seus vértices extremos (DI BATTISTA *et al.*, 1999).

Duas arestas são consideradas cruzadas quando a ordem relativa de seus vértices na camada fixa é inversa à ordem relativa de seus vértices na camada livre. Assim, o objetivo do OSCM consiste em determinar uma permutação dos vértices de Y que minimize o número total de cruzamentos induzidos pelas arestas do grafo.

A Figura 6 ilustra um exemplo do OSCM. Em ambos os desenhos, a ordenação dos vértices da camada superior (A) permanece fixa, enquanto apenas a ordenação dos vértices da camada inferior (B) é modificada. A disposição inicial dos vértices da camada livre resulta em 33 cruzamentos entre as arestas. Após uma nova permutação dos vértices da camada B reduz esse número para apenas 17 cruzamentos. Esse exemplo evidencia que a escolha da ordenação da camada livre exerce influência direta sobre a quantidade de cruzamentos, justificando o objetivo do OSCM de determinar a permutação que minimize esse valor.

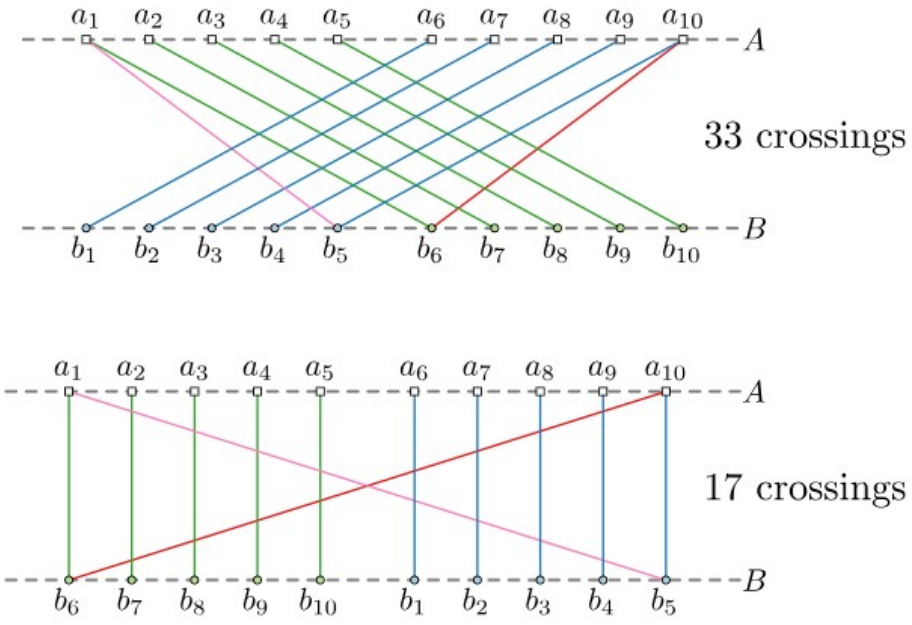

Figura 6 – Exemplo do Problema de Minimização de Cruzamentos Unilaterais (OSCM). (a) Ordenação inicial da camada livre, produzindo 33 cruzamentos. (b) Nova ordenação da camada livre, reduzindo o número de cruzamentos para 17

Fonte: (PACE CHALLENGE, 2024)

Esse problema diferencia-se do Problema de Minimização de Cruzamentos em Duas Camadas (*Two-Sided Crossing Minimization*), no qual as ordenações de ambas as camadas podem ser modificadas simultaneamente (GAREY; JOHNSON, 1983). Embora a fixação de uma das camadas reduza o espaço de busca, o OSCM permanece um problema computacionalmente difícil e de grande relevância para aplicações práticas de visualização de grafos.

3.2 Modelo Matemático e Função Objetivo

O objetivo do Problema de Minimização de Cruzamentos Unilaterais consiste em determinar uma permutação π dos vértices da camada livre Y que minimize o número total de cruzamentos entre as arestas do grafo bipartido. Para formalizar esse objetivo, considere duas arestas (x_1, y_1) e (x_2, y_2), em que $x_1, x_2 \in X$ e $y_1, y_2 \in Y$. Assumindo que a ordenação dos vértices da camada X é fixa, essas arestas produzem um cruzamento se, e somente se,

$$x(x_1) < x(x_2) \quad \text{e} \quad \pi(y_1) > \pi(y_2), \tag{4a}$$

ou, equivalentemente,

$$x(x_1) > x(x_2) \quad \text{e} \quad \pi(y_1) < \pi(y_2). \tag{4b}$$

onde $x(x)$ representa a posição fixa do vértice $x \in X$ e $\pi(y)$ representa a posição ocupada pelo vértice $y \in Y$ na ordenação considerada. Com base nessa definição, é possível pré-computar uma matriz de cruzamentos $C = (c_{ij})$, na qual cada elemento c_{ij} representa o número de cruzamentos gerados quando o vértice na posição i é posicionado antes do vértice na posição j na ordenação da camada livre. Dessa forma, cada par ordenado de vértices da camada Y contribui com um custo associado ao número de cruzamentos induzidos por essa relação de precedência.

Seja π uma permutação dos vértices de Y. A função objetivo do OSCM pode então ser expressa por:

$$f(\pi) = \sum_{i=1}^{|Y|-1} \sum_{j=i+1}^{|Y|} c_{\pi(i)\pi(j)} \tag{5}$$

sendo $f(\pi)$ o número total de cruzamentos produzidos pela ordenação π. O problema consiste em encontrar a permutação que minimize esse valor:

$$\min_{\pi \in \Pi(Y)} f(\pi),$$

onde $\Pi(Y)$ denota o conjunto de todas as permutações possíveis dos vértices da camada livre.

Essa formulação evidencia que o OSCM pode ser interpretado como um problema de ordenação combinatória, no qual o custo total de uma solução depende das relações de precedência estabelecidas entre pares de vértices. Sob essa perspectiva, o problema apresenta forte relação com problemas clássicos de agregação de rankings e ordenação com custos assimétricos (DWORK *et al.*, 2001). Do ponto de vista computacional, o cálculo direto do número de cruzamentos exige a verificação de todos os pares de arestas, resultando em complexidade $O(|E|^2)$.

Além de reduzir o custo de avaliação de uma solução, a utilização da matriz de cruzamentos pré-computada permite realizar atualizações incrementais da função objetivo durante a execução de heurísticas e metaheurísticas, tornando viável a exploração eficiente de grandes espaços de busca.

3.3 Complexidade Computacional

O Problema de Minimização de Cruzamentos Unilaterais é reconhecido como um problema NP-difícil. Esse resultado foi demonstrado por Garey e Johnson (1983) e posteriormente reforçado por estudos que mostraram que a dificuldade do problema permanece mesmo sob restrições estruturais significativas do grafo de entrada (MUÑOZ; UNGER; VRT'O, 2001). Dessa forma, a complexidade do OSCM não está associada apenas ao tamanho ou à densidade do grafo, mas à própria natureza combinatória da ordenação dos vértices da camada livre.

A classificação como NP-difícil implica que não existem algoritmos capazes de resolver todas as instâncias do problema em tempo polinomial. Consequentemente, a utilização de métodos exatos torna-se inviável para instâncias de grande porte, uma vez que o esforço computacional cresce rapidamente com o aumento do número de vértices e arestas.

Além disso, a dificuldade do OSCM está diretamente relacionada ao tamanho do espaço de busca. Considerando uma camada livre com $|Y|$ vértices, existem $|Y|!$ possíveis ordenações. Esse crescimento fatorial torna impraticável a enumeração completa das soluções, mesmo para instâncias de dimensão moderada.

Apesar dessas limitações, o problema apresenta propriedades que podem ser exploradas sob a ótica da complexidade parametrizada. Nesse contexto, foram desenvolvidos algoritmos eficientes para casos em que determinados parâmetros estruturais do problema permanecem pequenos, como o número de cruzamentos na solução ótima ou medidas específicas da estrutura do grafo (CYGAN *et al.*, 2015). Esses resultados motivaram o

desenvolvimento de algoritmos *Fixed-Parameter Tractable* (FPT), bem como de abordagens exatas baseadas em *Branch-and-Bound* e formulações de Programação Inteira (PI) (KINDERMANN; KLUTE; TERZIADIS, 2024).

Entretanto, para instâncias de grande porte, a aplicação de métodos exatos continua sendo limitada. Por essa razão, a literatura tem concentrado esforços no desenvolvimento de heurísticas e metaheurísticas capazes de produzir soluções de alta qualidade em tempos computacionais reduzidos. Entre as abordagens mais utilizadas destacam-se as heurísticas do Baricentro e da Mediana, amplamente empregadas em sistemas de desenho automático de grafos devido à sua simplicidade e eficiência computacional (EADES; SUGIYAMA, 1990; DI BATTISTA *et al.*, 1999).

Assim, a elevada complexidade computacional do OSCM justifica o crescente interesse no desenvolvimento de métodos aproximativos e estratégias de busca avançadas, tornando o problema um importante campo de investigação dentro da otimização combinatória e da visualização de grafos.

3.4 Aplicações Práticas

A relevância do Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* – OSCM) transcende o interesse puramente teórico, estando diretamente relacionada a diversas aplicações em engenharia, visualização da informação, ciência de dados, bioinformática e desenvolvimento de software. Em todos esses contextos, grafos bipartidos ou estruturas em camadas são empregados para representar relações entre diferentes entidades, de modo que a redução do número de cruzamentos contribui para melhorar a legibilidade, a interpretabilidade e a eficiência cognitiva das representações gráficas (PURCHASE, 2002). Além de favorecer a compreensão visual, essa redução simplifica atividades de projeto, análise, manutenção e tomada de decisão, justificando o contínuo interesse da comunidade científica pelo problema, evidenciado por iniciativas recentes como o *PACE Challenge* 2024 (SEGURA *et al.*, 2024).

Uma das aplicações mais tradicionais do OSCM encontra-se no projeto físico de circuitos eletrônicos, especialmente em circuitos integrados (*Very Large Scale Integration* - VLSI) e placas de circuito impresso (*Printed Circuit Boards* - PCB). Nesses projetos, grafos são utilizados para modelar as interconexões entre componentes eletrônicos, enquanto algoritmos de minimização de cruzamentos auxiliam na obtenção de roteamentos mais organizados, reduzindo conflitos entre trilhas, interferências eletromagnéticas, custos de fabricação e dificuldades de implementação física (TAMASSIA, 2013).

Outra importante área de aplicação corresponde à engenharia de software e à visualização de sistemas complexos. Diagramas de Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML), grafos de dependência entre módulos, arquiteturas

de software e grafos de chamadas de funções frequentemente apresentam elevado grau de conectividade. Nesses cenários, a redução dos cruzamentos facilita a compreensão da estrutura dos sistemas, melhora a identificação de dependências e simplifica atividades de manutenção e evolução do software (DI BATTISTA *et al.*, 1999).

Aplicações relevantes também são encontradas na visualização de dados e na ciência da informação. Grafos bipartidos são amplamente empregados para representar relacionamentos entre usuários e produtos em sistemas de recomendação, autores e publicações em redes bibliométricas, documentos e termos em bases textuais, entre diversas outras aplicações. Nesses cenários, uma ordenação adequada dos vértices melhora significativamente a clareza visual das representações e favorece a interpretação das relações existentes entre os dados (DWORK *et al.*, 2001). Além disso, grafos em camadas são frequentemente empregados na modelagem de processos organizacionais, cadeias de suprimentos e redes de atividades em gerenciamento de projetos. A redução do número de cruzamentos nesses diagramas facilita a análise de dependências, a identificação de gargalos e a comunicação entre equipes responsáveis pelo planejamento e execução das atividades.

O OSCM também desempenha papel central nos algoritmos de desenho hierárquico de grafos, particularmente no modelo proposto por Sugiyama, Tagawa e Toda (SUGIYAMA; TAGAWA; TODA, 1981). Nesse método, a etapa de ordenação dos vértices em cada camada é normalmente tratada por meio de sucessivas instâncias do OSCM. Dessa forma, melhorias na resolução desse problema refletem diretamente na qualidade visual dos desenhos produzidos (EADES; SUGIYAMA, 1990).

Mais recentemente, o avanço da Indústria 4.0, dos sistemas ciberfísicos (*Cyber-Physical Systems* - CPS), dos gêmeos digitais (*Digital Twins*) e das plataformas de engenharia colaborativa ampliou a necessidade de técnicas capazes de produzir representações gráficas organizadas e de fácil interpretação (LAMPROPOULOS; SIAKAS, 2023). Nesse contexto, algoritmos de desenho de grafos e técnicas de minimização de cruzamentos contribuem para gerar diagramas mais legíveis, facilitar a análise de grandes volumes de informações e apoiar atividades de projeto, monitoramento, manutenção e tomada de decisão.

A Tabela 1 apresenta um resumo das principais aplicações do OSCM e dos benefícios proporcionados pela minimização do número de cruzamentos.

Tabela 1 – Principais aplicações do OSCM

Área	**Exemplo de aplicação**	**Benefícios da minimização de cruzamentos**
Automação industrial	Painéis elétricos, CLPs, sensores e atuadores.	Redução do cruzamento de cabos, facilidade de montagem e manutenção.
Ciência da informação	Redes bibliométricas, sistemas de recomendação e mineração de texto.	Representações mais organizadas e melhor interpretação dos dados.
Circuitos eletrônicos (VLSI/PCB)	Roteamento de trilhas e interconexões entre componentes.	Redução da complexidade do roteamento, dos custos de fabricação e das interferências eletromagnéticas.
Desenho hierárquico de grafos	Método de *Sugiyama*.	Melhoria da qualidade visual dos desenhos gerados automaticamente.
Engenharia de software	Diagramas UML, grafos de dependência e arquiteturas de software.	Melhor compreensão da estrutura do sistema e facilidade de manutenção.
Redes industriais	PROFINET, *EtherNet*/IP, PROFIBUS e *AS-Interface*.	Simplificação do roteamento físico dos cabos e aumento da confiabilidade da comunicação.
Sistemas supervisórios	Diagramas FBD, *Grafcet*, Redes de Petri e SCADA.	Melhor legibilidade, diagnóstico mais rápido e maior facilidade de operação.

A Figura 7 apresenta uma placa de circuito impresso (*Printed Circuit Board* - PCB), na qual as trilhas condutoras estabelecem as conexões entre diferentes componentes eletrônicos. Ela ilustra uma aplicação clássica das técnicas de desenho de grafos e otimização empregadas no projeto de circuitos eletrônicos, especialmente em problemas de roteamento associados ao projeto de circuitos integrados (*Very Large Scale Integration* - VLSI).

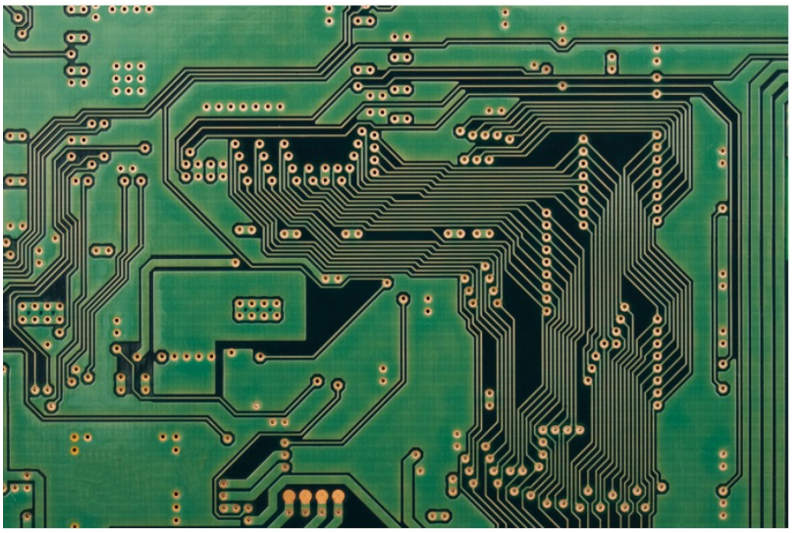

Figura 7 – Placa de Circuito Impresso (PCB), ilustrando uma aplicação de técnicas de otimização no projeto de circuitos eletrônicos

Fonte: (MAGALHÃES, 2024)

Nesse tipo de aplicação, a disposição das conexões influencia diretamente a complexidade do roteamento, o aproveitamento da área disponível e a qualidade do projeto. Embora o desenvolvimento de circuitos eletrônicos envolva diversas restrições adicionais, a redução de cruzamentos entre conexões constitui um objetivo importante, pois contribui para a obtenção de layouts mais organizados, facilita o processo de fabricação e pode reduzir custos e melhorar o desempenho do sistema.

Na Engenharia de Controle e Automação, problemas análogos ao OSCM também surgem durante o projeto de painéis elétricos e o roteamento de cabos entre sensores, atuadores, módulos de entradas e saídas e Controladores Lógicos Programáveis (CLPs). Em geral, a posição dos dispositivos de campo é determinada pelas restrições físicas do processo industrial, enquanto a disposição dos módulos e bornes pode ser reorganizada para minimizar cruzamentos entre cabos. Uma ordenação adequada das conexões contribui para reduzir interferências eletromagnéticas, facilitar a montagem e a manutenção dos painéis, melhorar a organização das canaletas e diminuir o comprimento total da fiação, resultando em projetos mais eficientes e confiáveis.

Problemas semelhantes também são encontrados no planejamento e na visualização de redes industriais baseadas em protocolos como PROFINET, *EtherNet*/IP, PROFIBUS e *AS-Interface*. Nesses cenários, dispositivos de campo, controladores e concentradores podem ser representados por grafos, permitindo empregar algoritmos de desenho de grafos e técnicas de minimização de cruzamentos para produzir topologias mais organizadas, facilitar o roteamento físico dos cabos, reduzir custos de instalação e simplificar atividades de manutenção e diagnóstico da rede (FREY *et al.*, 2017).

Outra aplicação relevante encontra-se na geração automática de diagramas de lógica de controle e Interfaces Homem-Máquina (IHM), incluindo *Function Block Diagram* (FBD), Redes de Petri, *Grafcet* e sistemas supervisórios SCADA. Nesses ambientes, técnicas de desenho hierárquico de grafos e minimização de cruzamentos contribuem para produzir diagramas mais legíveis, reduzindo ambiguidades visuais e facilitando as atividades de operação, manutenção e diagnóstico de sistemas automatizados (SPÖNEMANN; SCHULZE; HANXLEDEN, 2014).

Esse amplo conjunto de aplicações evidencia que o OSCM permanece como um problema de elevada relevância científica e tecnológica. Seu impacto estende-se desde aplicações clássicas, como o projeto de circuitos eletrônicos e o desenho hierárquico de grafos, até aplicações contemporâneas relacionadas à engenharia de software, automação industrial, redes de comunicação e visualização de sistemas complexos. Esse cenário justifica o desenvolvimento contínuo de heurísticas e metaheurísticas capazes de produzir soluções de alta qualidade para instâncias cada vez maiores.

3.5 Principais Métodos para o OSCM

Desde a proposição formal do *One-Sided Crossing Minimization*, diversos métodos têm sido desenvolvidos com o objetivo de reduzir o número de cruzamentos em grafos bipartidos de duas camadas. Em função da natureza NP-difícil do problema, a literatura apresenta abordagens que variam desde algoritmos exatos até heurísticas e metaheurísticas capazes de produzir soluções de alta qualidade em tempo computacional reduzido.

Os métodos exatos buscam encontrar a solução ótima do problema por meio de formulações matemáticas e técnicas de exploração sistemática do espaço de busca, incluindo Programação Inteira (PI), algoritmos de *Branch-and-Bound* e métodos baseados em complexidade parametrizada. Embora esses métodos sejam capazes de garantir a otimalidade da solução, seu uso torna-se limitado para instâncias de grande porte devido ao elevado custo computacional associado (KINDERMANN; KLUTE; TERZIADIS, 2024).

Como alternativa aos métodos exatos, diversas heurísticas construtivas foram propostas para gerar rapidamente soluções viáveis. Entre as mais conhecidas destacam-se as heurísticas do Baricentro (*Barycenter*) e da Mediana (*Median*), amplamente empregadas na etapa de minimização de cruzamentos em algoritmos de desenho hierárquico de grafos desde o trabalho pioneiro de (SUGIYAMA; TAGAWA; TODA, 1981). Essas abordagens ordenam os vértices da camada livre com base nas posições de seus vértices adjacentes na camada fixa, apresentando uma boa relação entre qualidade da solução e tempo de execução (EADES; SUGIYAMA, 1990).

Apesar de eficientes, as soluções produzidas por heurísticas construtivas nem sempre correspondem a ótimos locais. Por esse motivo, métodos de busca local são frequente-

mente utilizados como mecanismos de refinamento. Essas técnicas exploram a vizinhança de uma solução por meio de operações de reordenação dos vértices, buscando reduzir progressivamente o número de cruzamentos. Entre os movimentos mais empregados na literatura destacam-se *Swap*, *Insert* e *2-Opt*, os quais diferem na forma como modificam a permutação dos vértices da camada livre.

Além das buscas locais tradicionais, diversas metaheurísticas têm sido investigadas para o OSCM com o objetivo de escapar de ótimos locais e intensificar a exploração do espaço de busca. Entre elas destacam-se o *Simulated Annealing*, a Busca Tabu, os Algoritmos Genéticos (AG), o *Variable Neighborhood Search* (VNS) e algoritmos meméticos, que têm obtido resultados competitivos para o problema (SEGURA *et al.*, 2024). Essas abordagens compartilham mecanismos de diversificação e intensificação da busca, características fundamentais das metaheurísticas (BLUM; ROLI, 2003). As metaheurísticas tem apresentado resultados particularmente promissores para instâncias de grande porte, como evidenciado pelos trabalhos desenvolvidos no âmbito do PACE *Challenge* 2024 (KINDERMANN; KLUTE; TERZIADIS, 2024).

3.6 O Desafio PACE *Challenge* 2024

Apesar de o Problema de Minimização de Cruzamentos Unilaterais ser estudado há várias décadas, os avanços recentes em algoritmos exatos, heurísticos e parametrizados renovaram o interesse da comunidade científica pelo problema. Nesse contexto, um marco importante foi a escolha do OSCM como tema central do *Parameterized Algorithms and Computational Experiments Challenge* 2024 , uma competição internacional voltada à avaliação experimental de algoritmos para problemas computacionalmente difíceis (KINDERMANN; KLUTE; TERZIADIS, 2024).

O PACE *Challenge* é uma iniciativa científica que busca estimular o desenvolvimento de algoritmos práticos por meio da comparação sistemática de diferentes abordagens sob critérios padronizados de desempenho, escalabilidade e qualidade de solução. Em cada edição, um problema de pesquisa relevante é selecionado e um conjunto de instâncias de teste é disponibilizado à comunidade, juntamente com regras de avaliação e uma infraestrutura computacional comum, permitindo comparações justas e reprodutíveis entre os métodos propostos.

Na edição de 2024, o OSCM foi escolhido em razão de sua relevância teórica, de suas diversas aplicações práticas e, sobretudo, de seu caráter desafiador do ponto de vista algorítmico, uma vez que combina NP-dificuldade, rica estrutura combinatória e forte impacto em aplicações de visualização de grafos em camadas (MUÑOZ; UNGER; VRT'O, 2001; DI BATTISTA *et al.*, 1999).

A competição foi organizada em três trilhas principais:

- Trilha Exata (*Exact Track*): destinada ao desenvolvimento de algoritmos capazes de encontrar soluções ótimas comprovadas, priorizando a qualidade da solução em detrimento do tempo de execução;
- Trilha Heurística (*Heuristic Track*): voltada à obtenção de soluções de alta qualidade dentro de limites estritos de tempo, explorando heurísticas, metaheurísticas e métodos híbridos;
- Trilha Parametrizada (*Parameterized Track*): dedicada ao desenvolvimento de algoritmos baseados em complexidade parametrizada, explorando parâmetros estruturais do problema e técnicas de algoritmos FPT (KINDERMANN; KLUTE; TERZIADIS, 2024).

Embora as três trilhas considerassem a mesma formulação do OSCM, uma camada fixa, uma camada livre e o objetivo de minimizar o número total de cruzamentos de arestas, as abordagens algorítmicas diferiram substancialmente. Entre os métodos empregados destacam-se modelos de PI, algoritmos de *Branch-and-Bound*, técnicas de separação e corte (*Branch-and-Cut*), heurísticas construtivas, procedimentos de busca local e métodos híbridos que combinam diferentes estratégias de otimização.

Um dos principais legados do PACE 2024 foi a disponibilização pública de um conjunto amplo e diversificado de instâncias de teste, abrangendo desde grafos pequenos e estruturados até instâncias de grande porte e alta densidade. Além das instâncias, foram disponibilizadas soluções de referência e implementações base, estabelecendo um importante ambiente experimental para pesquisas futuras sobre o OSCM (KINDERMANN; KLUTE; TERZIADIS, 2024).

Os resultados da competição evidenciaram que, para muitas instâncias de pequeno e médio porte, algoritmos exatos são capazes de encontrar soluções ótimas dentro dos limites de tempo estabelecidos pela competição. Em contrapartida, instâncias de maior porte e maior complexidade permanecem desafiadoras, tornando necessária a utilização de heurísticas, metaheurísticas e métodos híbridos capazes de produzir soluções de alta qualidade em tempos computacionais compatíveis com aplicações práticas. Esse cenário reforça a importância do desenvolvimento de estratégias eficientes de otimização, capazes de equilibrar qualidade da solução e custo computacional.

Neste trabalho, o conjunto de instâncias disponibilizado pelo PACE *Challenge* 2024 foi utilizado como base experimental para avaliar o desempenho da heurística construtiva da Mediana e da metaheurística *Simulated Annealing*. A utilização desse *benchmark* possibilita comparar os resultados obtidos com soluções de referência amplamente aceitas pela comunidade científica, além de avaliar a capacidade de generalização dos métodos propostos em um conjunto diversificado de instâncias.

4 Metodologia

A metodologia proposta foi desenvolvida com o objetivo de aplicar técnicas heurísticas e metaheurísticas ao Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization Problem* - OSCM). A abordagem adotada combina heurísticas construtivas para geração de soluções iniciais, métodos de busca local voltados ao refinamento intensivo das soluções e metaheurísticas responsáveis por ampliar a exploração do espaço de busca e reduzir a estagnação em ótimos locais.

Os experimentos computacionais foram realizados utilizando instâncias propostas no desafio PACE (*Parameterized Algorithms and Computational Experiments Challenge*) 2024, conforme descrito em *The PACE 2024 Parameterized Algorithms and Computational Experiments Challenge: One-Sided Crossing Minimization* (KINDERMANN; KLUTE; TERZIADIS, 2024). Essas instâncias contemplam grafos bipartidos com diferentes características estruturais, tamanhos e níveis de dificuldade, permitindo avaliar o comportamento dos algoritmos em cenários variados.

Os algoritmos desenvolvidos foram implementados em linguagem C++, utilizando estruturas de dados otimizadas para reduzir o custo computacional das operações de avaliação e refinamento das soluções. Os experimentos computacionais foram conduzidos em dois ambientes distintos de hardware e software. A principal plataforma de execução foi composta por um processador *Intel Core* i5-10400 de 6 núcleos operando a 2,90 GHz, 16 GB de memória RAM DDR4 (2666 MT/s) e sistema operacional *Ubuntu* 22.04.5 LTS. Adicionalmente, parte dos testes foi realizada em um notebook *Acer Aspire 5* equipado com processador *Intel Core* i5-10210U de 10ª geração (*Quad-Core*), 8 GB de memória RAM DDR4, armazenamento SSD NVMe de 256 GB, tela Full HD de 15,6 polegadas (1920×1080) e sistema operacional Windows 11 *Home Single Language*.

Tabela 2 – Especificações dos ambientes computacionais utilizado nos experimentos

Especificação	**Máquina 1**	**Máquina 2**
Processador	*Intel Core* i5-10400 (2,90 GHz)	*Intel Core* i5-10210U (1,60 GHz)
Núcleos	6	4
Memória RAM	16 GB DDR4 (2666 MT/s)	8 GB DDR4
Sistema Operacional	*Ubuntu* 22.04.5 LTS	Windows 11 *Home Single Language*

4.1 Representação Computacional da Solução

No Problema de Minimização de Cruzamentos Unilaterais, a ordem dos vértices pertencentes à camada fixa permanece inalterada ao longo de toda a execução do algoritmo. Assim, uma solução é completamente determinada pela ordenação dos vértices pertencentes à camada livre.

Considere um grafo bipartido G, conforme definido pelas Equações 1, 2 e 3, em que X corresponde à camada fixa, Y à camada livre e $E \subseteq X \times Y$ representa o conjunto de arestas que conectam vértices pertencentes a partições distintas. Uma solução do problema é representada por uma permutação dos vértices de Y, dada por

$$\pi = [y_1, y_2, \dots, y_{|Y|}] \tag{6}$$

em que cada vértice ocupa uma posição específica na ordenação da camada livre.

Como consequência, o espaço de busca do problema é composto por todas as possíveis permutações dos vértices de Y, totalizando $|Y|!$ soluções distintas.

Essa representação é particularmente adequada ao OSCM, uma vez que todos os movimentos realizados pelas heurísticas e metaheurísticas consistem em modificações na ordem relativa dos vértices da camada livre, mantendo-se inalterada a disposição dos vértices da camada fixa.

Computacionalmente, a solução é armazenada por meio de um vetor de inteiros, denominado *permutation*, em que cada posição representa um vértice da camada livre em determinada posição da ordenação. Além disso, é mantido um vetor auxiliar de posições, denominado *position*, responsável por armazenar a posição corrente de cada vértice na permutação. Essa estrutura permite consultas e atualizações em tempo constante, reduzindo o custo computacional das operações realizadas durante as buscas locais e a metaheurística.

4.1.1 Avaliação da Função Objetivo

Durante a execução das heurísticas construtivas, das buscas locais e da metaheurística *Simulated Annealing*, a função objetivo é avaliada repetidamente para determinar o número de cruzamentos correspondente à ordenação corrente da camada livre.

Embora a função objetivo tenha sido formalmente definida na Seção 3.2, sua avaliação direta após cada modificação da solução apresenta elevado custo computacional, pois exige a recontagem completa dos cruzamentos. Como esse procedimento é realizado milhares de vezes durante a exploração do espaço de busca, seu custo torna-se um fator limitante para o desempenho dos algoritmos.

Para contornar essa limitação, adotou-se uma estratégia de avaliação incremental. Em vez de recalcular integralmente a função objetivo após cada movimento, calcula-se apenas a variação (Δ) provocada pela alteração da solução corrente. Assim, o novo valor da função objetivo é obtido a partir do valor da solução anterior acrescido da respectiva variação, reduzindo significativamente o esforço computacional necessário para avaliar cada movimento. A implementação dessa estratégia baseia-se em uma matriz de cruzamentos pré-computada, apresentada na Seção seguinte, que permite calcular eficientemente a variação da função objetivo para as diferentes estruturas de vizinhança utilizadas neste trabalho.

4.1.2 Matriz de Cruzamentos

A estratégia de avaliação incremental adotada neste trabalho baseia-se na construção prévia de uma matriz de cruzamentos:

$$C = (c_{ij}) \in \mathbb{N}^{|Y|\times|Y|}, \tag{7}$$

na qual cada elemento c_{ij} representa o número de cruzamentos produzidos quando o vértice i precede o vértice j na ordenação da camada livre (BOEHMER *et al.*, 2024).

Essa matriz armazena previamente a contribuição de cada par ordenado de vértices para a função objetivo. Assim, após uma modificação na permutação, não é necessário recalcular o número total de cruzamentos do grafo. Em vez disso, a variação da função objetivo (Δ) é obtida diretamente a partir dos elementos da matriz afetados pelo movimento realizado.

Essa estrutura é empregada em todas as buscas locais implementadas neste trabalho. Nas vizinhanças *Swap*, *Insert* e *2-Opt*, os valores da matriz C são utilizados para calcular incrementalmente a variação da função objetivo, permitindo avaliar rapidamente os movimentos candidatos e reduzindo significativamente o custo computacional da exploração das vizinhanças.

Como contrapartida, a matriz de cruzamentos apresenta complexidade espacial $O(|Y|^2)$, exigindo armazenamento quadrático em relação ao número de vértices da camada livre. Para instâncias de grande porte, esse requisito pode resultar em elevado consumo de memória. Nesses casos, estruturas como árvores de Fenwick (*Fenwick Trees*), árvores balanceadas ou estratégias de cálculo sob demanda podem ser empregadas para reduzir o consumo de memória, à custa de um maior tempo de avaliação dos movimentos.

4.2 Visão Geral dos Algoritmos Propostos

O presente trabalho concentra-se na avaliação de heurísticas construtivas, métodos de busca local e da metaheurística *Simulated Annealing*, buscando analisar a contribuição de cada componente para a obtenção de soluções de alta qualidade para o OSCM. A Tabela 3 apresenta uma comparação entre as principais classes de métodos empregadas na resolução do OSCM, destacando suas características, vantagens e limitações.

Tabela 3 – Comparação entre os principais métodos utilizados para o OSCM

Método	**Exemplos**	**Vantagens**	**Limitações**
Heurísticas construtivas	*Barycenter*, *Median*	Baixo tempo de execução e boa qualidade das soluções iniciais.	Não garantem otimalidade e podem produzir soluções distantes do ótimo.
Busca local	*Swap*, *Insert*, *2-Opt*	Refinam soluções iniciais e alcançam ótimos locais rapidamente.	Dependem da solução inicial e podem ficar presas em ótimos locais.
Metaheurísticas	*Simulated Annealing*	Maior capacidade de escapar de ótimos locais e produzir soluções de alta qualidade.	Maior tempo computacional e necessidade de ajuste de parâmetros.

4.3 Construção da Solução Inicial

A qualidade da solução inicial exerce influência direta no desempenho das heurísticas de refinamento e das metaheurísticas aplicadas ao *One-Sided Crossing Minimization*, uma vez que o ponto de partida da busca pode influenciar tanto a velocidade de convergência quanto a qualidade das soluções finais obtidas (TALBI, 2009). Dessa forma, a construção de uma solução inicial eficiente constitui uma etapa importante no desenvolvimento de métodos aproximativos para o OSCM.

Com esse propósito, foram implementadas duas heurísticas construtivas clássicas amplamente utilizadas na literatura de minimização de cruzamentos em grafos bipartidos: a heurística do Baricentro (*Barycenter*) e a heurística da Mediana (*Median*) (SUGIYAMA; TAGAWA; TODA, 1981). Ambas determinam uma ordenação inicial para os vértices da camada livre a partir das posições de seus vértices adjacentes na camada fixa, diferindo apenas no critério estatístico empregado para calcular essa ordenação.

Essas heurísticas foram selecionadas por apresentarem baixo custo computacional e produzirem soluções iniciais de boa qualidade, características que as tornam amplamente empregadas como ponto de partida para algoritmos de busca local e metaheurísticas voltadas à minimização de cruzamentos (EADES; SUGIYAMA, 1990).

As soluções geradas pelas heurísticas construtivas são utilizadas neste trabalho como ponto de partida para os procedimentos de busca local e para a metaheurística *Simulated Annealing*, descritos nas seções seguintes, permitindo avaliar a influência da solução inicial na qualidade dos resultados obtidos.

4.3.1 Heurística do Baricentro

A heurística do Baricentro (*Barycenter*) atribui a cada vértice da camada livre um valor correspondente à média das posições ocupadas por seus vértices adjacentes na camada fixa (JÜNGER; MUTZEL, 2002). A partir desses valores, os vértices são ordenados em ordem crescente, buscando posicionar vértices adjacentes próximos entre si e, consequentemente, reduzir a ocorrência de cruzamentos entre as arestas.

Em sua forma clássica, a heurística é determinística, produzindo sempre a mesma solução para uma determinada instância. Neste trabalho, entretanto, foi empregada uma adaptação parcialmente gulosa inspirada no procedimento construtivo do *Greedy Randomized Adaptive Search Procedure* (GRASP) (FEO; RESENDE, 1995). Nessa adaptação, em vez de selecionar a cada etapa apenas o vértice de menor valor de baricentro, a escolha é realizada a partir de uma Lista Restrita de Candidatos (*Restricted Candidate List* - RCL), composta pelos vértices considerados mais promissores.

A cada iteração, a RCL é formada pelos vértices cujos valores de baricentro pertencem a um intervalo definido pelo parâmetro $\alpha \in [0, 1]$. O limiar utilizado para construir essa lista é dado por:

$$\tau = b_{\min} + \alpha(b_{\max} - b_{\min}), \tag{8}$$

em que:

- $b_{\min}$ representa o menor valor de baricentro disponível;
- $b_{\max}$ representa o maior valor de baricentro disponível;
- $\alpha \in [0, 1]$ controla o grau de aleatoriedade durante a construção da solução.

Após a construção da RCL, um dos vértices pertencentes à lista é selecionado aleatoriamente e inserido na solução parcial. O parâmetro α estabelece o equilíbrio entre intensificação e diversificação do processo construtivo. Quando $\alpha = 0$, apenas os vértices com menor valor de baricentro são considerados, tornando o procedimento puramente guloso. Por outro lado, quando $\alpha = 1$, todos os vértices remanescentes tornam-se elegíveis, maximizando a diversificação durante a construção da solução.

Essa estratégia combina a eficiência da abordagem gulosa com a capacidade de exploração proporcionada pela aleatoriedade controlada, permitindo gerar diferentes soluções iniciais para uma mesma instância. As soluções construídas pela heurística do Baricentro são posteriormente utilizadas como ponto de partida para os procedimentos de busca local e para a metaheurística *Simulated Annealing*, descritos nas seções seguintes.

O Algoritmo 1 apresenta o procedimento utilizado para a construção da solução inicial baseada na heurística do Baricentro parcialmente gulosa. Inicialmente são calculados os valores de baricentro de todos os vértices da camada livre. Em seguida, a solução é construída iterativamente por meio da formação da Lista Restrita de Candidatos, da seleção aleatória de um vértice pertencente a essa lista e de sua inserção ao final da permutação até que todos os vértices sejam ordenados.

Algoritmo 1: Heurística do Baricentro Parcialmente Gulosa

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, e parâmetro $\alpha \in [0, 1]$.

Output: Permutação π dos vértices de Y.

$\pi \leftarrow \emptyset$
$C \leftarrow Y$
foreach $y \in Y$ **do**
    $BC(y) \leftarrow$ média das posições dos vizinhos de y em X
end
while $C \neq \emptyset$ **do**
    $b_{\min} \leftarrow \min\{BC(y) \mid y \in C\}$
    $b_{\max} \leftarrow \max\{BC(y) \mid y \in C\}$
    $\tau \leftarrow b_{\min} + \alpha(b_{\max} - b_{\min})$
    $RCL \leftarrow \{y \in C \mid BC(y) \leq \tau\}$
    Selecionar aleatoriamente $y \in RCL$
    Inserir y ao final de π
    Remover y de C
end
return π

A Figura 8 apresenta um exemplo da execução da heurística do Baricentro utilizando a estratégia parcialmente gulosa proposta neste trabalho.

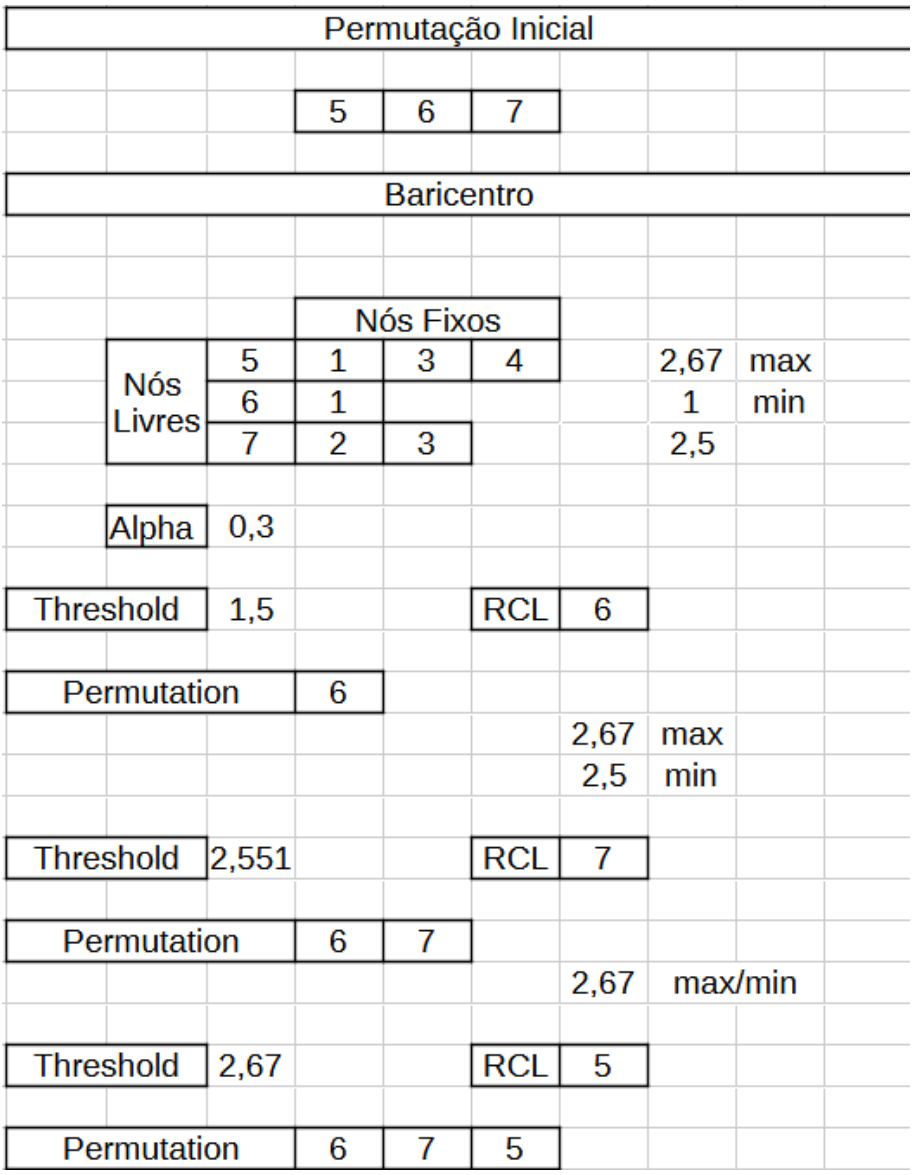

Figura 8 – Exemplo de execução da heurística do Baricentro

Fonte: Elaborado pelo autor (2026)

Inicialmente, para cada vértice da camada livre são calculados os respectivos valores de Baricentro a partir da média das posições de seus vértices adjacentes na camada fixa. No exemplo, os vértices 5, 6 e 7 recebem os valores 2,67, 1,00 e 2,50, respectivamente.

Considerando o parâmetro $\alpha = 0,3$, calcula-se o limiar Lista Restrita de Candidatos, obtendo-se $\tau = 1,5$. Dessa forma, apenas o vértice 6 satisfaz o critério de inclusão na RCL, sendo selecionado para compor a primeira posição da permutação.

Na segunda iteração, os valores mínimo e máximo são atualizados considerando apenas os vértices remanescentes, resultando em um novo limiar igual a 2,551. Nessa etapa,

somente o vértice 7 pertence à RCL e é inserido na segunda posição da solução. Por fim, resta apenas o vértice 5, que é automaticamente alocado na última posição, produzindo a permutação final (6, 7, 5).

O exemplo apresentado evidencia o funcionamento da adaptação proposta para a heurística do Baricentro, ilustrando como o parâmetro α controla a composição da Lista Restrita de Candidatos e, consequentemente, o grau de aleatoriedade empregado na construção da solução. Dessa forma, diferentes execuções podem produzir ordenações iniciais distintas para uma mesma instância, ampliando a diversidade das soluções utilizadas nas etapas posteriores de refinamento.

4.3.2 Heurística da Mediana

A heurística da Mediana (*Median*) atribui a cada vértice da camada livre um valor correspondente à mediana das posições ocupadas por seus vértices adjacentes na camada fixa (GUTOWSKI *et al.*, 2025). Diferentemente da heurística do Baricentro, essa abordagem apresenta maior robustez em relação à presença de vizinhos posicionados em regiões extremas da camada fixa, uma vez que a Mediana é menos sensível a valores discrepantes do que a média aritmética. Dessa forma, tende a produzir ordenações iniciais mais estáveis quando a distribuição dos vizinhos apresenta elevada dispersão ou assimetria.

Formalmente, seja

$$P(y) = [p_1, p_2, \dots, p_k] \tag{9}$$

a sequência ordenada das posições dos vértices adjacentes ao vértice y na camada fixa. O valor associado a y corresponde à Mediana dessa sequência. Quando o número de elementos é ímpar, utiliza-se o elemento central da sequência. Caso contrário, considera-se a média dos dois elementos centrais. Após o cálculo das medianas de todos os vértices da camada livre, estes são ordenados em ordem crescente de seus respectivos valores, produzindo uma permutação inicial para o problema.

O Algoritmo 2 apresenta o procedimento utilizado para a construção da solução inicial baseada na heurística da Mediana. Inicialmente, são obtidas as posições dos vértices adjacentes na camada fixa para cada vértice da camada livre. Em seguida, calcula-se a mediana dessas posições e armazena-se o par formado pelo valor da Mediana e pelo vértice correspondente. Ao final do processo, os vértices são ordenados de acordo com os valores calculados, produzindo a permutação inicial.

Algoritmo 2: Heurística da Mediana

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$

Output: Permutação π

$M \leftarrow$ lista vazia
foreach $y \in Y$ **do**
    $P \leftarrow$ lista ordenada das posições dos vizinhos de y em X
    $k \leftarrow |P|$
    if $k = 0$ **then**
        $m(y) \leftarrow$ posição original de y
    else if k *é ímpar* **then**
        $m(y) \leftarrow P[\lceil k/2 \rceil]$
    else
        $m(y) \leftarrow \dfrac{P[k/2] + P[k/2+1]}{2}$
    end
    Inserir $(m(y), y)$ em M
end
Ordenar M
$\pi \leftarrow$ sequência extraída de M
return π

A Figura 9 apresenta um exemplo da execução da heurística da Mediana.

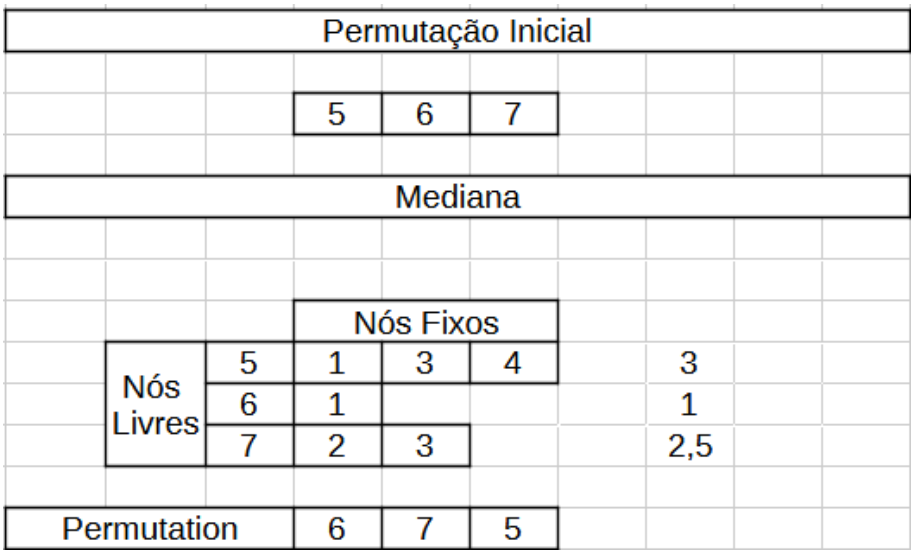

Figura 9 – Exemplo da execução da heurística da Mediana

Fonte: Elaborado pelo autor (2026)

Inicialmente, são identificadas as posições dos vértices adjacentes a cada vértice da camada livre e calcula-se a mediana correspondente. Esses valores passam a representar a prioridade de cada vértice durante a construção da solução. Em seguida, os vértices são ordenados em ordem não decrescente de suas medianas, produzindo diretamente a permutação inicial da camada livre.

O exemplo ilustra que a ordenação final depende exclusivamente da posição mediana dos vizinhos de cada vértice, sem a introdução de mecanismos de aleatoriedade. Diferentemente da adaptação proposta para a heurística do Baricentro, a heurística da Mediana é totalmente determinística, produzindo sempre a mesma solução para uma determinada instância.

Nos experimentos computacionais foram avaliadas tanto as soluções produzidas pela heurística da Mediana quanto pela versão parcialmente gulosa da heurística do Baricentro, com o objetivo de analisar a influência da qualidade e da diversidade das soluções iniciais sobre o desempenho das buscas locais e da metaheurística *Simulated Annealing*, apresentadas nas seções seguintes.

4.4 Refinamento por Busca Local

Após a construção da solução inicial, foi aplicada uma etapa de refinamento baseada em heurísticas de busca local. Essas abordagens operam iterativamente sobre uma solução corrente, buscando reduzir progressivamente o número de cruzamentos por meio da aplicação de movimentos locais (BLUM; ROLI, 2003).

No contexto do Problema de Minimização de Cruzamentos Unilaterais (OSCM), os movimentos correspondem a modificações na ordenação dos vértices pertencentes à camada livre do grafo bipartido. Cada modificação aplicada gera uma nova solução pertencente à vizinhança da solução atual.

Formalmente, dada uma permutação π, define-se sua vizinhança $N(\pi)$ como o conjunto de permutações que podem ser obtidas a partir de π mediante a aplicação de um único movimento elementar. O processo iterativo de busca consiste em explorar sucessivamente essas vizinhanças, substituindo a solução corrente por uma solução vizinha de melhor qualidade sempre que uma melhoria é encontrada. Uma solução é considerada um ótimo local quando não existe qualquer solução em sua vizinhança capaz de produzir uma redução adicional no número de cruzamentos, isto é,

$$f(\pi) \leq f(\pi'), \qquad \forall\, \pi' \in N(\pi), \tag{10}$$

em que $f(\pi)$ representa o número de cruzamentos da solução π, e π' representa uma solução pertencente à vizinhança $N(\pi)$.

Neste trabalho, as buscas locais foram implementadas utilizando duas estratégias de exploração da vizinhança: *Best Improvement* e *First Improvement*. Na estratégia *Best Improvement* (Melhor Melhora), todos os movimentos candidatos pertencentes à vizinhança são avaliados exaustivamente, sendo aplicado apenas o movimento que produz a maior redução na função objetivo. Embora apresente maior custo computacional por iteração, essa abordagem tende a produzir refinamentos mais intensivos e soluções locais de melhor qualidade.

Por outro lado, na estratégia *First Improvement* (Primeira Melhora), os movimentos são avaliados sequencialmente, e a busca é interrompida assim que a primeira melhoria é encontrada. Essa abordagem reduz o custo computacional por iteração e pode acelerar o processo de refinamento em instâncias de maior porte. O procedimento iterativo é encerrado quando nenhuma solução vizinha capaz de melhorar a função objetivo é encontrada, caracterizando a convergência para um ótimo local em relação à estrutura de vizinhança utilizada.

4.4.1 Estruturas de Vizinhança

Para investigar o comportamento das buscas locais em diferentes regiões do espaço de busca, foram consideradas três estruturas de vizinhança distintas: *Swap*, *Insert* e *2-Opt*. Cada uma delas define um conjunto distinto de movimentos capazes de gerar novas soluções a partir da solução corrente.

- *Swap:* consiste na troca de posição entre dois vértices da permutação.

- *Insert:* consiste na remoção de um vértice de sua posição atual e sua posterior reinserção em uma nova posição da ordenação.
- *2-Opt:* consiste na inversão completa da ordem dos vértices pertencentes a um subsegmento da solução.

Cada estrutura apresenta características distintas de exploração do espaço de busca. O movimento *Swap* realiza alterações mais localizadas, promovendo pequenas mudanças na ordenação dos vértices. Por outro lado, os movimentos *Insert* e *2-Opt* permitem modificações mais expressivas na solução, possibilitando a exploração de regiões mais distantes do espaço de busca e, consequentemente, aumentando a capacidade de escapar de ótimos locais.

Nas subseções seguintes, são apresentados em detalhes o funcionamento de cada estrutura de vizinhança, bem como os procedimentos adotados para a avaliação incremental da variação do número de cruzamentos após a realização de cada movimento.

4.4.2 Busca Local *Swap*

A vizinhança *Swap* consiste na troca de posição entre dois vértices x e y pertencentes à camada livre, localizados nas posições i e j da permutação corrente. Para essa estrutura de vizinhança foram implementadas as estratégias *Best Improvement* e *First Improvement*. Na estratégia *Best Improvement*, todos os pares de posições (i, j), com $i < j$, são avaliados exaustivamente, sendo aplicado o movimento que produz a maior redução no número de cruzamentos. Já na estratégia *First Improvement*, a busca é interrompida assim que uma troca capaz de melhorar a solução corrente é encontrada, reduzindo o custo computacional por iteração.

A avaliação dos movimentos é realizada de forma incremental por meio da matriz de cruzamentos pré-computada. O cálculo da variação da função objetivo (Δ) considera a inversão da ordem relativa entre os vértices trocados e os efeitos provocados sobre os vértices intermediários compreendidos entre as posições i e j. A nova quantidade de cruzamentos após a aplicação do movimento é dada por

$$f(\pi') = f(\pi) + \Delta \tag{11}$$

Como a vizinhança *Swap* possui

$$\frac{|Y|(|Y|-1)}{2}$$

movimentos possíveis, sua exploração completa apresenta complexidade $O(|Y|^2)$. O processo iterativo é encerrado quando nenhuma troca produz melhoria adicional na solução ou quando o limite de tempo estabelecido é atingido.

Os procedimentos de busca local baseados na vizinhança Swap são apresentados nos Algoritmos 3 e 4. Em ambos os casos, são avaliadas trocas entre pares de vértices consecutivos da partição livre, sendo a variação no número de cruzamentos calculada de forma incremental por meio da matriz de cruzamentos. No Algoritmo 3, correspondente à estratégia *Best Improvement*, toda a vizinhança é percorrida para identificar o movimento que proporciona a maior redução no número de cruzamentos, o qual é aplicado somente após a conclusão da avaliação de todos os candidatos. Em contraste, o Algoritmo 4, correspondente à estratégia *First Improvement*, interrompe a exploração da vizinhança assim que encontra o primeiro movimento capaz de reduzir o número de cruzamentos, aplicando imediatamente a troca e reiniciando o processo de busca.

Algoritmo 3: Busca Local *Swap - Best Improvement*

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo L

Output: Permutação refinada π

```
1  improved ← true
2  while improved do
3      improved ← false
4      best_delta ← 0
5      best_i ← −1
6      best_j ← −1
7      for i ← 0 to |Y| − 2 do
8          for j ← i + 1 to |Y| − 1 do
9              Δ ← calculateSwapDelta(G, π, i, j)
10             if Δ < best_delta then
11                 best_delta ← Δ
12                 best_i ← i
13                 best_j ← j
14             end
15         end
16         if TempoDecorrido() > L then
17             return π
18         end
19     end
20     if best_delta < 0 then
21         Trocar π[best_i] e π[best_j]
22         improved ← true
23     end
24 end
25 return π
```

Algoritmo 4: Busca Local *Swap - First Improvement*

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo L

Output: Permutação refinada π

```
1  improved ← true
2  while improved do
3      improved ← false
4      for i ← 0 to |Y| − 2 do
5          for j ← i + 1 to |Y| − 1 do
6              Δ ← calculateSwapDelta(G, π, i, j)
7              if Δ < 0 then
8                  Trocar π[i] e π[j]
9                  improved ← true
10                 break
11             end
12         end
13         if improved then
14             break
15         end
16         if TempoDecorrido() > L then
17             return π
18         end
19     end
20 end
21 return π
```

A Figura 10 ilustra a aplicação de um movimento da vizinhança *Swap*.

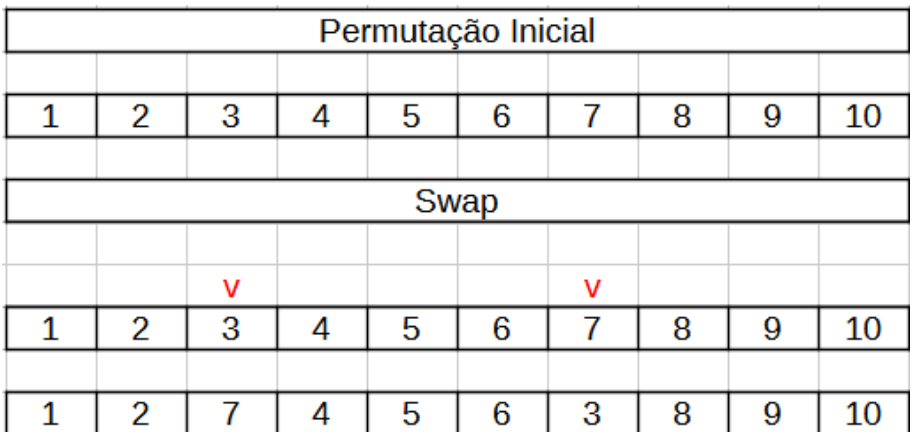

Figura 10 – Aplicação de um movimento da vizinhança *Swap*

Fonte: Elaborado pelo autor (2026)

No exemplo apresentado, a solução inicial é representada pela permutação sequencial de 1 a 10. O movimento consiste na troca das posições dos vértices 3 e 7, destacados em vermelho, resultando na nova permutação (1, 2, 7, 4, 5, 6, 3, 8, 9, 10). Observa-se que apenas os dois vértices selecionados têm suas posições alteradas, enquanto os demais permanecem inalterados.

Essa modificação altera a ordem relativa das arestas incidentes aos vértices envolvidos, podendo reduzir ou aumentar o número total de cruzamentos do desenho bipartido. Após a realização da troca, a variação da função objetivo é calculada de forma incremental por meio da matriz de cruzamentos. Caso o movimento resulte em uma redução no número de cruzamentos, ele poderá ser incorporado à solução corrente conforme a estratégia de busca adotada (*Best Improvement* ou *First Improvement*).

4.4.3 Busca Local *Insert*

A vizinhança *Insert* consiste em remover um vértice de sua posição original i e reinseri-lo em uma nova posição j da permutação. Diferentemente da vizinhança *Swap*, essa estrutura permite deslocamentos mais amplos na solução, possibilitando que um vértice atravesse blocos inteiros da permutação e altere simultaneamente sua relação de precedência com diversos vértices.

As estratégias *Best Improvement* e *First Improvement* também foram empregadas nessa vizinhança. Na primeira, todas as combinações possíveis de remoção e reinserção são avaliadas, sendo aplicado o movimento que produz a maior redução no número de cruzamentos. Na segunda, a busca é interrompida assim que uma reinserção capaz de melhorar a solução corrente é encontrada.

A avaliação dos movimentos é realizada incrementalmente utilizando a matriz de cruzamentos pré-computada, reduzindo significativamente o custo computacional associado ao cálculo da função objetivo. O cálculo da variação da função objetivo depende da direção do deslocamento realizado:

- Deslocamento para a direita ($i < j$): o vértice passa a ser posicionado após os vértices pertencentes ao intervalo $[i+1, j]$;
- Deslocamento para a esquerda ($j < i$): o vértice passa a anteceder os vértices pertencentes ao intervalo $[j, i-1]$.

A vizinhança *Insert* possui $|Y|(|Y|-1)$ movimentos possíveis, resultando em complexidade $O(|Y|^2)$ para sua exploração completa. O procedimento iterativo é encerrado quando nenhuma reinserção produz melhoria adicional na solução ou quando o limite de tempo estipulado é atingido.

Os procedimentos de refinamento baseados na vizinhança *Insert* são descritos nos Algoritmos 5 e 6. Nessa vizinhança, cada movimento consiste na remoção de um vértice de sua posição atual e sua reinserção em outra posição da permutação, sendo a variação no número de cruzamentos obtida de forma incremental por meio da matriz de cruzamentos. No Algoritmo 5, todos os movimentos possíveis são avaliados, e apenas aquele que produz a maior redução no número de cruzamentos é efetivamente executado ao final da iteração. Já o Algoritmo 6 percorre a vizinhança até encontrar o primeiro movimento que melhora a solução corrente, realizando imediatamente a inserção correspondente e reiniciando a exploração da vizinhança.

Algoritmo 5: Busca Local *Insert - Best Improvement*

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo L

Output: Permutação refinada π

```
1  improved ← true
2  while improved do
3      improved ← false
4      best_delta ← 0
5      best_from ← −1, best_to ← −1
6      for i ← 0 to |Y| − 1 do
7          for j ← 0 to |Y| − 1 do
8              if i = j ou j = i + 1 then
9                  continue
10             end
11             Δ ← calculateInsertDelta(G, π, i, j)
12             if Δ < best_delta then
13                 best_delta ← Δ
14                 best_from ← i
15                 best_to ← j
16             end
17         end
18         if TempoDecorrido() > L then
19             return π
20         end
21     end
22     if best_delta < 0 then
23         Remover π[best_from] e inserir na posição best_to
24         improved ← true
25     end
26 end
27 return π
```

Algoritmo 6: Busca Local *Insert - First Improvement*

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo L

Output: Permutação refinada π

```
1  improved ← true
2  while improved do
3      improved ← false
4      for i ← 0 to |Y| − 1 do
5          for j ← 0 to |Y| − 1 do
6              if i = j ou j = i + 1 then
7                  continue
8              end
9              Δ ← calculateInsertDelta(G, π, i, j)
10             if Δ < 0 then
11                 Remover π[i] e inserir na posição j
12                 improved ← true
13                 break
14             end
15         end
16         if improved then
17             break
18         end
19         if TempoDecorrido() > L then
20             return π
21         end
22     end
23 end
24 return π
```

A Figura 11 ilustra a aplicação de um movimento da vizinhança *Insert*.

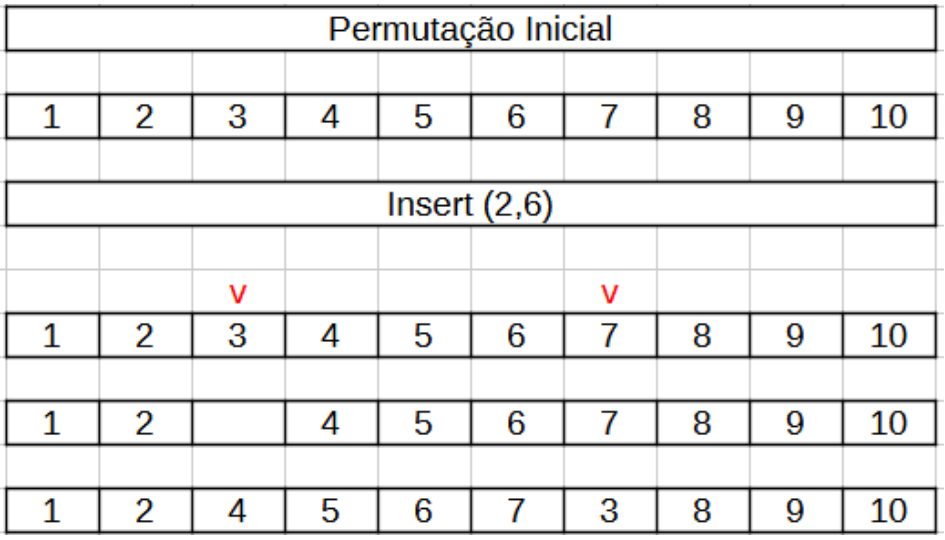

Figura 11 – Aplicação de um movimento da vizinhança *Insert*

Fonte: Elaborado pelo autor (2026)

No exemplo apresentado, a solução inicial é representada pela permutação em sequência de 1 a 10. O movimento *Insert*(2, 6) consiste em remover o vértice 3 de sua posição original e reinseri-lo na posição correspondente ao índice 6 da permutação. Como consequência, os vértices 4, 5, 6 e 7 são deslocados uma posição para a esquerda, resultando na nova permutação (1, 2, 4, 5, 6, 7, 3, 8, 9, 10).

Diferentemente da vizinhança *Swap*, em que apenas dois vértices têm suas posições trocadas, a vizinhança *Insert* modifica a posição de um único vértice, preservando a ordem relativa dos demais elementos da sequência. Essa característica permite deslocamentos mais amplos na solução, possibilitando que um vértice seja movido para qualquer posição da permutação e altere simultaneamente sua relação de precedência com diversos outros vértices.

Após a realização do movimento, a variação da função objetivo é calculada de forma incremental utilizando a matriz de cruzamentos. Caso a reinserção reduza o número total de cruzamentos, o movimento é incorporado à solução corrente conforme a estratégia de busca empregada (*Best Improvement* ou *First Improvement*).

4.4.4 Busca Local *2-Opt*

A vizinhança *2-Opt* consiste na inversão completa da ordem dos vértices pertencentes a um subsegmento da permutação. Dado um intervalo delimitado pelas posições i e j, o movimento inverte a sequência de vértices compreendida nesse intervalo, produzindo uma reorganização mais significativa da solução. Esse operador permite alterar simultaneamente diversas relações de precedência entre vértices, tornando-se particularmente

eficiente para escapar de ótimos locais associados a movimentos mais restritos, como *Swap* e *Insert*.

Assim como nas demais buscas locais, foram implementadas as estratégias *Best Improvement* e *First Improvement*. Na estratégia *Best Improvement*, todas as inversões possíveis são avaliadas, sendo aplicada aquela que produz a maior redução na função objetivo. Já na estratégia *First Improvement*, a busca é interrompida assim que uma inversão capaz de melhorar a solução corrente é encontrada.

A avaliação dos movimentos é realizada incrementalmente utilizando a matriz de cruzamentos pré-computada, reduzindo significativamente o custo de exploração da vizinhança. A vizinhança *2-Opt* possui

$$\frac{|Y|(|Y|-1)}{2}$$

movimentos possíveis e, consequentemente, complexidade $O(Y^2)$ para sua exploração completa. O processo iterativo é encerrado quando nenhuma inversão produz redução adicional no número de cruzamentos ou quando o limite de tempo estabelecido é atingido.

Os Algoritmos 7 e 8 apresentam as implementações das estratégias *Best Improvement* e *First Improvement* para a vizinhança *2-Opt*. Em ambos os procedimentos, a avaliação das inversões é realizada de maneira incremental, utilizando a matriz de cruzamentos pré-computada para reduzir o custo computacional das iterações.

Algoritmo 7: Busca Local *2-Opt - Best Improvement*

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo L

Output: Permutação refinada π

```
1  improved ← true
2  while improved do
3      improved ← false
4      best_delta ← 0
5      best_i ← −1, best_j ← −1
6      for i ← 0 to |Y| − 2 do
7          current_row_delta ← 0
8          for j ← i + 1 to |Y| − 1 do
9              u ← π[j]
               // Calcula a variação da função objetivo utilizando a
                  matriz de cruzamentos
10             for k_idx ← i to j − 1 do
11                 k ← π[k_idx]
12                 current_row_delta ← current_row_delta + (C[u][k] − C[k][u])
13             end
14             if current_row_delta < best_delta then
15                 best_delta ← current_row_delta
16                 best_i ← i
17                 best_j ← j
18             end
19         end
20         if TempoDecorrido() > L then
21             return π
22         end
23     end
24     if best_delta < 0 then
25         Inverter a subsequência de π entre as posições best_i e best_j
26         improved ← true
27     end
28 end
29 return π
```

Algoritmo 8: Busca Local *2-Opt - First Improvement*

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo L

Output: Permutação refinada π

```
1  improved ← true
2  while improved do
3      improved ← false
4      for i ← 0 to |Y| − 2 do
5          current_row_delta ← 0
6          for j ← i + 1 to |Y| − 1 do
7              u ← π[j]
               // Cálculo incremental da variação da função objetivo
                  utilizando a matriz de cruzamentos
8              for k_idx ← i to j − 1 do
9                  k ← π[k_idx]
10                 current_row_delta ← current_row_delta + (C[u][k] − C[k][u])
11             end
12             if current_row_delta < 0 then
13                 Inverter a subsequência de π entre as posições i e j
14                 improved ← true
15                 break
16             end
17         end
18         if improved then
19             break
20         end
21         if TempoDecorrido() > L then
22             return π
23         end
24     end
25 end
26 return π
```

A Figura 12 ilustra a aplicação de um movimento da vizinhança *2-Opt*.

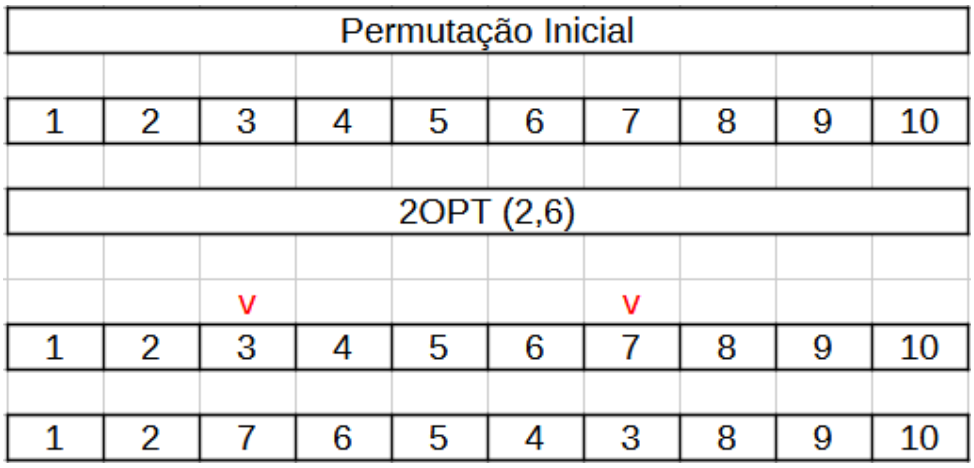

Figura 12 – Aplicação de um movimento da vizinhança *2-Opt*

Fonte: Elaborado pelo autor (2026)

Na permutação inicial, os vértices encontram-se ordenados de forma crescente de 1 a 10. O movimento *2-Opt* (2, 6) seleciona o segmento compreendido entre as posições 2 e 6 da permutação, destacadas pelas setas vermelhas, e realiza a inversão completa da ordem dos vértices pertencentes a esse intervalo. Nesse exemplo, a subsequência (3, 4, 5, 6, 7) é substituída por (7, 6, 5, 4, 3), enquanto os vértices localizados fora do intervalo permanecem inalterados. Como resultado, obtém-se a nova permutação (1, 2, 7, 6, 5, 4, 3, 8, 9, 10).

Esse tipo de movimento promove modificações mais amplas na solução quando comparado às vizinhanças *Swap* e *Insert*, pois altera simultaneamente a ordem relativa de todos os vértices pertencentes ao segmento invertido. Essa característica amplia a capacidade de exploração do espaço de busca, favorecendo a obtenção de soluções de melhor qualidade e reduzindo a probabilidade de aprisionamento em ótimos locais.

4.4.5 *Variable Neighborhood Descent* (VND)

Com o objetivo de ampliar a capacidade de exploração do espaço de busca e reduzir a estagnação em ótimos locais associados a uma única estrutura de vizinhança, foi implementada a heurística *Variable Neighborhood Descent* (VND) (HANSEN *et al.*, 2018).

O VND explora sistematicamente múltiplas estruturas de vizinhança, baseando-se no princípio de que uma solução considerada ótima local em uma determinada vizinhança pode ainda apresentar melhorias quando analisada sob outras estruturas. O funcionamento do algoritmo segue uma estratégia de reinicialização. Inicialmente, a solução corrente é refinada utilizando a primeira vizinhança da sequência. Sempre que uma melhoria é encontrada, a nova solução é aceita e o algoritmo retorna imediatamente à primeira estrutura de vizinhança, reiniciando o processo de refinamento.

Caso nenhuma melhoria seja encontrada, o algoritmo avança para a próxima vizinhança da sequência. Dessa forma, o VND combina diferentes níveis de exploração do espaço de busca, alternando entre movimentos mais locais, como o *Swap*, e movimentos mais estruturais, como o *2-Opt* e o *Insert*. O processo iterativo é encerrado quando nenhuma das estruturas de vizinhança consegue produzir melhorias adicionais na solução ou quando o limite de tempo global é atingido. O procedimento completo do *Variable Neighborhood Descent* implementado neste trabalho é apresentado no Algoritmo 9.

Algoritmo 9: *Variable Neighborhood Descent* (VND)

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, permutação inicial π, limite de tempo total L_{VND}, limite de tempo por vizinhança L_{neigh}

Output: Permutação refinada π

```
1  current_crossings ← countCrossings(π)
2  k ← 1
3  k_max ← 3
4  while k ≤ k_max do
5      π' ← ∅
6      if k = 1 then
7          π' ← localSearchSwapBest(G, π, L_neigh)
8      else if k = 2 then
9          π' ← localSearch2OptBest(G, π, L_neigh)
10     else
11         π' ← localSearchInsertionBest(G, π, L_neigh)
12     end
13     new_crossings ← countCrossings(π')
14     if new_crossings < current_crossings then
15         π ← π'
16         current_crossings ← new_crossings
           // Retorna à primeira vizinhança
17         k ← 1
18     end
19     else
           // Avança para a próxima vizinhança
20         k ← k + 1
21     end
22     if TempoTotal() > L_VND then
23         return π
24     end
25 end
26 return π
```

A Figura 13 ilustra o mecanismo de exploração das múltiplas vizinhanças empregado pelo VND.

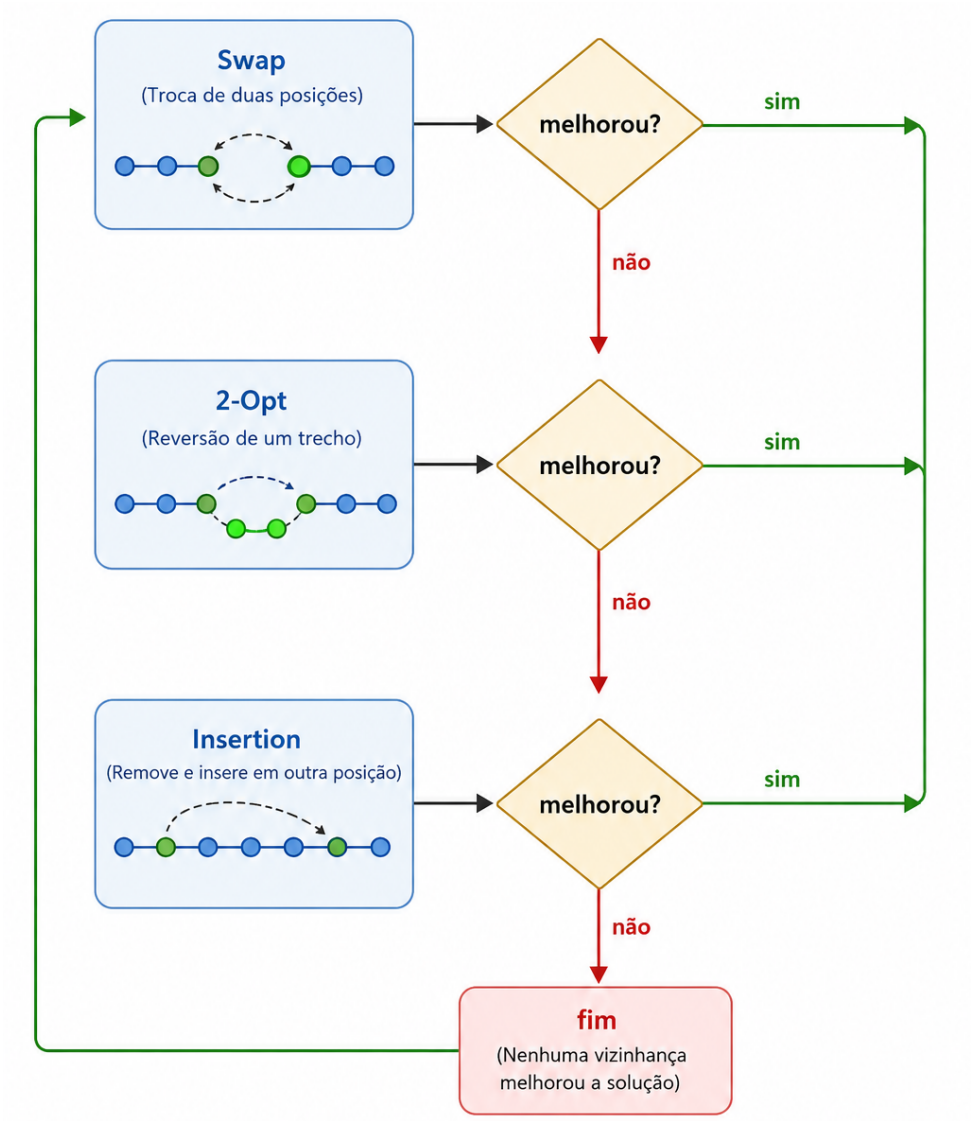

Figura 13 – Exploração de múltiplas vizinhanças pelo VND

Fonte: Elaborado pelo autor (2026)

O processo inicia pela aplicação da vizinhança *Swap*. Caso seja encontrada uma solução com menor número de cruzamentos, essa solução passa a ser a solução corrente e o algoritmo reinicia imediatamente a exploração a partir da própria vizinhança *Swap*, intensificando a busca na região recém-descoberta.

Quando nenhuma melhoria é obtida com a vizinhança *Swap*, o algoritmo passa a explorar a vizinhança *2-Opt*. Novamente, caso uma solução melhor seja encontrada, ela é aceita e o processo retorna à primeira vizinhança da sequência. Na ausência de melhorias, a busca prossegue para a vizinhança *Insert*, responsável por realizar movimentos de reinserção de vértices na permutação. Se a vizinhança *Insert* também não produzir redução no

número de cruzamentos, considera-se que nenhuma das estruturas de vizinhança é capaz de melhorar a solução corrente, encerrando o algoritmo. Esse mecanismo de alternância entre diferentes vizinhanças permite combinar movimentos de diferentes amplitudes e favorece a exploração de regiões distintas do espaço de busca, reduzindo a probabilidade de aprisionamento em ótimos locais.

4.5 Exploração Global do Espaço de Busca

Embora as heurísticas construtivas e os procedimentos de busca local sejam capazes de produzir soluções de boa qualidade para o Problema de Minimização de Cruzamentos Unilaterais (OSCM), esses métodos tendem a concentrar a busca em regiões específicas do espaço de soluções, podendo ficar aprisionados em ótimos locais. Para contornar essa limitação e ampliar a capacidade de exploração global, foi implementada a metaheurística *Simulated Annealing* (SA), proposta por Scott Kirkpatrick, C. Daniel Gelatt e Mario P. Vecchi, em 1983 (KIRKPATRICK; GELATT; VECCHI, 1983).

O SA é inspirado no processo físico de recozimento (*annealing*) utilizado na metalurgia. Nesse processo, um material é inicialmente aquecido a uma elevada temperatura e, em seguida, resfriado de forma lenta e controlada. Durante a etapa de aquecimento, os átomos possuem elevada mobilidade, permitindo que o material explore diferentes configurações estruturais. À medida que a temperatura diminui, essa mobilidade é reduzida gradativamente, conduzindo o sistema a estados de menor energia e maior estabilidade (KIRKPATRICK; GELATT; VECCHI, 1983).

Inspirado nesse fenômeno, o algoritmo associa cada solução do problema a um estado do sistema físico e utiliza um parâmetro denominado *temperatura* para controlar o processo de busca. Em temperaturas elevadas, o algoritmo aceita com maior probabilidade soluções de pior qualidade, favorecendo uma exploração mais ampla do espaço de busca. À medida que a temperatura é reduzida segundo um plano de resfriamento, essa probabilidade diminui, tornando a busca progressivamente mais intensiva nas regiões consideradas promissoras.

A principal característica do *Simulated Annealing* é justamente permitir a aceitação temporária de soluções de pior qualidade durante a execução. Esse mecanismo probabilístico possibilita que o algoritmo escape de ótimos locais e explore regiões do espaço de busca que dificilmente seriam alcançadas por métodos puramente determinísticos. A probabilidade de aceitação dessas soluções é controlada pela temperatura do sistema, diminuindo gradualmente ao longo da execução conforme o processo de resfriamento (KIRKPATRICK; GELATT; VECCHI, 1983).

A solução inicial utilizada pelo algoritmo é obtida por uma das heurísticas construtivas implementadas neste trabalho (Mediana ou Baricentro). A partir dessa solução,

são realizadas perturbações sucessivas por meio de diferentes estruturas de vizinhança, incluindo os movimentos *Swap*, *Insert* e *2-Opt*. As soluções geradas são avaliadas utilizando a matriz de cruzamentos previamente computada, permitindo calcular de forma incremental a variação da função objetivo.

Durante toda a execução são mantidas duas soluções: a solução corrente, que representa o estado atual da busca, e a melhor solução global encontrada até o momento. O processo iterativo prossegue enquanto a temperatura permanecer acima de um valor mínimo previamente definido e o limite de tempo de execução não for atingido. Dessa forma, o algoritmo inicia privilegiando a exploração global do espaço de busca e, à medida que a temperatura diminui, concentra-se progressivamente na intensificação da busca em torno das melhores soluções encontradas.

4.5.1 Critério de Aceitação

A aceitação de movimentos deteriorantes é controlada probabilisticamente de acordo com a temperatura atual do sistema.

A probabilidade de aceitação de uma solução pior é definida por:

$$P = e^{-\Delta/T} \tag{12}$$

em que:

- Δ representa o aumento na função objetivo;
- T representa a temperatura atual do sistema.

Quando uma solução apresenta melhoria ($\Delta < 0$), ela é sempre aceita. Caso contrário, a solução pode ser aceita com probabilidade P, que diminui à medida que a temperatura reduz, tornando o algoritmo progressivamente mais exploratório no início e mais intensivo no final da execução.

4.5.2 Esquema de Resfriamento

A redução da temperatura foi realizada utilizando um esquema geométrico de resfriamento, em que Tk representa a temperatura T na iteração k:

$$T_{k+1} = \alpha_{SA} T_k \tag{13}$$

em que $\alpha_{SA} \in (0, 1)$ representa o fator de resfriamento responsável por controlar a taxa de convergência do algoritmo.

Valores de α_{SA} próximos de 1 promovem um resfriamento mais lento e uma maior exploração do espaço de busca, enquanto valores menores aceleram a convergência, reduzindo o tempo de exploração. O processo é encerrado quando a temperatura atinge um valor mínimo T_{min} ou quando o limite de tempo de execução é alcançado.

4.5.3 Estruturas de Vizinhança

As perturbações aplicadas pelo *Simulated Annealing* utilizam as mesmas estruturas de vizinhança empregadas nas buscas locais: *Swap*, *Insert* e *2-Opt*. A utilização de diferentes estruturas de vizinhança permite combinar movimentos locais e perturbações mais amplas, aumentando a diversidade da exploração do espaço de busca e reduzindo a probabilidade de estagnação em ótimos locais.

4.5.4 Algoritmo Proposto

O Algoritmo 10 inicia a busca a partir de uma solução construída pelas heurísticas do Baricentro ou da Mediana. Em seguida, são realizadas perturbações sucessivas sobre a solução corrente utilizando uma das estruturas de vizinhança implementadas. A cada iteração, um movimento é gerado aleatoriamente e sua variação no número de cruzamentos (Δ) é calculada incrementalmente por meio da matriz de cruzamentos, permitindo avaliar a qualidade da nova solução de forma eficiente. Caso o movimento produza uma melhoria, a nova solução é imediatamente aceita. Caso contrário, sua aceitação é determinada pelo critério probabilístico do *Simulated Annealing*, que permite, com uma probabilidade dependente da temperatura atual do sistema, a aceitação ocasional de soluções de pior qualidade, favorecendo a exploração de diferentes regiões do espaço de busca.

Ao longo da execução, a temperatura é gradativamente reduzida de acordo com a taxa de resfriamento definida, tornando o critério de aceitação progressivamente mais restritivo. Dessa forma, nas etapas iniciais da busca o algoritmo apresenta um comportamento mais exploratório, permitindo escapar de ótimos locais, enquanto nas etapas finais passa a privilegiar movimentos de melhoria, intensificando a busca em regiões promissoras do espaço de soluções. O processo iterativo é repetido até que a temperatura mínima seja atingida ou que o limite de tempo previamente estabelecido seja alcançado.

Algoritmo 10: *Simulated Annealing* aplicado ao OSCM

Input: Grafo bipartido $G = (V, E)$, em que $V = X \cup Y$ e $X \cap Y = \emptyset$, parâmetros $(T_{\max}, T_{\min}, \alpha, L_{iter}, tipo\_viz)$ e limite de tempo L

Output: Melhor permutação π^*

```
1  π ← solução inicial
2  f ← countCrossings(π)
3  π* ← π
4  f* ← f
5  T ← T_max
6  while T > T_min do
7      for i ← 1 to max(|Y|, L_iter) do
8          Selecionar aleatoriamente source, target ∈ {0, ..., |Y| − 1}
9          if source = target then
10             continue
11         end
12         if tipo_viz = 0 then
13             Δ ← calculateSwapDelta(G, π, source, target)
14         else if tipo_viz = 1 then
15             Δ ← calculateInsertDelta(G, π, source, target)
16         else
17             Δ ← calculate2OptDelta(G, π, source, target)
18         end
19         if Δ < 0 or Aleatorio(0, 1) < e^(−Δ/T) then
20             Aplicar movimento em π
21             f ← f + Δ
22             if f < f* then
23                 f* ← f
24                 π* ← π
25             end
26         end
27         if TempoDecorrido() > L then
28             return π*
29         end
30     end
31     T ← αT
32 end
33 return π*
```

4.5.5 Calibração de Parâmetros

O desempenho do *Simulated Annealing* (SA) é fortemente influenciado pela escolha de seus parâmetros, tais como a temperatura inicial, a taxa de resfriamento, a temperatura mínima, o número de iterações realizadas em cada nível de temperatura e a estratégia de geração da solução inicial. Com o objetivo de obter uma configuração robusta e adequada às instâncias do problema estudado, foi empregada a ferramenta *iRace* (*Iterated Racing for Automatic Algorithm Configuration*) (LÓPEZ-IBÁÑEZ *et al.*, 2016) para a calibração automática dos parâmetros do algoritmo.

O *iRace* é um método amplamente utilizado para a configuração automática de algoritmos heurísticos e metaheurísticos, baseado na técnica de *Iterated Racing*. Inicialmente, um conjunto de configurações candidatas é gerado a partir dos intervalos de valores especificados para cada parâmetro. Essas configurações são então avaliadas progressivamente sobre um conjunto de instâncias de treinamento, permitindo comparar seu desempenho de forma sistemática (LÓPEZ-IBÁÑEZ *et al.*, 2016).

Durante o processo de calibração, o *iRace* aplica testes estatísticos paramétricos para identificar configurações significativamente inferiores, descartando-as antecipadamente e concentrando o orçamento computacional nas alternativas mais promissoras. Ao final de cada rodada de avaliações, novas configurações são geradas por meio de amostragem probabilística baseada nas melhores soluções encontradas, refinando progressivamente a exploração do espaço de parâmetros até o esgotamento do orçamento computacional. Como resultado, obtém-se uma configuração de parâmetros capaz de apresentar elevado desempenho médio para o conjunto de instâncias de treinamento (LÓPEZ-IBÁÑEZ *et al.*, 2016).

Os parâmetros considerados durante a calibração e seus respectivos intervalos de valores são apresentados na Tabela 4. Os intervalos de valores foram definidos com base em estudos preliminares e em recomendações da literatura, buscando contemplar diferentes níveis de exploração do espaço de busca sem restringir excessivamente o processo de calibração automática. O parâmetro *init_solution* define a heurística utilizada para construir a solução inicial, sendo o valor 0 correspondente à heurística do Baricentro e o valor 1 à heurística da Mediana. O parâmetro *neighType* determina a estrutura de vizinhança empregada pelo *Simulated Annealing*, assumindo o valor 0 para a vizinhança *Swap* e o valor 1 para a vizinhança *Insert*. Os demais parâmetros controlam o cronograma de resfriamento do algoritmo, incluindo a temperatura inicial (T_max), a temperatura mínima (T_min), o fator de resfriamento (α_{SA}) e o número máximo de iterações realizadas em cada nível de temperatura (L_{iter}).

Tabela 4 – Parâmetros testados no *iRace*

Parâmetro	**Descrição**	**Intervalo testado**
init_solution	Tipo de solução inicial (Baricentro: 0 ou Mediana: 1)	{0,1}
α_{SA}	Fator de resfriamento	{0.85, 0.90, 0.92, 0.95, 0.98, 0.99}
T_max	Temperatura inicial	{100.0, 1000.0, 5000.0, 10000.0, 25000.0, 50000.0}
T_min	Temperatura mínima	{0.00001, 0.0001, 0.001, 0.01, 0.1}
L_{iter}	Número máximo de iterações por temperatura	{1000, 5000, 10000, 25000, 50000}
neighType	Estrutura de vizinhança (*Swap*: 0 ou *Insert*: 1)	{0,1}

Para o processo de calibração foram utilizadas 20 instâncias representativas do *benchmark* do PACE *Challenge* 2024, abrangendo diferentes portes e níveis de complexidade. Esse conjunto inclui instâncias pequenas, médias e grandes, permitindo avaliar o comportamento do algoritmo em cenários com diferentes números de vértices, arestas e níveis de dificuldade. Cada configuração candidata foi avaliada progressivamente sobre esse conjunto de instâncias, sendo comparada às demais com base na qualidade das soluções obtidas dentro do limite de tempo estabelecido. Configurações com desempenho estatisticamente inferior foram descartadas durante o processo de *racing*, permitindo direcionar o esforço computacional para as alternativas mais promissoras.

Ao final do processo de calibração, o *iRace* convergiu para a configuração apresentada na Tabela 5, a qual foi utilizada em todos os experimentos computacionais realizados neste trabalho.

Tabela 5 – Configuração dos parâmetros selecionados pelo *iRace*

Parâmetro	**Valor**
init_solution	1 (Mediana)
α_{SA}	0,95
T_max	10.000
T_min	0.0001
L_{iter}	50.000
neighType	1 (*Insert*)

Observa-se que o processo de calibração selecionou a heurística da Mediana para a geração da solução inicial e a vizinhança *Insert* para a realização dos movimentos do *Simulated Annealing*. Além disso, foram escolhidos um fator de resfriamento igual a 0,95 e um elevado número de iterações por nível de temperatura, indicando que uma exploração mais intensiva do espaço de busca em cada estágio do cronograma de resfriamento produziu melhores resultados para as instâncias avaliadas. Cada execução foi limitada a 300 segundos, em conformidade com o tempo máximo estabelecido pelo PACE *Challenge* 2024 (KINDERMANN; KLUTE; TERZIADIS, 2024). Todos os experimentos foram conduzidos pela nos ambientes computacionais descritos na Tabela 2. A configuração obtida foi adotada em todos os experimentos apresentados neste trabalho, garantindo a utilização de parâmetros calibrados de forma sistemática e reprodutível.

5 Resultados

Neste capítulo, são apresentados e discutidos os resultados obtidos a partir da execução dos algoritmos propostos para o *One-Sided Crossing Minimization*. Os experimentos foram realizados utilizando um conjunto de instâncias de teste com diferentes características estruturais, permitindo avaliar o comportamento das heurísticas construtivas, dos métodos de busca local e da metaheurística *Simulated Annealing*.

Para cada instância, foram analisados o número de cruzamentos produzidos pelas soluções obtidas e o tempo computacional de execução dos algoritmos, medido em segundos. As heurísticas construtivas Baricentro e Mediana apresentaram tempo de execução praticamente instantâneo, enquanto os métodos de busca local (*Swap*, *Insert*, *2-Opt* e *Variable Neighborhood Descent*) e a metaheurística *Simulated Annealing* foram avaliados sob um limite máximo de execução (*timeout*) de 3.600 segundos (1 hora). Os resultados experimentais foram obtidos considerando os ambientes computacionais descritos na Tabela 2.

5.1 Resultados das Heurísticas Construtivas

Com o objetivo de avaliar a qualidade das soluções iniciais geradas pelas heurísticas construtivas, foram realizados testes utilizando as heurísticas da Mediana (*Median*) e do Baricentro (*Barycenter*) em um conjunto de 10 instâncias de pequeno porte do problema OSCM. A análise considera o número de cruzamentos produzidos por cada método e sua proximidade em relação ao valor ótimo conhecido. Ressalta-se que as instâncias identificadas pela letra “M” não correspondem às mesmas instâncias utilizadas nos experimentos apresentados nas seções seguintes.

A Tabela 6 apresenta os resultados obtidos pelas heurísticas construtivas da Mediana (*Median*) e do Baricentro (*Barycenter*) para as instâncias avaliadas. De maneira geral, observa-se que a heurística da Mediana apresentou desempenho superior na maior parte dos casos, produzindo soluções mais próximas do ótimo global conhecido.

Tabela 6 – Comparação de resultados entre as heurísticas da Mediana e do Baricentro para 10 pequenas instâncias de teste

	Dimensões do Grafo			**Número de Cruzamentos**		
Instância	**Fixos** ($\|X\|$)	**Livres** ($\|Y\|$)	**Arestas** ($\|E\|$)	**Mediana**	**Baricentro**	**Ótimo**
Instância 01M	136	122	257	240	604 (240)	240
Instância 02M	351	327	677	650	3026 (650)	650
Instância 03M	14	20	59	495	490 (542)	489
Instância 04M	26	26	139	3409	3412 (3363)	3341
Instância 05M	31	37	250	11592	11471 (11488)	11450
Instância 06M	143	152	294	3162	3719 (4682)	3141
Instância 07M	244	215	458	6678	8140 (10277)	6641
Instância 08M	346	308	653	16995	20156 (24484)	16859
Instância 09M	375	344	718	25033	28707 (33393)	24661
Instância 10M	393	333	725	20830	24012 (30576)	20653

Nota: $|X|$ e $|Y|$ representam a quantidade de vértices nas camadas fixa e livre, respectivamente. A coluna "Ótimo" refere-se ao valor da solução exata conhecida. O valor apresentado em Baricentro() corresponde ao melhor resultado obtido após 1000 repetições aleatórias.

Nas instâncias 01M e 02M, a heurística da Mediana foi capaz de atingir exatamente o valor ótimo, obtendo respectivamente 240 e 650 cruzamentos. Em contrapartida, o método do Baricentro apresentou inicialmente soluções significativamente inferiores, embora tenha alcançado o ótimo após a realização de 1000 execuções aleatórias, conforme indicado entre parênteses na tabela. Em instâncias de maior porte, como 06M, 07M, 08M, 09M e 10M, a diferença entre os resultados das duas heurísticas tornou-se ainda mais evidente. Na instância 09M, por exemplo, a Mediana produziu uma solução com 25.033 cruzamentos, enquanto o Baricentro obteve inicialmente 28.707 cruzamentos, resultando em uma diferença superior a 3.600 cruzamentos. Esse comportamento indica maior robustez da heurística da Mediana em grafos mais densos e complexos.

Apesar disso, o método do Baricentro apresentou desempenho competitivo em algumas instâncias específicas. Na instância 03M, por exemplo, o Baricentro obteve um resultado ligeiramente melhor que a Mediana em sua execução inicial, produzindo 490 cruzamentos contra 495 da Mediana, ambos muito próximos do ótimo conhecido de 489 cruzamentos. Situação semelhante ocorreu na instância 05M, na qual o Baricentro apresentou uma solução discretamente superior.

Outro aspecto relevante refere-se ao impacto da aleatoriedade aplicada ao método do Baricentro. Observa-se que, em algumas instâncias, as múltiplas execuções permitiram encontrar soluções significativamente melhores do que aquelas produzidas pela versão determinística inicial. Entretanto, mesmo considerando os melhores resultados obtidos após as repetições aleatórias, a heurística da Mediana manteve desempenho superior ou equivalente na maioria das instâncias analisadas.

Dessa forma, os resultados indicam que a heurística da Mediana apresenta maior es-

tabilidade e qualidade na geração de soluções iniciais para o problema estudado, tornando-se uma alternativa mais adequada para servir como ponto de partida para métodos de refinamento e metaheurísticas aplicadas posteriormente.

A figura 14 ilustra a comparação dos melhores resultados obtidos para as 10 instâncias avaliadas.

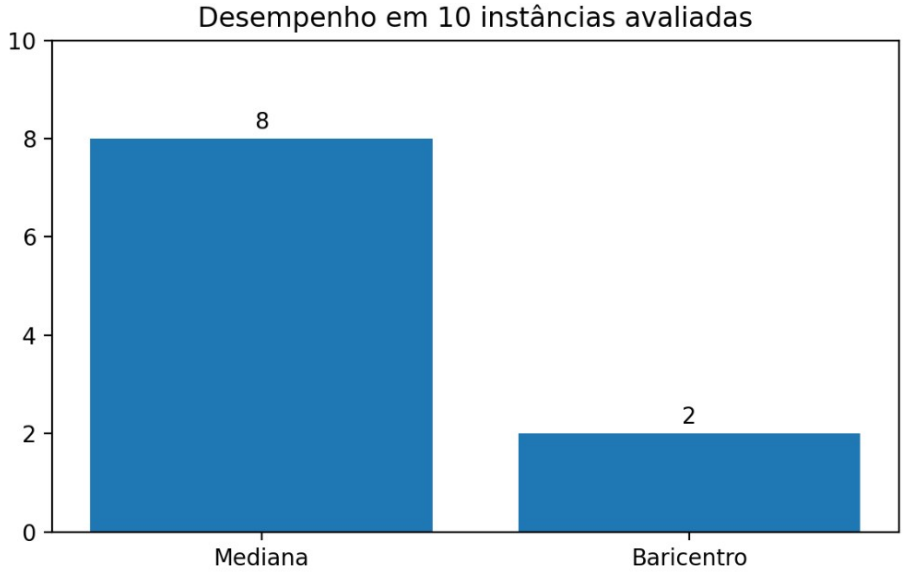

Figura 14 – Melhor desempenho obtido nas 10 instâncias avaliadas

Fonte: Elaborado pelo autor (2026)

5.2 Resultados das Heurísticas de Buscas Locais

5.2.1 Resultados com uma solução inicial aleatória

Esta seção apresenta os resultados obtidos pelos algoritmos de busca local em um conjunto de 20 instâncias do problema. Diferentemente dos experimentos anteriores, as buscas locais foram executadas sem o fornecimento de uma solução inicial gerada por heurísticas construtivas, sendo a solução inicial construída aleatoriamente. Foram avaliados os métodos *Swap*, *Insert*, *2-Opt* e *Variable Neighborhood Descent* (VND), utilizando como métrica de desempenho o número de cruzamentos obtido ao final da execução. Como referência para comparação, também são apresentados os resultados da heurística da Mediana e do Algoritmo Genético (AG).

A Tabela 7 mostra que, em instâncias de menor porte, as buscas locais foram capazes de alcançar a melhor solução conhecida, produzindo resultados equivalentes aos obtidos pelos métodos de referência. Entretanto, à medida que a complexidade das instâncias aumenta, observa-se uma maior dificuldade dos algoritmos em encontrar soluções competitivas quando iniciados a partir de uma ordenação aleatória.

De maneira geral, as vizinhanças *Insert* e *2-Opt* apresentaram desempenho superior ao *Swap*, produzindo soluções de melhor qualidade em diversas instâncias. O método VND, apesar de explorar múltiplas estruturas de vizinhança, não apresentou ganhos consistentes em relação às melhores buscas locais individuais, indicando que a ausência de uma solução inicial de qualidade pode limitar a efetividade do processo de refinamento.

Tabela 7 – Resultados globais com solução inicial aleatória para as 20 instâncias testadas

Instância	**Genético (Ref)**	***Median***	***Swap***	***Insert***	***2-Opt***	**VND**
1	12.432	12.432	12.432	12.432	12.432	12.432
2	20.022	20.022	997.362	20.022	20.022	997.362
3	2.862	2.862	2.862	2.862	2.862	2.862
4	2.408	2.408	2.408	2.408	2.408	2.408
5	7.536	7.536	7.536	7.536	7.536	7.536
6	2.166	2.166	2.166	2.166	2.166	2.166
7	10.837	10.837	11.228.927	10.941.710	10.673.758	11.320.576
8	1.962	1.962	1.962	1.962	1.962	1.962
9	177.606	354.023	71.519.228	900.999.038	66.267.084	71.519.228
10	317.359	321.971	318.029	317.508	320.297	317.402
11	2.030.125	2.067.826	2.078.819	2.055.383	2.075.785	2.092.606
12	2.464.107	2.512.773	2.663.069	2.656.651	2.637.411	2.736.868
13	3.361.112	3.443.013	4.405.305	4.351.701	3.979.256	4.303.569
14	1.442.485	1.442.579	1.442.485	1.442.485	1.442.485	1.442.485
15	10.852.981	10.858.472	12.047.389	10.852.981	10.852.981	11.526.801
16	205.310	207.212	207.111	205.311	214.570	206.312
17	779.264	785.537	3.269.529	3.095.501	2.542.926	3.158.373
18	976.915	984.400	4.079.785	3.922.870	3.215.399	3.951.443
19	11.031.567	11.308.771	13.751.778	13.612.780	12.487.975	13.394.910
20	182.579	185.073	183.070	182.742	184.049	182.758

A Figura 15 apresenta um resumo dos resultados obtidos pelas buscas locais nas 20 instâncias avaliadas.

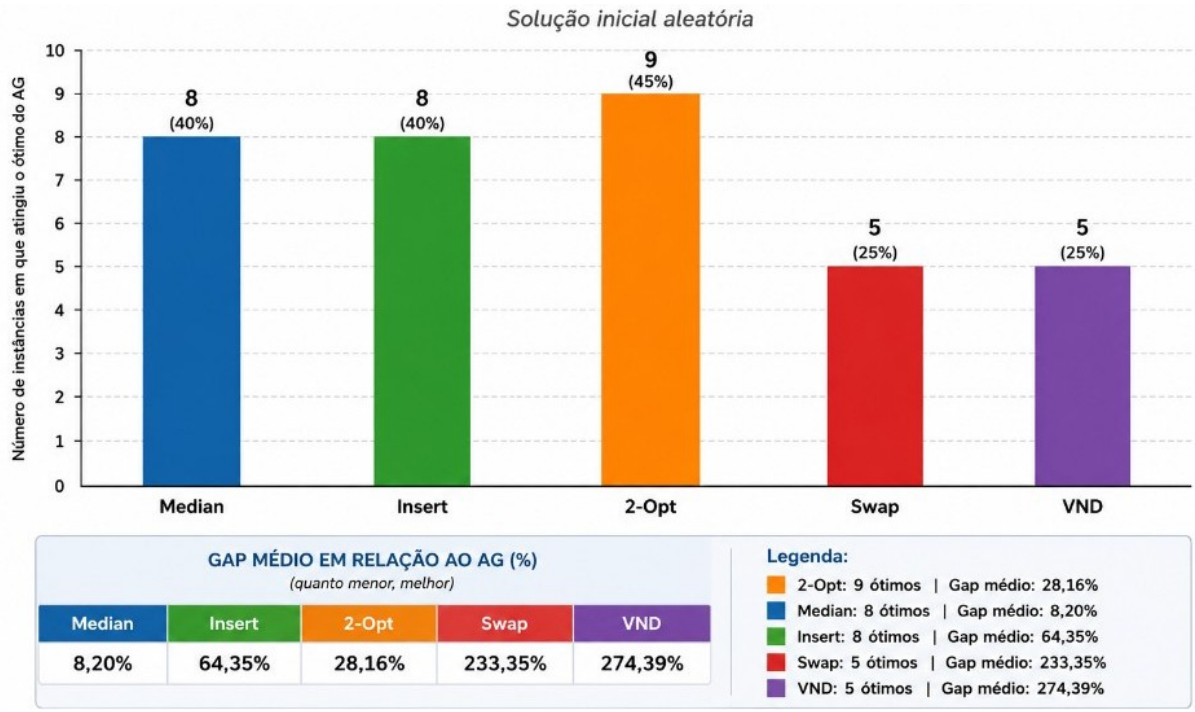

Figura 15 – Desempenho obtido nas 20 instâncias avaliadas

Fonte: Elaborado pelo autor (2026)

Observa-se que o método *2-Opt* foi o que mais frequentemente atingiu a solução ótima obtida pelo Algoritmo Genético (AG), alcançando esse resultado em 9 das 20 instâncias (45%). As heurísticas da Mediana e *Insert* obtiveram desempenho semelhante, encontrando a solução ótima em 8 instâncias (40%), enquanto *Swap* e VND atingiram o ótimo em apenas 5 instâncias (25%).

Além da frequência com que cada método encontrou a melhor solução conhecida, o *gap* médio em relação ao AG permite avaliar a qualidade das soluções nas instâncias em que o ótimo não foi alcançado. Nesse aspecto, a heurística da Mediana apresentou o menor desvio médio (8,20%), seguida pelo *2-Opt* (28,16%). Embora o *Insert* tenha encontrado o mesmo número de soluções ótimas que a Mediana, seu *gap* médio foi significativamente superior (64,35%), indicando menor robustez nas instâncias mais difíceis. Já os métodos *Swap* e VND apresentaram os maiores desvios médios (233,35% e 274,39%, respectivamente), evidenciando que, quando não encontram boas soluções, tendem a permanecer distantes da referência produzida pelo AG.

Em conjunto, esses resultados reforçam que o desempenho das buscas locais depende fortemente da qualidade da solução inicial. Embora algumas vizinhanças sejam capazes de recuperar a solução ótima em parte das instâncias, a elevada variação do *gap* médio demonstra que uma inicialização aleatória dificulta a convergência para regiões

promissoras do espaço de busca. Esse comportamento justifica a utilização de heurísticas construtivas para fornecer soluções iniciais de maior qualidade antes da aplicação das etapas de refinamento por busca local.

5.2.1.1 Análise Percentual: Refinamento vs. Mediana

A Tabela 8 apresenta o desempenho relativo das buscas locais e do VND em comparação direta com a heurística da Mediana. O cálculo é dado por $(\frac{\text{Busca Local}-\text{Mediana}}{\text{Mediana}}) \times 100$.

Tabela 8 – Variação percentual das buscas locais em relação à heurística da Mediana

Instância	***Swap*** **(%)**	***Insert*** **(%)**	***2-Opt*** **(%)**	**VND (%)**
1	+0,00%	+0,00%	+0,00%	+0,00%
2	+4.881,33%	+0,00%	+0,00%	+4.881,33%
3	+0,00%	+0,00%	+0,00%	+0,00%
4	+0,00%	+0,00%	+0,00%	+0,00%
5	+0,00%	+0,00%	+0,00%	+0,00%
6	+0,00%	+0,00%	+0,00%	+0,00%
7	+103.516,56%	+100.866,23%	+98.393,66%	+104.362,27%
8	+0,00%	+0,00%	+0,00%	+0,00%
9	+20.101,86%	+254.402,97%	+18.618,30%	+20.101,86%
10	-1,22%	-1,39%	-0,52%	-1,42%
11	+0,53%	-0,60%	+0,38%	+1,20%
12	+5,98%	+5,73%	+4,96%	+8,92%
13	+27,95%	+26,39%	+15,57%	+24,99%
14	-0,01%	-0,01%	-0,01%	-0,01%
15	+10,95%	-0,05%	-0,05%	+6,15%
16	-0,05%	-0,92%	+3,55%	-0,43%
17	+316,22%	+294,06%	+223,72%	+302,07%
18	+314,44%	+298,50%	+226,64%	+301,41%
19	+21,60%	+20,37%	+10,43%	+18,45%
20	-1,08%	-1,26%	-0,55%	-1,25%

5.2.1.2 Tempos de Execução

A Tabela 9 apresenta os tempos de execução dos métodos avaliados. Enquanto as heurísticas construtivas Baricentro e Mediana apresentaram tempos computacionais desprezíveis, as buscas locais, texecutadas sob a estratégia *Best Improvement*, frequentemente atingiram o limite de 3.600 segundos nas instâncias mais complexas, evidenciando o elevado custo computacional associado ao processo de refinamento das soluções.

Tabela 9 – Tempo de execução (em segundos) por algoritmo para cada instância

Instância	***Barycenter* (s)**	***Median* (s)**	***Swap* (s)**	***Insert* (s)**	***2-Opt* (s)**	**VND (s)**
1	0,0576	0,0129	1.681,6804	1.224,9861	939,3819	2.959,7688
2	0,1729	0,0171	3.601,4462	3.601,6872	3.602,3489	3.601,8022
3	0,0539	0,0057	12,7482	9,3023	6,0466	22,0887
4	0,0506	0,0063	8,1781	6,9288	3,8046	11,1600
5	0,0575	0,0074	867,5267	585,9154	464,9433	1.078,8573
6	0,0518	0,0062	5,2852	4,5456	2,4378	7,0104
7	0,0770	0,0102	3.600,6094	3.601,7047	3.602,1832	3.601,1235
8	0,0067	0,0032	210,9530	191,5062	163,8192	213,9289
9	3,2298	0,0708	3.621,8670	3.624,3039	3.644,2827	3.653,4684
10	0,1688	0,1774	1.162,8558	771,5273	880,6097	1.399,7452
11	0,0350	0,0046	3.600,3282	3.612,0163	3.604,4803	3.604,2191
12	0,0511	0,0052	3.607,1618	3.604,4916	3.604,4198	3.600,2817
13	0,2269	0,1045	3.603,0025	3.603,9877	3.600,3175	3.600,3513
14	0,2549	0,0973	2.225,6431	539,1522	1.786,8314	2.096,1562
15	0,2529	0,0947	3.600,9814	1.055,4837	2.951,0501	3.601,0390
16	0,0464	0,0038	3.600,3227	1.787,8484	3.600,2699	3.600,1234
17	0,2247	0,1062	3.600,4284	3.600,3276	3.600,3688	3.600,3879
18	0,1102	0,0066	3.600,1447	3.600,1179	3.600,3903	3.600,3909
19	0,0664	0,0171	3.600,1935	3.600,3931	3.600,5097	3.600,4718
20	0,1552	0,0852	252,2301	175,3435	219,6099	289,9410

5.2.2 Buscas Locais Iniciando a partir de soluções das Heurísticas Construtivas

Nesta seção, são analisados os resultados das buscas locais quando iniciadas a partir das soluções geradas pelas heurísticas construtivas Baricentro e Mediana. O objetivo é investigar de que forma a qualidade da solução inicial influencia o desempenho dos métodos de refinamento, tanto em termos da qualidade das soluções obtidas quanto do esforço computacional necessário para alcançá-las.

Além da solução inicial, também é analisado o impacto da estratégia de exploração da vizinhança. Para isso, são consideradas duas abordagens clássicas: *Best Improvement*, que examina toda a vizinhança e seleciona o movimento que produz a maior redução no número de cruzamentos, e *First Improvement*, que aceita o primeiro movimento capaz

de melhorar a solução corrente. A comparação entre essas estratégias permite avaliar o equilíbrio entre intensidade de busca, tempo de execução e qualidade das soluções finais.

5.2.2.1 *Best Improvement* (Melhor Melhora)

Nesta subseção são apresentados os resultados obtidos com a estratégia *Best Improvement*. Nessa abordagem, todas as soluções pertencentes à vizinhança são avaliadas a cada iteração, sendo selecionado o movimento que proporciona a maior redução no valor da função objetivo. Embora essa estratégia possua maior potencial de intensificação da busca, ela também demanda um esforço computacional mais elevado devido ao número de avaliações realizadas.

Os experimentos foram conduzidos utilizando duas soluções iniciais distintas: a solução gerada pela heurística do Baricentro (*initial_sol = 0*) e a solução gerada pela heurística da Mediana (*initial_sol = 1*). A Tabela 10 apresenta, para cada instância, o número de cruzamentos obtido pelas buscas locais *Swap*, *Insert*, *2-Opt* e VND, bem como os respectivos tempos de execução e a diferença em relação à melhor solução conhecida, representada pelo Algoritmo Genético.

De maneira geral, observa-se que a utilização da solução da Mediana como ponto de partida resultou em soluções significativamente superiores às obtidas a partir do Baricentro. Em diversas instâncias, os métodos de busca local foram capazes de preservar a qualidade da solução inicial da Mediana, alcançando resultados equivalentes à melhor solução conhecida. Em contrapartida, quando inicializados com o Baricentro, os algoritmos frequentemente apresentaram maiores desvios em relação à referência e, nas instâncias mais complexas, atingiram o limite máximo de execução sem convergir para soluções competitivas.

Entre as estruturas de vizinhança avaliadas, os métodos *Insert* e *2-Opt* apresentaram desempenho mais consistente, produzindo, em geral, soluções de melhor qualidade que a vizinhança *Swap*. O método VND, apesar de combinar múltiplas estruturas de vizinhança, não apresentou ganhos expressivos em relação às melhores buscas locais individuais, principalmente em função do elevado custo computacional associado à exploração sucessiva das diferentes vizinhanças.

Tabela 10 – Resultados completos das 20 instâncias – *Best Improvement*

nome	Genético	initial_sol	swap_cross	swap_t(s)	insert_cross	insert_t(s)	2opt_cross	2opt_t(s)	vnd_cross	vnd_t(s)	Swap_X_Ref	Insert_X_Ref	2opt_X_Ref	VND_X_Ref
1	12432	0	12432	3155.6021	12432	1255.8443	12432	1911.2915	12432	4050.1116	0	0	0	0
1	12432	1	12432	558.3377	12432	393.5025	12432	292.9343	12432	1277.1935	0	0	0	0
2	20022	0	318222	3602.5655	20022	3601.7247	52082	3601.8329	47742	3602.6261	-298200	0	-32060	-27720
2	20022	1	20022	2222.5261	20022	1824.7565	20022	1106.7621	20022	4015.3696	0	0	0	0
3	2862	0	2862	16.5090	2862	7.0254	2862	7.9656	2862	32.4104	0	0	0	0
3	2862	1	2862	4.7505	2862	3.2690	2862	2.4325	2862	10.3202	0	0	0	0
4	2408	0	2408	260.5466	2408	193.3843	2408	157.5736	2408	379.7184	0	0	0	0
4	2408	1	2408	1.4845	2408	1.0707	2408	0.6750	2408	2.8145	0	0	0	0
5	7536	0	538182	3600.5982	484662	3600.4960	355416	3600.6013	526336	3600.6774	-530646	-477126	-347880	-518800
5	7536	1	7536	85.8479	7536	60.5340	7536	44.7914	7536	184.1530	0	0	0	0
6	2166	0	2166	167.0534	2166	133.5949	2166	97.0133	2166	141.6179	0	0	0	0
6	2166	1	2166	0.8992	2166	0.7718	2166	0.4780	2166	1.8492	0	0	0	0
7	10837	0	53550	3600.9849	15164	3600.9573	27869	3600.8418	69380	3601.2066	-42713	-4327	-17032	-58543
7	10837	1	10837	542.2085	10837	317.2695	10837	333.7174	10837	1310.0145	0	0	0	0
8	1962	0	1962	14.1986	1962	9.8329	1962	7.9685	1962	14.8147	0	0	0	0
8	1962	1	1962	0.5501	1962	0.4807	1962	0.4582	1962	1.1483	0	0	0	0
9	177606	0	22443159	3728.5839	24801835	3649.5336	15365737	3634.7437	22885218	3717.9613	-22265553	-24624229	-15188131	-22707612
9	177606	1	354023	3624.3173	354023	3666.6517	354023	3647.2379	354023	3646.8914	-176417	-176417	-176417	-176417
10	317359	0	318303	910.9337	317399	690.7521	320856	519.4029	317479	1016.2105	-944	-40	-3497	-120
10	317359	1	318939	314.3800	317538	295.9105	320823	89.6617	317444	517.7443	-1580	-179	-3464	-85
11	2030125	0	2046256	3600.6078	2035679	3600.5671	2047830	3600.4702	2047169	3600.6348	-16131	-5554	-17705	-17044
11	2030125	1	2037872	3600.5446	2031363	3600.5614	2050320	2577.1166	2038531	3600.4218	-7747	-1238	-20195	-8406
12	2464107	0	2500656	3600.7930	2481695	3600.4870	2495112	3600.8825	2506939	3600.5528	-36549	-17588	-31005	-42832
12	2464107	1	2483226	3600.7538	2470445	3600.5713	2493120	3600.3557	2483719	3600.6890	-19119	-6338	-29013	-19612
13	3361112	0	3454855	3600.8152	3424749	3600.5464	3435991	3600.7295	3466338	3600.4898	-93743	-63637	-74879	-105226
13	3361112	1	3408328	3600.7274	3389205	3600.6316	3414744	3600.3146	3407177	3600.5824	-47216	-28093	-53632	-46065
14	1442485	0	1442485	909.8014	1442485	488.5315	1442485	627.5255	1442485	909.0171	0	0	0	0
14	1442485	1	1442485	10.5254	1442485	7.5495	1442485	5.9280	1442485	12.6194	0	0	0	0
15	10852981	0	10852981	2740.7958	10852981	958.7195	10852981	1396.1489	10852981	2138.8200	0	0	0	0
15	10852981	1	10852981	210.3835	10852981	116.0901	10852981	94.3745	10852981	209.5430	0	0	0	0
16	205310	0	206699	3600.6373	205322	1666.4268	210817	2932.8491	205323	3688.3753	-1389	-12	-5507	-13
16	205310	1	206913	165.1076	205319	221.2881	207071	57.1988	205322	392.0533	-1603	-9	-1761	-12
17	779264	0	1133008	3600.5311	1053560	3600.6979	1075758	3600.6469	1138885	3601.1600	-353744	-274296	-296494	-359621
17	779264	1	785171	3600.5024	779278	3600.5467	785396	1241.4981	785178	3600.5625	-5907	-14	-6132	-5914
18	976915	0	1385360	3600.5595	1320577	3600.5492	1365276	3600.7069	1450918	3600.7691	-408445	-343662	-388361	-474003
18	976915	1	983923	3600.7724	977549	3600.6694	984243	2692.2174	983937	3600.4067	-7008	-634	-7328	-7022
19	11031567	0	11215450	3601.0391	11208099	3600.4874	11202897	3600.8426	11314613	3600.4810	-183883	-176532	-171330	-283046
19	11031567	1	11200276	3600.6573	11155482	3600.4642	11210032	3600.8374	11187680	3600.7314	-168709	-123915	-178465	-156113
20	182579	0	183041	178.3871	182635	149.7312	184726	104.2817	182688	228.1422	-462	-56	-2147	-109
20	182579	1	183202	66.5491	182659	59.5683	184413	22.1179	182628	110.7629	-623	-80	-1834	-49

5.2.2.2 *First Improvement* (Primeira Melhora)

Esta subseção apresenta os resultados obtidos com a estratégia *First Improvement*. Diferentemente da abordagem anterior, o primeiro movimento que produz uma melhoria na solução corrente é imediatamente aceito, dispensando a avaliação completa da vizinhança. Essa característica reduz significativamente o custo computacional de cada iteração, podendo acelerar o processo de busca, especialmente em instâncias de grande porte.

Assim como na estratégia *Best Improvement*, os experimentos foram realizados utilizando as soluções geradas pelas heurísticas do Baricentro (*initial_sol = 0*) e da Mediana (*initial_sol = 1*) como pontos de partida. A Tabela 11 apresenta os resultados obtidos pelas buscas locais *Swap*, *Insert*, *2-Opt* e VND, incluindo o número de cruzamentos encontrados, os tempos de execução e a diferença em relação à melhor solução conhecida.

Os resultados evidenciam novamente a forte influência da solução inicial na qualidade das soluções finais. Quando inicializadas a partir da heurística da Mediana, as buscas locais produziram resultados próximos ou equivalentes à referência em diversas instâncias. Por outro lado, a utilização do Baricentro como solução inicial resultou em desempenhos significativamente inferiores em vários casos, principalmente nas instâncias de maior complexidade.

Comparando as estratégias de busca, observa-se que o *First Improvement* reduziu o esforço de exploração da vizinhança, mas nem sempre foi capaz de alcançar soluções de qualidade equivalente às obtidas pelo *Best Improvement*. Isso ocorre porque a aceitação

imediata de movimentos de melhoria tende a favorecer uma convergência mais rápida, porém com menor capacidade de exploração do espaço de busca. Ainda assim, a estratégia apresentou resultados competitivos em diversas instâncias, demonstrando ser uma alternativa interessante quando se busca reduzir o tempo computacional sem comprometer significativamente a qualidade das soluções.

Tabela 11 – Resultados completos das 20 instâncias - *First Improvement*

nome	Genético	initial_sol	swap_cross	swap_t(s)	insert_cross	insert_t(s)	2opt_cross	2opt_t(s)	vnd_cross	vnd_t(s)	Swap_X_Ref	Insert_X_Ref	2opt_X_Ref	VND_X_Ref
1	12432	0	583992	3601.2662	56872	3601.0489	276102	3601.1022	75462	3601.0757	-571560	-44440	-263670	-63030
1	12432	1	12432	558.7053	12432	411.3439	12432	310.2596	12432	1300.6761	0	0	0	0
2	20022	0	793942	3601.6626	179622	3601.6050	722122	3602.0522	694402	3602.0225	-773920	-159600	-702100	-674380
2	20022	1	20022	2280.1610	20022	1927.0308	20022	1145.5413	20022	4150.6914	0	0	0	0
3	2862	0	2862	2723.3628	2862	762.9638	2862	376.6951	34686	3600.3815	0	0	0	-31824
3	2862	1	2862	4.8640	2862	3.4179	2862	2.2404	2862	10.2433	0	0	0	0
4	2408	0	2408	3120.2324	2408	2775.5186	2408	912.2727	2408	3236.6457	0	0	0	0
4	2408	1	2408	1.4678	2408	1.1313	2408	0.7789	2408	2.8069	0	0	0	0
5	7536	0	995307	3600.5083	1319293	3600.3983	827789	3600.4180	1050438	3600.4322	-987771	-1311757	-820253	-1042902
5	7536	1	7536	85.2045	7536	60.9848	7536	43.6303	7536	196.8755	0	0	0	0
6	2166	0	2166	1059.8186	2166	1649.7873	2166	553.8489	2166	971.9419	0	0	0	0
6	2166	1	2166	0.9640	2166	0.7359	2166	0.4548	2166	1.9007	0	0	0	0
7	10837	0	1217385	3601.2017	692341	3600.7637	552739	3600.8549	318389	3601.1612	-1206548	-681504	-541902	-307552
7	10837	1	10837	566.6019	10837	338.9375	10837	351.1566	10837	1274.2254	0	0	0	0
8	1962	0	1962	495.6898	1962	283.7588	1962	169.6021	1962	479.7384	0	0	0	0
8	1962	1	1962	0.5567	1962	0.4365	1962	0.3361	1962	1.1542	0	0	0	0
9	177606	0	24722244	3695.8595	16772119	3637.3072	25573159	3645.1042	23678901	3723.5724	-24544638	-16594513	-25395553	-23501295
9	177606	1	354023	3703.0426	354023	3636.0575	354023	3639.0062	354023	3688.9215	-176417	-176417	-176417	-176417
10	317359	0	326315	3600.1389	317649	3600.2688	322410	2115.9429	325624	3600.2286	-8956	-290	-5051	-8265
10	317359	1	318732	1269.3525	317471	805.6021	320735	288.9789	317420	1480.7466	-1373	-112	-3376	-61
11	2030125	0	2115800	3600.3485	2127213	3600.3422	2105383	3600.2785	2118272	3600.2586	-85675	-97088	-75258	-88147
11	2030125	1	2061252	3600.1928	2062143	3600.4067	2063921	3600.3905	2061273	3600.3068	-31127	-32018	-33796	-31148
12	2464107	0	2563000	3600.2665	2550665	3600.1544	2565277	3600.2716	2579397	3600.3768	-98893	-86558	-101170	-115290
12	2464107	1	2507539	3600.4158	2506857	3600.3293	2510304	3600.3263	2507462	3600.1887	-43432	-42750	-46197	-43355
13	3361112	0	3522125	3600.3254	3529626	3600.4155	3513104	3600.3600	3542269	3600.1971	-161013	-168514	-151992	-181157
13	3361112	1	3439144	3600.3607	3440116	3600.1597	3442288	3600.3818	3438760	3600.3797	-78032	-79004	-81176	-77648
14	1442485	0	4302688	3600.3388	4206702	3600.3005	1442485	2973.8374	3900407	3600.4301	-2860203	-2764217	0	-2457922
14	1442485	1	1442485	10.4742	1442485	8.8022	1442485	5.4473	1442485	12.2521	0	0	0	0
15	10852981	0	28916705	3601.1067	24028843	3600.9771	21996942	3600.8836	27281284	3601.0579	-18063724	-13175862	-11143961	-16428303
15	10852981	1	10852981	188.5516	10852981	98.6363	10852981	90.4894	10852981	174.9049	0	0	0	0
16	205310	0	270108	3600.2554	266511	3600.2739	249597	3600.2614	276949	3600.0698	-64798	-61201	-44287	-71639
16	205310	1	206878	365.6703	205322	638.9359	207079	62.3033	205322	906.3080	-1568	-12	-1769	-12
17	779264	0	1220084	3600.4313	1254696	3600.2415	1208305	3600.4406	1205255	3600.3973	-440820	-475432	-429041	-425991
17	779264	1	785352	3600.2238	782655	3600.3992	785402	1679.3633	785355	3600.3659	-6088	-3391	-6138	-6091
18	976915	0	1494512	3600.4256	1512830	3600.2444	1432546	3600.2347	1482134	3600.8369	-517597	-535915	-455631	-505219
18	976915	1	984222	3600.3286	980956	3600.1997	984246	2582.2310	984227	3600.2524	-7307	-4041	-7331	-7312
19	11031567	0	11371609	3600.5766	11277021	3600.3403	11298930	3600.3893	11400185	3600.3400	-340042	-245454	-267363	-368618
19	11031567	1	11306605	3600.3019	11300791	3600.4264	11304685	3600.4522	11306504	3600.3044	-275038	-269224	-273118	-274937
20	182579	0	183226	1178.9164	182639	752.2512	184654	399.3897	182633	1419.7837	-647	-60	-2075	-54
20	182579	1	183171	235.4321	182615	158.6117	184227	67.3274	182611	271.2250	-592	-36	-1648	-32

5.2.2.3 Análise Comparativa das Estratégias *Best Improvement* e *First Improvement*

A Figura 16 apresenta a relação entre a qualidade das soluções obtidas, expressa pelo Desvio Relativo Médio (DMR), que representa a proximidade em relação à melhor solução de referência conhecida, e o tempo computacional das estratégias *Best Improvement* e *First Improvement* aplicadas às diferentes buscas locais. De maneira geral, a estratégia *Best Improvement* apresentou soluções de melhor qualidade, embora com maior custo computacional, enquanto o *First Improvement* obteve resultados competitivos com menor esforço de busca. Entre as vizinhanças avaliadas, os métodos *Insert* e *2-Opt* destacaram-se pelo melhor equilíbrio entre qualidade da solução e tempo de execução. Por outro lado, o VND, apesar do elevado custo computacional decorrente da exploração sucessiva de múltiplas vizinhanças, não apresentou ganhos proporcionais em relação às melhores buscas locais individuais. Já a vizinhança *Swap* apresentou, de forma geral, os resultados menos competitivos em termos da qualidade das soluções obtidas.

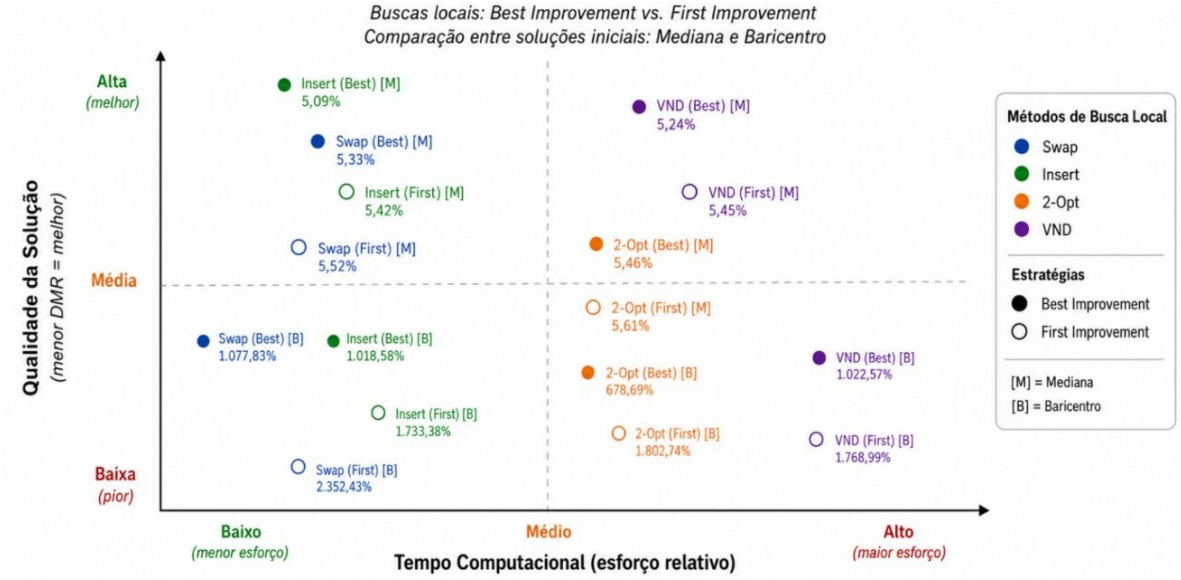

Figura 16 – Relação entre Tempo Computacional e Qualidade da Solução

Fonte: Elaborado pelo autor (2026)

5.3 Resultados do *Simulated Annealing*

Os resultados apresentados nas seções anteriores evidenciaram que o desempenho das buscas locais está fortemente associado à qualidade da solução inicial utilizada. Embora métodos como *Insert*, *2-Opt* e VND tenham obtido resultados satisfatórios quando iniciados a partir de soluções geradas pela heurística da Mediana, observou-se uma limitação comum a todas as abordagens baseadas exclusivamente em busca local: a dificuldade em escapar de ótimos locais, especialmente em instâncias de maior porte e complexidade.

Com o objetivo de ampliar a capacidade de exploração do espaço de busca e reduzir a dependência da solução inicial, foi avaliada a metaheurística *Simulated Annealing* (SA). Inspirado no processo físico de recozimento de metais, o método permite a aceitação controlada de soluções temporariamente piores ao longo da execução, favorecendo a diversificação da busca e aumentando a probabilidade de escapar de ótimos locais profundos.

Nesta seção são apresentados os resultados obtidos pelo SA para as 20 instâncias do Problema de Minimização de Cruzamentos Unilaterais. A análise considera a qualidade das soluções encontradas, medida pelo número de cruzamentos obtido ao final da execução, bem como a comparação com os resultados produzidos pelas heurísticas construtivas, pelas buscas locais e pelo algoritmo genético utilizado como referência. Dessa forma, busca-se avaliar a capacidade da metaheurística em produzir soluções competitivas e superar as limitações observadas nos métodos de refinamento puramente locais.

5.3.1 Instâncias de teste

A análise do desempenho das heurísticas e do *Simulated Annealing* depende diretamente das características estruturais das instâncias avaliadas. O conjunto de testes utilizado neste trabalho foi composto por instâncias com diferentes dimensões e níveis de complexidade, possibilitando analisar o comportamento dos métodos em cenários variados e representativos.

A Figura 17 apresenta a caracterização das instâncias, indicando a quantidade de vértices do conjunto fixo (*fixed*), a quantidade de vértices do conjunto livre (*free*) e o número total de arestas (*edges*).

id	***fixed***	***free***	***edges***
1	6328	6218	12545
2	10153	10013	20165
3	1485	1433	2917
4	1078	991	2148
5	3638	3426	7191
6	949	870	1898
7	5580	5356	10973
8	791	718	1581
9	45740	45444	91183
10	794	895	1688
11	1467	1483	3723
12	1608	1647	4099
13	1920	1998	4890
14	847	839	15362
15	982	973	30672
16	1098	1109	2206
17	2245	2224	4468
18	2502	2371	4872
19	2125	2096	8257
20	639	626	1264

Figura 17 – Caracterização das instâncias de teste utilizadas nos experimentos

Fonte: Elaborado pelo autor (2026)

Cabe destacar que as instâncias 9, 14 e 15 se destacam como as mais desafiadoras do conjunto de testes. A instância 9 apresenta mais de 91 mil arestas, valor significativa-

mente superior ao das demais instâncias, resultando em um espaço de busca consideravelmente maior e em um aumento substancial da quantidade potencial de cruzamentos. Já as instâncias 14 e 15, embora possuam menor número de vértices, caracterizam-se por uma elevada densidade de conexões, com 15.362 e 30.672 arestas, respectivamente. Essa alta densidade implica um maior grau de interação entre os vértices e aumenta a complexidade da avaliação e da otimização das ordenações. Como consequência, essas instâncias tendem a apresentar maior dificuldade para os algoritmos de busca local, que podem ficar mais suscetíveis à convergência prematura em ótimos locais, justificando os resultados inferiores observados em alguns métodos quando comparados às instâncias menos complexas.

5.3.2 Tabela coparativa dos resultados

A Tabela 12 apresenta os resultados de cruzamentos obtidos em todas as 20 instâncias do experimento com SA comparando as heurísticas de refinamento com a metaheurística, após a execução com limite de 3600 segundos (1 hora).

Tabela 12 – Resultados globais de cruzamentos para as 20 instâncias testadas (3600 segundos)

Instância	Genético (Ref)	*Median*	*Swap*	*Insert*	*2-Opt*	VND	*Simulated Annealing*
1	12.432	12.432	12.432	12.432	12.432	12.432	12.432
2	20.022	20.022	997.362	20.022	20.022	997.362	20.022
3	2.862	2.862	2.862	2.862	2.862	2.862	2.862
4	2.408	2.408	2.408	2.408	2.408	2.408	2.408
5	7.536	7.536	7.536	7.536	7.536	7.536	7.536
6	2.166	2.166	2.166	2.166	2.166	2.166	2.166
7	10.837	10.837	11.228.927	10.941.710	10.673.758	11.320.576	10.837
8	1.962	1.962	1.962	1.962	1.962	1.962	1.962
9	177.606	354.023	71.519.228	900.999.038	66.267.084	71.519.228	354.023
10	317.359	321.971	318.029	317.508	320.297	317.402	317.372
11	2.030.125	2.067.826	2.078.819	2.055.383	2.075.785	2.092.606	2.030.358
12	2.464.107	2.512.773	2.663.069	2.656.651	2.637.411	2.736.868	2.464.215
13	3.361.112	3.443.013	4.405.305	4.351.701	3.979.256	4.303.569	3.361.411
14	1.442.485	1.442.579	1.442.485	1.442.485	1.442.485	1.442.485	1.442.508
15	10.852.981	10.858.472	12.047.389	10.852.981	10.852.981	11.526.801	10.854.893
16	205.310	207.212	207.111	205.311	214.570	206.312	205.349
17	779.264	785.537	3.269.529	3.095.501	2.542.926	3.158.373	779.470
18	976.915	984.400	4.079.785	3.922.870	3.215.399	3.951.443	977.144
19	11.031.567	11.308.771	13.751.778	13.612.780	12.487.975	13.394.910	11.032.163
20	182.579	185.073	183.070	182.742	184.049	182.758	182.584

Nota 1: *Simulated Annealing*: resultados obtidos com a configuração pelo *irace* descrito no seção 4.5.5 desse trabalho.

Nota 2: Os resultados das buscas locais (*Median*, *Swap*, *Insert* e *2-Opt*), bem como do método VND, correspondem à estratégia de exploração de vizinhança *Best Improvement*.

5.3.3 Tempo de execução

A Tabela 13 apresenta o tempo de processamento, em segundos, requerido por cada método avaliado, incluindo as heurísticas construtivas e as buscas locais. Ressalta-se

que as buscas locais e as metaheurísticas foram configuradas com um limite máximo de execução de 3.600 segundos por instância, garantindo uniformidade na comparação do desempenho computacional entre os métodos.

Tabela 13 – Tempo de execução (em segundos) por algoritmo para cada instância

Instância	*Barycenter*	*Median*	*Swap*	*Insert*	*2-Opt*	VND	*Simulated Annealing*
1	0,0576	0,0129	1681,6804	1224,9861	939,3819	2959,7688	1312,9001
2	0,1729	0,0171	3601,4462	3601,6872	3602,3489	3601,8022	1926,4665
3	0,0539	0,0057	12,7482	9,3023	6,0466	22,0887	270,2097
4	0,0506	0,0063	8,1781	6,9288	3,8046	11,1600	82,1875
5	0,0575	0,0074	867,5267	585,9154	464,9433	1078,8573	737,1780
6	0,0518	0,0062	5,2852	4,5456	2,4378	7,0104	245,8445
7	0,0770	0,0102	3600,6094	3601,7047	3602,1832	3601,1235	1161,2143
8	0,0067	0,0032	210,9530	191,5062	163,8192	213,9289	47,8996
9	3,2298	0,0708	3621,8670	3624,3039	3644,2827	3653,4684	3666,1134
10	0,1688	0,1774	1162,8558	771,5273	880,6097	1399,7452	43,3056
11	0,0350	0,0046	3600,3282	3612,0163	3604,4803	3604,2191	269,3049
12	0,0511	0,0052	3607,1618	3604,4916	3604,4198	3600,2817	317,4493
13	0,2269	0,1045	3603,0025	3603,9877	3600,3175	3600,3513	422,5052
14	0,2549	0,0973	2225,6431	539,1522	1786,8314	2096,1562	244,6177
15	0,2529	0,0947	3600,9814	1055,4837	2951,0501	3601,0390	276,3362
16	0,0464	0,0038	3600,3227	1787,8484	3600,2699	3600,1234	100,3939
17	0,2247	0,1062	3600,4284	3600,3276	3600,3688	3600,3879	507,5643
18	0,1102	0,0066	3600,1447	3600,1179	3600,3903	3600,3909	515,4157
19	0,0664	0,0171	3600,1935	3600,3931	3600,5097	3600,4718	467,2280
20	0,1552	0,0852	252,2301	175,3435	219,6099	289,9410	18,2798

Nota 1: *Simulated Annealing*: resultados obtidos com a configuração pelo *irace* descrito no seção 4.5.5 desse trabalho.

Nota 2: Os resultados das buscas locais (*Median*, *Swap*, *Insert* e *2-Opt*), bem como do método VND, correspondem à estratégia de exploração de vizinhança *Best Improvement*.

5.4 Desvio Médio Relativo em Relação à Referência

Com o objetivo de avaliar a qualidade das soluções obtidas pelos diferentes métodos propostos, foi calculado o Desvio Médio Relativo (DMR) em relação às soluções de referência geradas pelo Algoritmo Genético (AG). Essa métrica expressa, em termos percentuais, o afastamento médio entre o valor obtido por cada método e a melhor solução conhecida para cada instância, permitindo uma comparação padronizada entre as abordagens analisadas. Quanto menor o valor do DMR, mais próxima a solução se encontra da referência e, consequentemente, maior é sua qualidade.

Cabe ressaltar que o DMR é uma medida baseada na média aritmética e, portanto, pode ser influenciado pela presença de valores extremos. Em situações nas quais um método apresenta desempenho muito inferior em poucas instâncias específicas, o valor médio pode ser significativamente elevado, mascarando um comportamento satisfatório observado na maior parte dos casos. Dessa forma, os resultados devem ser interpretados considerando também a consistência geral das soluções produzidas.

A Figura 18 apresenta os valores de DMR obtidos por todas as heurísticas construtivas, buscas locais e pela metaheurística *Simulated Annealing*.

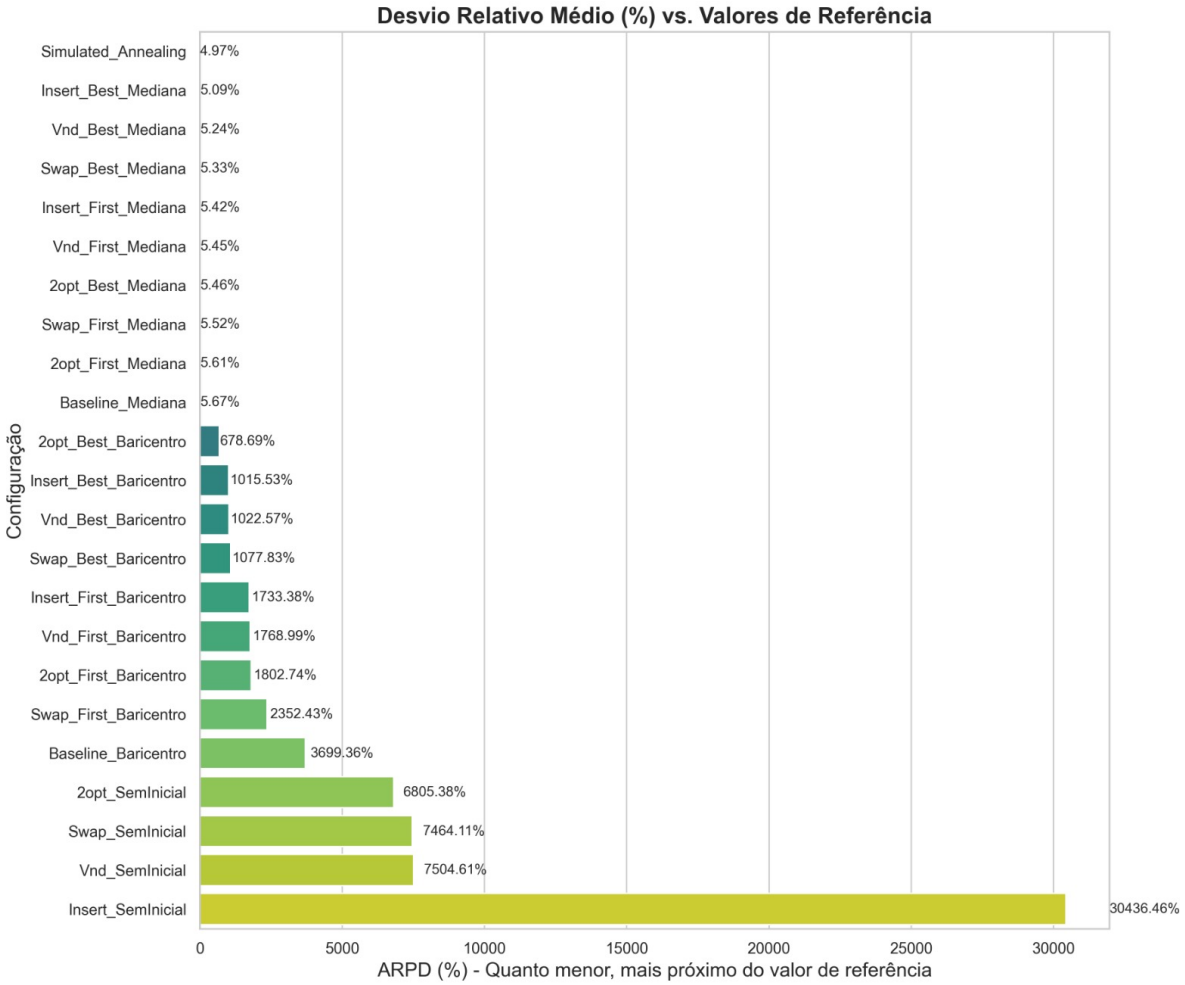

Figura 18 – Desvio Médio Relativo (DMR) em relação às soluções de referência

Fonte: Elaborado pelo autor (2026)

Os resultados evidenciam três grupos de desempenho claramente distintos. O primeiro é composto pelas buscas locais executadas a partir de soluções aleatórias, que apresentaram os maiores desvios médios relativos. Nessa categoria, os valores variaram de 6.805,38% a 30.436,46%, demonstrando que a aplicação isolada dos operadores de busca local não foi suficiente para conduzir o processo a regiões promissoras do espaço de busca. Em particular, o método *Insert* apresentou o pior desempenho, alcançando um desvio superior a 30 mil por cento, evidenciando a forte dependência dessas abordagens em relação à qualidade da solução inicial.

O segundo grupo corresponde às buscas locais inicializadas pela heurística construtiva Baricentro. Embora tenha ocorrido uma redução significativa dos desvios em comparação às soluções aleatórias, os resultados ainda permaneceram consideravelmente distantes

das soluções de referência. Os valores observados variaram entre 678,69% e 3.699,36%, indicando que, apesar de produzir soluções iniciais superiores às aleatórias, a heurística Baricentro não gera pontos de partida suficientemente próximos dos melhores resultados conhecidos.

O terceiro e melhor grupo é formado pelas abordagens baseadas na heurística construtiva Mediana. Nesse caso, os desvios médios relativos permaneceram concentrados em uma faixa bastante estreita, entre 5,09% e 5,67%. A própria heurística Mediana, sem qualquer etapa de refinamento, apresentou DMR de 5,67%, valor significativamente inferior aos obtidos pelas demais heurísticas construtivas. Esse resultado evidencia a elevada qualidade das soluções iniciais geradas por esse método.

Entre as buscas locais aplicadas sobre a solução produzida pela Mediana, o operador *Insert Best Improvement* apresentou o melhor desempenho, alcançando DMR de 5,09%, seguido por *VND Best Improvement* (5,24%) e *Swap Best Improvement* (5,33%). Observa-se ainda que as diferenças entre os métodos foram pequenas, todas inferiores a 0,6 ponto percentual. Esse comportamento sugere que, quando a solução inicial já possui elevada qualidade, os diferentes mecanismos de exploração de vizinhança tendem a convergir para soluções finais muito semelhantes.

O melhor resultado geral foi obtido pela metaheurística *Simulated Annealing*, que atingiu um DMR de 4,97%. Embora a diferença para o melhor método de busca local tenha sido relativamente pequena, de apenas 0,12 ponto percentual, esse resultado confirma a capacidade da metaheurística de escapar de ótimos locais e explorar regiões adicionais do espaço de busca. Tal característica permitiu produzir soluções ligeiramente mais próximas das referências do que aquelas obtidas exclusivamente por métodos de busca local.

De forma geral, os resultados demonstram que a qualidade da solução inicial exerce influência decisiva no desempenho dos algoritmos de otimização empregados. Enquanto as buscas iniciadas a partir de soluções aleatórias ou geradas pela heurística Baricentro permaneceram distantes dos valores de referência, as abordagens baseadas na heurística Mediana alcançaram desvios substancialmente menores. Além disso, o desempenho superior do *Simulated Annealing* reforça o potencial das metaheurísticas como mecanismos de refinamento capazes de explorar regiões promissoras do espaço de busca e produzir soluções ainda mais próximas das melhores soluções conhecidas.

5.5 *Simulated Annealing* com 200 instâncias

Os resultados obtidos na seção anterior evidenciaram a superioridade do *Simulated Annealing* (SA) em relação às demais abordagens avaliadas no conjunto inicial de 20 instâncias. Considerando o Desvio Médio Relativo (DMR) em relação às soluções de referência, o SA apresentou o melhor desempenho global, alcançando o menor desvio médio

entre todos os métodos analisados. Esse resultado demonstrou o potencial da combinação entre a heurística construtiva da Mediana e o mecanismo de refinamento proporcionado pelo SA para a resolução do *One-Sided Crossing Minimization*.

Com o objetivo de verificar se esse desempenho se mantém em um conjunto mais amplo e representativo de instâncias, foi conduzida uma nova etapa experimental utilizando novamente o *benchmark* oficial do PACE *Challenge* 2024 (KINDERMANN; KLUTE; TERZIADIS, 2024), agora com o conjunto completo de 200 instâncias com diferentes níveis de complexidade.

Para fins de comparação, foram utilizados os resultados obtidos pelo algoritmo *MAEDM-OCM*, desenvolvido pela equipe CIMAT, vencedora da categoria *Heuristic Track* do PACE *Challenge* 2024 (SEGURA *et al.*, 2024). Esse método consiste em um algoritmo memético que combina mecanismos de preservação da diversidade populacional com uma estratégia de Busca Local Iterada (*Iterated Local Search* - ILS), visando intensificar a exploração do espaço de busca. As melhores soluções reportadas pelos autores foram adotadas como referência para o cálculo do *gap* relativo, permitindo avaliar a proximidade das soluções obtidas pelo *Simulated Annealing*.

Cada instância foi executada dez vezes de forma independente, adotando-se um limite máximo de 300 segundos por execução, em conformidade com os critérios estabelecidos pela competição. Para cada instância foram consideradas três soluções distintas: a solução inicial produzida pela heurística da Mediana, a melhor solução encontrada pelo *Simulated Annealing* após o processo de refinamento e a melhor solução conhecida (*Best Known Solution* - BKS), obtida pelo algoritmo *MAEDM-OCM*.

A Tabela 14 apresenta os resultados obtidos para 196 das 200 instâncias disponibilizadas pelo *benchmark*. As instâncias 10, 44, 109 e 110 não puderam ser processadas devido às limitações computacionais do ambiente experimental. Como a implementação utiliza uma matriz de cruzamentos para acelerar a avaliação das soluções, a construção dessa estrutura para essas instâncias exigiu uma quantidade de memória superior aos 16 GB de RAM disponíveis na máquina utilizada, inviabilizando sua execução.

Cada um dos dois blocos da tabela contém 100 instâncias e apresenta tanto as características estruturais dos grafos quanto os resultados obtidos pelos métodos avaliados. As colunas *fixed*, *free* e *edges* representam, respectivamente, o número de vértices da camada fixa, o número de vértices da camada livre e a quantidade total de arestas da instância. A coluna BKS corresponde à melhor solução conhecida para cada caso. Em seguida, são apresentados os resultados da heurística da Mediana, incluindo o número de cruzamentos obtido (*cross*) e o respectivo *gap*. Por fim, são exibidos os resultados do *Simulated Annealing*, contendo o melhor número de cruzamentos encontrado entre as dez execuções (*best cross*), a média de cruzamentos obtida (*avg cross*), o *gap* relativo da melhor solução (*gap best*) e o tempo médio de execução em segundos.

Tabela 14 – Resultados completos do *Simulated Annealing* para as 200 instâncias

ID	fixed	free	edges	BKS	Mediana		SA			
					cross	gap	best cross	avg cross	gap best	time(s)
1	6328	6218	12545	12432	12432	**0,0000%**	12432	12432	**0,0000%**	301.35
2	10153	10013	20165	20022	20022	**0,0000%**	20022	20022	**0,0000%**	302.77
3	1485	1433	2917	2862	2862	**0,0000%**	2862	2862	**0,0000%**	108.3
4	1078	991	2148	2408	2408	**0,0000%**	2408	2408	**0,0000%**	38.35
5	3638	3426	7191	7536	7536	**0,0000%**	7536	7536	**0,0000%**	300.63
6	949	870	1898	2166	2166	**0,0000%**	2166	2166	**0,0000%**	26.59
7	5580	5356	10973	10837	10837	**0,0000%**	10837	10837	**0,0000%**	301.79
8	791	718	1581	1962	1962	**0,0000%**	1962	1962	**0,0000%**	20.99
9	45740	45444	91183	177606	354023	99,3305%	354023	354023	99,3305%	328.95
10	131315	130809	262123	—	—	—	—	—	—	—
11	1467	1483	3723	2030125	2067826	1,8571%	2030174	2030283.7	0,0024%	105.94
12	1608	1647	4099	2464107	2512773	1,9750%	2464139	2464162.1	0,0013%	161.66
13	1920	1998	4890	3361112	3443013	2,4367%	3361300	3361398.6	0,0056%	227.61
14	847	839	15362	1442485	1442579	0,0065%	1442485	1442531.8	**0,0000%**	25.94
15	982	973	30672	10852981	10858472	0,0506%	10854563	10855064.7	0,0146%	31.2
16	1098	1109	2206	205310	207212	0,9264%	205335	205345.1	0,0122%	47.32
17	2245	2224	4468	779264	785537	0,8050%	779400	779437	0,0175%	265.65
18	2502	2371	4872	976915	984400	0,7662%	977152	977180.3	0,0243%	290.88
19	2125	2096	8257	11031567	11308771	2,5128%	11032036	11032163.1	0,0043%	239.76
20	639	626	1264	182579	185073	1,3666%	182583	182586.7	0,0022%	18.3
21	794	895	1688	317359	321971	1,4532%	317363	317373.1	0,0013%	29.24
22	1070	1070	3210	1565493	1584204	1,1952%	1565537	1565567.2	0,0028%	42.44
23	1564	1564	4692	3362106	3394604	0,9666%	3362371	3362426.8	0,0079%	145.01
24	1828	1828	5484	4450936	4494664	0,9824%	4451259	4451439	0,0073%	185.28
25	6889	6682	20273	265109	270958	2,2063%	270958	270958	2,2063%	302.22
26	10343	10092	30536	398836	408115	2,3265%	408115	408115	2,3265%	304.92
27	7511	7804	15237	117089	119676	2,2094%	119676	119676	2,2094%	302.15
28	10040	10453	20447	156448	160085	2,3247%	160085	160085	2,3247%	303.6
29	15829	2461	18289	0	0	**0,0000%**	0	0	**0,0000%**	291.14
30	17219	2657	19875	0	0	**0,0000%**	0	0	**0,0000%**	301.11
31	657	657	1314	213510	213510	**0,0000%**	213510	213510	**0,0000%**	18.97
32	745	745	1490	280916	280916	**0,0000%**	280916	280916	**0,0000%**	21.97
33	922	922	1844	440799	440799	**0,0000%**	440799	440799	**0,0000%**	29.34
34	13122	1747	13122	0	0	**0,0000%**	0	0	**0,0000%**	171.01
35	21362	2847	21362	0	0	**0,0000%**	0	0	**0,0000%**	301.22
36	1599	1548	1991	432850	438726	1,3575%	432865	432894.1	0,0035%	129.35
37	1597	1564	13283	33588812	34272590	2,0357%	33589012	33589392.4	0,0006%	144.05
38	1629	1692	12744	30602516	31351334	2,4469%	30602813	30603000	0,0010%	165.05
39	2484	2596	9919	748666	823173	9,9520%	750466	750727	0,2324%	300.7
40	2860	2943	11305	229527	251177	9,4324%	232441	232629.7	1,2696%	300.87
41	4800	4800	19004	1333254	1333254	**0,0000%**	1333254	1333254	**0,0000%**	301.6
42	2345	2345	9243	447480	447511	0,0069%	447511	447511	0,0069%	276.59
43	3321	3321	13121	772760	772797	0,0048%	772797	772797	0,0048%	300.79
44	65536	65536	1114112	—	—	—	—	—	—	—
45	256	256	2304	1019861	1046100	2,5728%	1019867	1019871.9	0,0006%	6.07
46	16077	16077	33433	30871	30871	**0,0000%**	30871	30871	**0,0000%**	304.79
47	794	794	1891	1281	1281	**0,0000%**	1281	1281	**0,0000%**	22.5
48	1234	1234	2967	1965	1965	**0,0000%**	1965	1965	**0,0000%**	52.38
49	493	493	1189	779	779	**0,0000%**	779	779	**0,0000%**	13.05
50	975	975	2344	1552	1552	**0,0000%**	1552	1552	**0,0000%**	33.09
51	1086	1086	2631	1709	1709	**0,0000%**	1709	1709	**0,0000%**	43.73
52	2095	2074	4168	1552645	1642504	5,7875%	1552771	1552792.6	0,0081%	241.12
53	912	886	1797	274412	290621	5,9068%	274430	274435.9	0,0066%	26.31
54	1183	1230	2412	528705	559467	5,8184%	528733	528745.1	0,0053%	58.62
55	1644	1634	3277	1282785	1301429	1,4534%	1282813	1282853.5	0,0022%	167.87
56	1152	1201	2352	663370	672482	1,3736%	663386	663401.4	0,0024%	57.95
57	2006	2006	4010	252004	754005	199,2036%	252004	252004	**0,0000%**	223.1
58	2606	2606	5210	425104	1272705	199,3867%	425104	425104	**0,0000%**	300.59
59	2934	2934	5866	538756	1613333	199,4552%	538756	538756	**0,0000%**	300.36
60	1442	1472	3028	1247557	1270398	1,8309%	1247570	1247608.4	0,0010%	121.04
61	804	801	1696	360752	367541	1,8819%	360758	360761.5	0,0017%	23.56
62	943	907	2024	551792	565349	2,4569%	551803	551815.7	0,0020%	27.72
63	1186	1200	1719	360851	367369	1,8063%	360854	360858.6	0,0008%	60.83
64	1580	1535	2146	549411	557022	1,3853%	549427	549464.1	0,0029%	145.48
65	518	504	763	72910	73948	1,4237%	72910	72910	**0,0000%**	14.4
66	1013	1077	1074	103362	104697	1,2916%	103362	103370.4	**0,0000%**	48.6
67	1719	1786	2158	462690	469068	1,3785%	462692	462739.4	0,0004%	190.05
68	1840	1866	2310	567661	575275	1,3413%	567690	567720.7	0,0051%	208.05
69	26294	7532	26294	717380	739508	3,0846%	739508	739508	3,0846%	303.08
70	30514	8706	30514	834407	860251	3,0973%	860251	860251	3,0973%	304.71
71	11678	6020	20924	562159	580144	3,1993%	580144	580144	3,1993%	302.66
72	15818	8772	30611	829117	854823	3,1004%	854823	854823	3,1004%	304.78
73	1492	1445	4427	2833849	2896812	2,2218%	2833895	2833918.3	0,0016%	120.91
74	1517	1543	4610	3089399	3154664	2,1125%	3089455	3089545	0,0018%	142.79
75	1980	1989	6024	5452984	5579909	2,3276%	5453120	5453195.8	0,0025%	224.87
76	498	528	1025	117153	119237	1,7789%	117153	117169.8	**0,0000%**	14.93
77	654	645	1298	201996	205766	1,8664%	201996	202002.4	**0,0000%**	18.44
78	1508	1466	2973	1064781	1079823	1,4127%	1064830	1064895.9	0,0046%	119.78
79	525	539	1063	135176	137384	1,6334%	135176	135183.8	**0,0000%**	15.12
80	581	593	1173	159436	161959	1,5825%	159438	159439.9	0,0013%	16.96
81	1230	1277	2506	739955	750044	1,3635%	739973	740040.1	0,0024%	72.33
82	1700	1667	3313	1490828	1518689	1,8688%	1490979	1491008	0,0101%	173.95
83	1738	1731	3466	1660982	1698369	2,2509%	1661023	1661099.6	0,0025%	182.67
84	1804	1832	3549	1719214	1762167	2,4984%	1719254	1719300.6	0,0023%	199.39
85	674	673	1759	469640	481048	2,4291%	469652	469666.6	0,0026%	19.34
86	865	884	2700	1117722	1144515	2,3971%	1117734	1117742	0,0011%	26.58
87	971	989	2630	1019257	1041757	2,2075%	1019273	1019291	0,0016%	32.82
88	1179	1184	4051	223970	229120	2,2994%	223980	223989.6	0,0045%	52.03
89	1531	1537	5157	365618	374670	2,4758%	365659	365669.8	0,0112%	122.54
90	720	724	2532	78243	80194	2,4935%	78243	78245.5	**0,0000%**	20.36
91	1033	1038	3335	159343	162756	2,1419%	159352	159359.2	0,0056%	34.76
92	519	522	1705	59032	60509	2,5020%	59032	59032.3	**0,0000%**	14.21
93	769	773	2389	131308	134666	2,5573%	131317	131321	0,0069%	22.78
94	1352	1377	4749	3599098	3687888	2,4670%	3599147	3599178.7	0,0014%	110.92
95	1463	1429	3906	2183248	2227624	2,0326%	2183308	2183351.1	0,0027%	128.65
96	2005	2097	2161	452004	456050	0,8951%	452055	452086.3	0,0113%	245.91
97	1180	1110	3431	1948997	1981032	1,6437%	1949004	1949026.6	0,0004%	49.49
98	1736	1719	5182	4484095	4559102	1,6727%	4484137	4484212.4	0,0009%	177.06
99	7006	9340	18680	51792631	51792631	**0,0000%**	51792631	51792631	**0,0000%**	303.11
100	8794	11724	23448	81607829	81607829	**0,0000%**	81607829	81607829	**0,0000%**	304.43

ID	fixed	free	edges	BKS	Mediana		SA			
					cross	gap	best cross	avg cross	gap best	time(s)
101	7626	7505	15130	15006	15006	**0,0000%**	15006	15006	**0,0000%**	302.12
102	8911	8780	17690	17556	17556	**0,0000%**	17556	17556	**0,0000%**	302.48
103	2080	2018	4097	4032	4032	**0,0000%**	4032	4032	**0,0000%**	227.32
104	1203	1105	2403	2722	2722	**0,0000%**	2722	2722	**0,0000%**	44.23
105	1455	1346	2896	3204	3204	**0,0000%**	3204	3204	**0,0000%**	79.76
106	4310	4074	8511	8832	8832	**0,0000%**	8832	8832	**0,0000%**	300.90
107	5596	5352	11026	11068	11068	**0,0000%**	11068	11068	**0,0000%**	301.38
108	878	802	1714	1752	1752	**0,0000%**	1752	1752	**0,0000%**	23.88
109	47573	47271	94843	—	—	—	—	—	—	—
110	111615	111149	222763	—	—	—	—	—	—	—
111	1597	1543	4138	2546098	2607044	2,3937%	2546146	2546178.3	0,0019%	120.49
112	1729	1718	4297	2657421	2716467	2,2219%	2657591	2657776.1	0,0064%	162.36
113	1846	1819	4768	3306412	3372875	2,0101%	3306699	3306757.6	0,0087%	193.41
114	752	791	27331	6707399	6721090	0,2041%	6711617	6711932.1	0,0629%	23.69
115	907	913	42176	32557048	32742525	0,5697%	32562561	32563491.2	0,0169%	30.75
116	1364	1368	2731	299975	302253	0,7594%	300012	300022.7	0,0123%	103.30
117	2025	2056	4080	708313	713125	0,6794%	708430	708454	0,0165%	235.73
118	2835	2779	5613	1241626	1249708	0,6509%	1242213	1242273.2	0,0473%	300.69
119	2326	2459	9403	14132996	14469601	2,3817%	14133544	14133845	0,0039%	294.71
120	549	574	1122	148891	151222	1,5656%	148892	148899	0,0007%	16.11
121	837	785	1621	313862	318777	1,5660%	313868	313872.8	0,0019%	22.87
122	1188	1188	3564	1920947	1943946	1,1973%	1921134	1921180.6	0,0097%	51.93
123	1654	1654	4962	3716753	3756945	1,0814%	3716912	3717017.8	0,0043%	157.84
124	1948	1948	5844	5008039	5056061	0,9589%	5008360	5008539.4	0,0064%	205.44
125	7040	7231	21417	275850	282642	2,4622%	282642	282642	2,4622%	302.33
126	9848	9856	29483	388423	397954	2,4538%	397954	397954	2,4538%	304.10
127	6938	7097	13964	105667	107886	2,1000%	107886	107886	2,1000%	302.49
128	9846	9737	19520	150765	154405	2,4144%	154405	154405	2,4144%	303.38
129	15384	2361	17744	0	0	**0,0000%**	0	0	**0,0000%**	277.11
130	18738	2891	21628	0	0	**0,0000%**	0	0	**0,0000%**	301.18
131	624	624	1248	203056	203056	**0,0000%**	203056	203056	**0,0000%**	18.55
132	766	766	1532	287525	287525	**0,0000%**	287525	287525	**0,0000%**	22.91
133	974	974	1948	463175	463175	**0,0000%**	463175	463175	**0,0000%**	31.72
134	14413	1922	14413	0	0	**0,0000%**	0	0	**0,0000%**	207.48
135	20141	2713	20141	0	0	**0,0000%**	0	0	**0,0000%**	301.19
136	1575	1530	4458	3188938	3263968	2,3528%	3188974	3189007.8	0,0011%	130.35
137	1633	1641	4955	3717792	3800569	2,2265%	3717877	3717950.9	0,0023%	164.81
138	1738	1748	12434	29471085	30186008	2,4258%	29471439	29471552.4	0,0012%	188.00
139	2424	2340	9367	955706	1047908	9,6475%	957138	957679.8	0,1498%	283.83
140	2753	2719	10923	183384	199855	8,9817%	185153	185415.7	0,9646%	300.72
141	2700	2700	10653	555664	555697	0,0059%	555697	555697	0,0059%	300.66
142	3741	3741	14791	937486	937527	0,0044%	937527	937527	0,0044%	301.15
143	4275	4275	16915	1173214	1173259	0,0038%	1173259	1173259	0,0038%	300.87
144	1024	1024	11264	25191079	25862212	2,6642%	25191179	25191286.3	0,0004%	35.58
145	32768	32768	524288	57436264952	58473205432	1,8054%	57591104410	57600771528.8	0,2696%	1348.48
146	16944	16944	35224	32548	32548	**0,0000%**	32548	32548	**0,0000%**	305.10
147	688	688	1656	1092	1092	**0,0000%**	1092	1092	**0,0000%**	19.02
148	6570	6570	13664	12612	12612	**0,0000%**	12612	12612	**0,0000%**	301.45
149	585	585	1405	931	931	**0,0000%**	931	931	**0,0000%**	16.85
150	1056	1056	2524	1696	1696	**0,0000%**	1696	1696	**0,0000%**	31.26
151	1215	1215	2899	1958	1958	**0,0000%**	1958	1958	**0,0000%**	41.39
152	2144	2180	4323	1578588	1663980	5,4094%	1578752	1578774.5	0,0104%	227.55
153	877	924	1800	282777	298923	5,7098%	282793	282799.2	0,0057%	26.71
154	1266	1331	2596	597779	632654	5,8341%	597822	597831.7	0,0072%	60.92
155	1615	1631	3245	1229491	1248743	1,5659%	1229522	1229543.9	0,0025%	125.87
156	964	857	1820	413422	420025	1,5972%	413429	413436.4	0,0017%	28.30
157	1874	1874	3746	219961	658008	199,1476%	219961	219961	**0,0000%**	203.67
158	2554	2554	5106	408321	1222408	199,3743%	408321	408321	**0,0000%**	300.58
159	2942	2942	5882	541696	1622145	199,4567%	541696	541696	**0,0000%**	300.52
160	1591	1526	3326	1480888	1515100	2,3102%	1480910	1480956	0,0015%	122.15
161	758	753	1592	328031	335196	2,1842%	328043	328048.4	0,0037%	22.10
162	958	974	2020	549129	559937	1,9682%	549142	549156.8	0,0024%	30.51
163	1061	1042	1417	242376	247292	2,0283%	242376	242381.8	**0,0000%**	39.57
164	1665	1685	2353	663371	674535	1,6829%	663390	663409.5	0,0029%	177.48
165	587	550	794	75689	77180	1,9699%	75689	75693.2	**0,0000%**	15.79
166	1138	1143	1263	158264	160038	1,1209%	158283	158295.3	0,0120%	49.08
167	1627	1642	1882	349826	354365	1,2975%	349837	349909.3	0,0031%	168.31
168	1912	1914	2083	425798	430422	1,0860%	425817	425839.6	0,0045%	206.09
169	25017	7149	25017	685363	706917	3,1449%	706917	706917	3,1449%	303.01
170	29694	8474	29694	825014	851230	3,1776%	851230	851230	3,1776%	304.27
171	10062	5553	19451	529019	545459	3,1076%	545459	545459	3,1076%	302.17
172	17125	9176	32251	889284	917046	3,1218%	917046	917046	3,1218%	304.92
173	1364	1383	4069	2437293	2497140	2,4555%	2437334	2437373.9	0,0017%	96.85
174	1682	1612	4992	3760272	3841727	2,1662%	3760344	3760413.1	0,0019%	154.74
175	1801	1845	5538	4512558	4612033	2,2044%	4512674	4512694.6	0,0026%	201.15
176	535	544	1078	130590	132775	1,6732%	130590	130601.2	**0,0000%**	15.41
177	643	635	1277	198135	201698	1,7983%	198152	198156.9	0,0086%	18.02
178	1557	1618	3174	1208102	1226981	1,5627%	1208166	1208193.4	0,0053%	161.67
179	520	510	1029	134971	136867	1,4047%	134971	134980.9	**0,0000%**	14.70
180	596	569	1164	166735	169741	1,8029%	166736	166739.2	0,0006%	16.28
181	1393	1357	2749	908632	922861	1,5660%	908670	908707.6	0,0042%	97.33
182	1449	1401	2780	1060100	1083229	2,1818%	1060117	1060122.8	0,0016%	119.71
183	1573	1503	3047	1307161	1334352	2,0802%	1307189	1307211.4	0,0021%	144.46
184	1959	1910	3838	2029443	2069925	1,9947%	2029499	2029511.1	0,0028%	210.54
185	650	621	1822	524933	537183	2,3336%	524948	524957.8	0,0029%	17.70
186	829	849	2709	1161138	1186932	2,2214%	1161150	1161162.8	0,0010%	25.83
187	904	911	2323	764103	780146	2,0996%	764134	764163	0,0041%	29.60
188	1229	1234	3881	215392	220898	2,5563%	215401	215411.1	0,0042%	59.74
189	1484	1488	5183	611805	625922	2,3074%	611854	611890.7	0,0080%	115.30
190	720	722	2455	124982	127949	2,3739%	124983	124985.4	0,0008%	20.52
191	1143	1148	3433	202793	207582	2,3615%	202807	202813.6	0,0069%	52.04
192	546	551	1890	46296	47393	2,3695%	46296	46298.1	**0,0000%**	15.04
193	771	775	2562	132222	135389	2,3952%	132226	132228.8	0,0030%	22.76
194	1292	1280	3273	1505071	1536533	2,0904%	1505093	1505103.5	0,0015%	81.05
195	1546	1559	1914	395405	399686	1,0827%	395417	395429.1	0,0030%	148.62
196	1985	1922	2238	510108	517098	1,3703%	510151	510172.9	0,0084%	225.42
197	1218	1285	3754	2483180	2530339	1,8991%	2483193	2483223.3	0,0005%	74.85
198	1632	1660	4938	4058230	4116705	1,4409%	4058269	4058391.1	0,0010%	161.98
199	7456	9940	19880	58660906	58660906	**0,0000%**	58660906	58660906	**0,0000%**	303.76
200	8371	11160	22320	73944761	73944761	**0,0000%**	73944761	73944761	**0,0000%**	303.87

A análise dos resultados demonstra que a heurística da Mediana já produz soluções iniciais de elevada qualidade. Em média, o *gap* em relação às soluções de referência foi de 8,20%, indicando que a heurística construtiva é capaz de gerar ordenações competitivas com baixo custo computacional. Em diversas instâncias, os resultados obtidos ficaram muito próximos das melhores soluções conhecidas, corroborando observações da literatura que apontam a Mediana como uma das principais heurísticas construtivas para problemas de minimização de cruzamentos (EADES; SUGIYAMA, 1990). Entretanto, a ocorrência de desvios mais expressivos em determinadas instâncias evidencia a existência de espaço para

refinamento adicional.

Na Tabela 14, os resultados destacados em negrito representam soluções que alcançaram exatamente a solução de referência (*gap* igual a 0,00%). Já os resultados sublinhados indicam soluções com *gap* inferior a 0,01%, ou seja, diferenças perceptíveis apenas a partir da terceira casa decimal. A elevada quantidade de resultados nessas duas categorias evidencia a capacidade do *Simulated Annealing* de produzir soluções altamente competitivas quando comparadas às melhores soluções conhecidas.

O comportamento observado reforça uma das principais características do SA: sua capacidade de escapar de ótimos locais por meio da aceitação probabilística de soluções temporariamente inferiores (KIRKPATRICK; GELATT; VECCHI, 1983). Esse mecanismo amplia a exploração do espaço de busca e favorece a descoberta de regiões mais promissoras, permitindo o refinamento de soluções iniciais já competitivas e produzindo resultados significativamente mais próximos das melhores soluções conhecidas.

A Figura 19 apresenta uma síntese dos resultados por meio do Desvio Médio Relativo (DMR), comparando a qualidade das soluções iniciais produzidas pela heurística da Mediana com aquelas obtidas após o refinamento pelo *Simulated Annealing*. Também é apresentado o DMR obtido ao desconsiderar a instância 9, única em que não houve melhoria em relação à solução inicial.

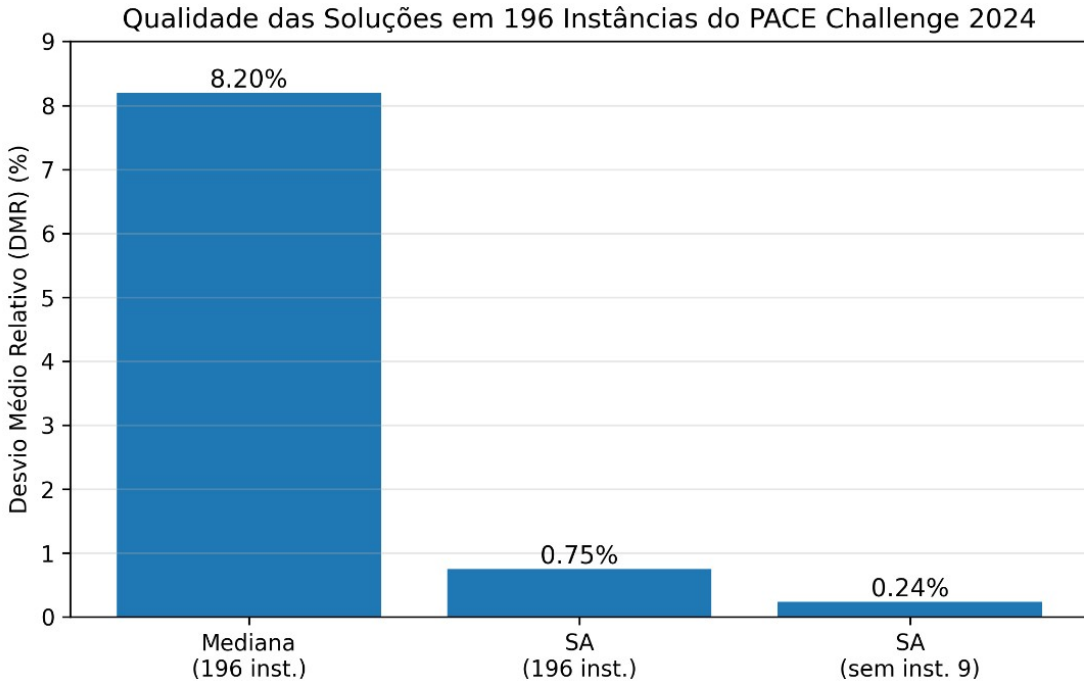

Figura 19 – DMR das 196 instâncias testadas

Fonte: Elaborado pelo autor (2026)

Conforme ilustrado na Figura 19, a aplicação do SA reduziu substancialmente esses desvios. Considerando todas as 196 instâncias avaliadas, o *gap* médio em relação às

soluções obtidas pelo *MAEDM-OCM* foi de apenas 0,75%, representando uma redução superior a 90% quando comparada ao desvio médio obtido pela heurística da Mediana. Além disso, ao desconsiderar a instância 9, na qual não houve melhora em relação à solução inicial produzida pela Mediana, o *gap* médio reduz-se para apenas 0,24%. Esses resultados demonstram a elevada capacidade de refinamento do método e sua proximidade em relação ao estado da arte para o problema.

De maneira geral, os resultados demonstram que a combinação da heurística da Mediana com o *Simulated Annealing* constitui uma abordagem eficiente para o OSCM. Considerando as 196 instâncias avaliadas, o método proposto apresentou um desvio médio inferior a 1% em relação às soluções obtidas pelo algoritmo vencedor do PACE *Challenge* 2024, evidenciando sua eficácia e robustez mesmo quando comparado a uma abordagem significativamente mais sofisticada.

6 Conclusão

Este trabalho investigou a aplicação de diferentes abordagens para o Problema de Minimização de Cruzamentos Unilaterais (*One-Sided Crossing Minimization* – OSCM), um problema clássico NP-difícil amplamente estudado no contexto do desenho automático de grafos. O estudo teve como objetivo principal avaliar o impacto de heurísticas construtivas e métodos de refinamento na qualidade das soluções obtidas, com ênfase na heurística da Mediana e na metaheurística *Simulated Annealing* (SA).

Inicialmente, foram analisadas diferentes estratégias de geração de soluções e refinamento, incluindo as heurísticas construtivas do Baricentro e da Mediana, bem como métodos de busca local baseados nas vizinhanças *Swap*, *Insert*, *2-Opt* e *Variable Neighborhood Descent* (VND), avaliados sob as estratégias *First Improvement* e *Best Improvement*. Os resultados evidenciaram que a qualidade da solução inicial exerce influência decisiva sobre o desempenho final dos algoritmos. Enquanto soluções geradas aleatoriamente ou pela heurística do Baricentro apresentaram desvios elevados em relação às referências, a heurística da Mediana produziu soluções significativamente mais próximas dos melhores resultados conhecidos, confirmando sua eficácia como método construtivo para o OSCM.

Os experimentos também demonstraram que as buscas locais foram capazes de promover melhorias consistentes sobre as soluções iniciais geradas pela Mediana. Dentre os métodos avaliados, a vizinhança *Insert* associada à estratégia *Best Improvement* apresentou o melhor desempenho entre as abordagens de descida, obtendo reduções expressivas no número de cruzamentos com custo computacional compatível.

Entretanto, o principal destaque deste trabalho foi o desempenho alcançado pelo *Simulated Annealing*. Graças ao mecanismo probabilístico de aceitação de soluções temporariamente inferiores, o método mostrou-se capaz de escapar de ótimos locais que limitaram o desempenho das buscas puramente gulosas. Após o processo de calibração de parâmetros realizado com o pacote *iRace*, o SA obteve o melhor desempenho global entre todas as abordagens avaliadas no conjunto inicial de instâncias, alcançando o menor Desvio Médio Relativo em relação às soluções de referência.

A robustez da abordagem foi posteriormente validada em um conjunto mais abrangente de experimentos utilizando o *benchmark* oficial do PACE *Challenge* 2024. Nessa etapa, foram analisadas 196 das 200 instâncias disponibilizadas pela competição, sendo as demais inviabilizadas por limitações de memória decorrentes da construção da matriz de cruzamentos utilizada na implementação. Os resultados mostraram que a heurística da Mediana apresentou um *gap* médio de 8,20% em relação às soluções produzidas pelo

algoritmo vencedor da competição, enquanto a combinação Mediana + SA reduziu esse valor para apenas 0,75%. Quando desconsiderada uma instância específica na qual não houve melhoria em relação à solução inicial, o *gap* médio foi reduzido para apenas 0,24%, evidenciando a elevada qualidade das soluções produzidas pelo método proposto.

Além da qualidade das soluções obtidas, o trabalho também destacou a importância de estruturas eficientes para avaliação incremental dos movimentos de vizinhança. A utilização de uma matriz de cruzamentos permitiu reduzir significativamente o custo computacional das buscas locais e da metaheurística, tornando viável a exploração de um grande número de soluções em tempos compatíveis com os limites estabelecidos pela competição. Dessa forma, as contribuições desta pesquisa abrangem tanto aspectos algorítmicos quanto computacionais relacionados ao tratamento do OSCM.

De maneira geral, os resultados demonstram que a combinação entre uma heurística construtiva de alta qualidade e uma metaheurística capaz de escapar de ótimos locais constitui uma estratégia altamente eficaz para o problema estudado. A heurística da Mediana fornece soluções iniciais competitivas, enquanto o *Simulated Annealing* atua como mecanismo de refinamento capaz de explorar regiões promissoras do espaço de busca e produzir soluções muito próximas ao estado da arte, mesmo empregando uma estratégia conceitualmente mais simples do que aquelas utilizadas pelos melhores métodos da literatura recente.

Como trabalhos futuros, destacam-se algumas direções promissoras. A primeira consiste no desenvolvimento de estratégias capazes de operar com apenas parte da matriz de cruzamentos em memória, permitindo o tratamento de instâncias ainda maiores sem comprometer a eficiência computacional. Também se mostra relevante a investigação de metaheurísticas mais avançadas, como *Greedy Randomized Adaptive Search Procedure* (GRASP), *Iterated Local Search* (ILS) e algoritmos meméticos, bem como o estudo de mecanismos adaptativos de controle de temperatura, reaquecimento e resfriamento no *Simulated Annealing*. Além disso, a exploração de paralelismo computacional e a aplicação das abordagens desenvolvidas a variantes mais gerais do problema de minimização de cruzamentos constituem oportunidades interessantes para a continuidade desta linha de pesquisa.

Em síntese, conclui-se que a heurística da Mediana constitui uma excelente estratégia construtiva para o OSCM e que o *Simulated Annealing* é capaz de potencializar significativamente essa solução inicial. Os resultados obtidos demonstram que a abordagem proposta produz soluções de alta qualidade, com desvios muito reduzidos em relação às melhores soluções conhecidas, confirmando sua eficácia, robustez e potencial para aplicações futuras no contexto da otimização combinatória e do desenho de grafos.

Referências

AARTS, Emile H. L.; LENSTRA, Jan Karel (ed.). *Local Search in Combinatorial Optimization*. 2. ed. Princeton, NJ: Princeton University Press, 2003. Citado 2 vezes na página 15.

ARROYO, José Elias Claudio. *Heurísticas e Metaheurísticas para Otimização Combinatória Multiobjetivo*. 2002. Tese (Doutorado) – Universidade Estadual de Campinas, Campinas, SP. Citado 1 vez na página 24.

BALIEIRO FILHO, Inocêncio Fernandes. *Arquimedes, Pappus, Descartes e Polya: Quatro Episódios da História da Heurística*. São Paulo: Editora UNESP, 2017. ISBN 9788595461765. Disponível em: https://books.scielo.org/id/mvxd6/pdf/balieiro-9788595461765.pdf. Citado 1 vez na página 23.

BELAN, Peterson Adriano. *Métodos Heurísticos e Metaheurísticos: Aplicações em Problemas de Otimização Combinatória*. 2024. Material didático do Programa de Pós-Graduação em Informática (PPGI). Disponível em: https://ppgi.belan.pro.br/metodos/MHM_2024_Aula_01.pdf. Citado 0 vez na página 25.

BELFIORE, Patrícia; FÁVERO, Luiz Paulo. *Pesquisa Operacional para Cursos de Engenharia*. Rio de Janeiro: LTC, 2013. Citado 3 vezes nas páginas 20–23.

BERTSIMAS, Dimitris; DUNN, Jack. *Machine Learning Under a Modern Optimization Lens*. Belmont: Dynamic Ideas LLC, 2019. Citado 1 vez na página 14.

BLUM, Christian; ROLI, Andrea. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268–308, 2003. DOI: 10.1145/937503.937505. Citado 4 vezes nas páginas 24, 25, 38, 49.

BOEHMER, Kimon *et al.* Arcee: An OCM-Solver. *arXiv preprint*, arXiv:2411.17596, 2024. Disponível em: https://arxiv.org/abs/2411.17596. Citado 1 vez na página 42.

BONDY, J. A.; MURTY, U. S. R. *Graph Theory*. Springer, 2008. Citado 1 vez na página 26.

BUENO, Fabrício. *Métodos Heurísticos: Teoria e Implementações*. Araranguá, SC, 2009. Tutorial. Disponível em: https://www.academia.edu/33490463/M%C3%A9todos_Heur%C3%ADsticos_Teoria_e_Implementa%C3%A7%C3%B5es. Citado 1 vez na página 25.

CEDER, Avishai. *Public Transit Planning and Operation: Modeling, Practice and Behavior*. 2. ed. Boca Raton: CRC Press, 2016. Citado 1 vez na página 14.

CHAVES, Viviane Hengler Corrêa. *Perspectivas Históricas da Pesquisa Operacional*. 2011. Dissertação (Mestrado em Educação Matemática) – Universidade Estadual Paulista (UNESP),

Rio Claro, SP. Disponível em: https://repositorio.unesp.br/server/api/core/bitstreams/e5882288-94c5-4f44-963d-4c9f19c83b63/content. Citado 1 vez na página 22.

CYGAN, Marek *et al. Parameterized Algorithms*. Springer, 2015. ISBN 9783319212753. DOI: 10.1007/978-3-319-21275-3. Citado 1 vez na página 32.

DANTZIG, George B. *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press, 1963. Disponível em: https://www.rand.org/content/dam/rand/pubs/reports/2007/R366part1.pdf. Citado 1 vez na página 14.

DI BATTISTA, Giuseppe *et al. Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ: Prentice Hall, 1999. Disponível em: https://dl.acm.org/doi/book/10.5555/551884. Citado 6 vezes nas páginas 15, 28, 30, 33, 34, 38.

DIESTEL, Reinhard. *Graph Theory*. 5. ed.: Springer, 2017. Citado 2 vezes nas páginas 26, 28.

DWORK, Cynthia *et al.* Rank Aggregation Methods for the Web. *Proceedings of the 10th International Conference on World Wide Web (WWW)*, 2001. Citado 2 vezes nas páginas 32, 34.

EADES, Peter; SUGIYAMA, Kozo. How to Draw a Directed Graph. *Journal of Information Processing*, v. 13, n. 4, p. 424–437, 1990. Citado 8 vezes nas páginas 15, 29, 33, 34, 37, 43, 90.

EULER, Leonhard. Solutio Problematis ad Geometriam Situs Pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, v. 8, p. 128–140, 1736. Citado 1 vez na página 26.

FABOZZI, Frank J. *et al. Robust Portfolio Optimization and Management*. Hoboken, NJ: John Wiley & Sons, 2007. Citado 1 vez na página 14.

FEO, Thomas A.; RESENDE, Mauricio G. C. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, v. 6, n. 2, p. 109–133, 1995. DOI: 10.1007/BF01096763. Citado 1 vez na página 44.

FERREIRA, Aurélio Buarque de Holanda. *Dicionário Aurélio da Língua Portuguesa*. Rio de Janeiro: Nova Fronteira, 1997. Citado 1 vez na página 23.

FREY, Johannes *et al.* Detailed Visualization of Communication Network Topologies. *IFAC-PapersOnLine*, v. 50, n. 1, p. 1199–1204, 2017. DOI: 10.1016/j.ifacol.2017.08.342. Citado 1 vez na página 36.

GAREY, Michael R.; JOHNSON, David S. Crossing Number is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods*, v. 4, n. 3, p. 312–316, 1983. DOI: 10.1137/0604033. Citado 2 vezes nas páginas 31, 32.

GATTI JUNIOR, Wilian. *Pesquisa Operacional*. São Paulo, SP: Editora Senac São Paulo, 2020. Disponível em: https://books.google.com/books?id=-3LgDwAAQBAJ. Citado 0 vez na página 21.

GLOVER, Fred W.; KOCHENBERGER, Gary A. (ed.). *Handbook of Metaheuristics*. Boston, MA: Springer, 2003. v. 57. (International Series in Operations Research and Management Science). ISBN 9781402072635. Disponível em: https://books.google.com/books?id=xmhoG772iuQC. Citado 1 vez na página 24.

GUTOWSKI, Grzegorz *et al.* *One-Sided Local Crossing Minimization*. 2025. arXiv: 2510.00331 [cs.DS]. Disponível em: https://arxiv.org/abs/2510.00331. Citado 1 vez na página 47.

HANSEN, Pierre *et al.* Variable Neighborhood Search. *In:* HANDBOOK of Metaheuristics. Springer, 2018. p. 57–97. Citado 1 vez na página 63.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. *Introdução à Pesquisa Operacional*. 9. ed. São Paulo: McGraw-Hill, 2013. Disponível em: https://books.google.com/books?id=-A88a0-KxQ0C. Citado 6 vezes nas páginas 14, 18, 19, 21, 23.

JÜNGER, Michael; MUTZEL, Petra. 2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms. *In:* GRAPH Algorithms and Applications I. World Scientific, 2002. p. 3–27. Citado 2 vezes nas páginas 15, 44.

KINDERMANN, Philipp; KLUTE, Fabian; TERZIADIS, Soeren. The PACE 2024 Parameterized Algorithms and Computational Experiments Challenge: One-Sided Crossing Minimization. *In:* BONNET, Édouard; RZĄŻEWSKI, Paweł (ed.). *Proceedings of the 19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. v. 321. (Leibniz International Proceedings in Informatics (LIPIcs)), 26:1–26:20. DOI: 10.4230/LIPIcs.IPEC.2024.26. Citado 10 vezes nas páginas 16, 33, 37–40, 72, 89.

KIRKPATRICK, Scott; GELATT, C. Daniel; VECCHI, Mario P. Optimization by Simulated Annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983. DOI: 10.1126/science.220.4598.671. Citado 4 vezes nas páginas 67, 91.

LAMPROPOULOS, Georgios; SIAKAS, Konstantinos. Enhancing and Securing Cyber-Physical Systems and Industry 4.0 through Digital Twins: A Critical Review. *Journal of Software: Evolution and Process*, v. 35, n. 7, e2494, 2023. DOI: 10.1002/smr.2494. Citado 1 vez na página 34.

LENGAUER, Thomas. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, 1990. Citado 2 vezes nas páginas 15, 29.

LIMA, Edirlei Soares de. *Busca em Profundidade e Busca em Largura*. 2017. Projeto e Análise de Algoritmos – Aula 11. Disponível em: https://edirlei.com/aulas/paa_2017_1/PAA_Aula_11_Busca_Grafos_2017.html. Citado 0 vez na página 27.

LIMA, Gizelle Cristina Guisso de. *Emparelhamento em Grafos Bipartidos*. 2017. Dissertação (Mestrado) – Programa de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional, Maringá, PR. Disponível em: http://repositorio.uem.br:8080/jspui/bitstream/1/5545/1/000225973.pdf. Citado 1 vez na página 27.

LISBOA, Erico Fagundes Anicet. *Apostila do Curso: Pesquisa Operacional*. Fev. 2002. Versão digital. M.Sc. Disponível em: http://www.ericolisboa.eng.br. Citado 1 vez na página 20.

LÓPEZ-IBÁÑEZ, Manuel *et al.* The irace Package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives*, v. 3, p. 43–58, 2016. DOI: 10.1016/j.orp.2016.09.002. Citado 3 vezes na página 71.

MAGALHÃES, Roberto. *Quais são os diferentes tipos de circuitos integrados?* Jun. 2024. Disponível em: https://compraco.com.br/zh-hant/blogs/tecnologia-e-desenvolvimento/quais-sao-os-diferentes-tipos-de-circuitos-integrados. Citado 0 vez na página 36.

MARINS, Fernando Augusto Silva. *Introdução à Pesquisa Operacional*. Guaratinguetá, SP, 2009. Notas de aula. Departamento de Produção. Disponível em: https://www.researchgate.net/publication/267271949_Introducao_a_Pesquisa_Operacional. Citado 2 vezes nas páginas 20, 22.

MORSE, Philip McCord; KIMBALL, George E. *Methods of Operations Research*. Reprint: Courier Corporation, 2003. p. 158. Disponível em: https://books.google.com/books?id=ChvP-vdjl8wC. Citado 1 vez na página 18.

MUÑOZ, Xavier; UNGER, Walter; VRT'O, Imrich. One-Sided Crossing Minimization is NP-Hard for Sparse Graphs. *In:* PROCEEDINGS of the 9th International Symposium on Graph Drawing (GD 2001). Springer, 2001. v. 2265. (Lecture Notes in Computer Science), p. 115–123. Citado 2 vezes nas páginas 32, 38.

NICOLETTI, Maria do C. *Fundamentos da Teoria dos Grafos para Computação*. 3. ed. Rio de Janeiro: LTC, 2018. E-book, p. 76. ISBN 9788521634775. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788521634775/. Citado 1 vez na página 26.

PACE CHALLENGE. *PACE 2024: One-Sided Crossing Minimization*. 2024. Disponível em: https://pacechallenge.org/2024/. Citado 0 vez na página 30.

PETROPOULOS, Fotios *et al.* Operational Research: Methods and Applications. *Journal of the Operational Research Society*, Taylor & Francis, v. 75, n. 3, p. 423–617, 2024. DOI: 10.1080/01605682.2023.2253852. Citado 1 vez na página 14.

PINEDO, Michael L. *Scheduling: Theory, Algorithms, and Systems*. 5. ed. Cham: Springer, 2016. Citado 1 vez na página 14.

PURCHASE, Helen C. Metrics for Graph Drawing Aesthetics. *Journal of Visual Languages & Computing*, v. 13, n. 5, p. 501–516, 2002. DOI: 10.1006/jvlc.2002.0232. Citado 3 vezes nas páginas 28, 29, 33.

RABENSCHLAG, Denis Rasquin. *Pesquisa Operacional*. Santa Maria, RS, 2005. Centro de Tecnologia, Departamento de Produção e Sistemas. Disponível em: https://www.researchgate.net/publication/296325826_Pesquisa_Operacional. Citado 4 vezes nas páginas 19, 22, 23.

SEGURA, Carlos *et al.* PACE Solver Description: CIMAT Team. *In:* BONNET, Édouard; RZĄŻEWSKI, Paweł (ed.). *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. v. 321. (Leibniz International Proceedings in Informatics (LIPIcs)), 31:1–31:4. DOI: 10.4230/LIPIcs.IPEC.2024.31. Citado 3 vezes nas páginas 33, 38, 89.

SIMCHI-LEVI, David; KAMINSKY, Philip; SIMCHI-LEVI, Edith. *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*. 3. ed. New York: McGraw-Hill, 2008. Citado 1 vez na página 14.

SPÖNEMANN, M.; SCHULZE, C. D.; HANXLEDEN, R. von. Drawing Layered Graphs with Port Constraints. *Journal of Visual Languages and Computing*, v. 25, n. 2, p. 89–106, 2014. DOI: 10.1016/j.jvlc.2013.11.005. Citado 1 vez na página 37.

SUCUPIRA, Igor Ribeiro. *Métodos Heurísticos Genéricos: Meta-Heurísticas e Hiper-Heurísticas*. 2004. Monografia – Universidade de São Paulo, São Paulo, SP. Citado 1 vez na página 24.

SUGIYAMA, Kozo; TAGAWA, Shojiro; TODA, Mitsuhiko. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 11, n. 2, p. 109–125, 1981. DOI: 10.1109/TSMC.1981.4308636. Citado 4 vezes nas páginas 29, 34, 37, 43.

TAHA, Hamdy A. *Operations Research: An Introduction*. 10. ed. Harlow, England: Pearson Education Limited, 2017. Disponível em: http://zalamsyah.staff.unja.ac.id/wp-content/uploads/sites/286/2019/11/9-Operations-Research-An-Introduction-10th-Ed.-Hamdy-A-Taha.pdf. Citado 9 vezes nas páginas 14, 18, 19, 22.

TAHA, Hamdy A. *Pesquisa Operacional*. 8. ed. São Paulo: Pearson Education, 2008. Citado 1 vez na página 14.

TALBI, El-Ghazali. *Metaheuristics: From Design to Implementation*. Hoboken, NJ: John Wiley & Sons, 2009. ISBN 9780470278581. Disponível em: https://books.google.com/books?id=SIsa6zi5XV8C. Citado 3 vezes nas páginas 24, 26, 43.

TAMASSIA, Roberto (ed.). *Handbook of Graph Drawing and Visualization*. Boca Raton, FL: CRC Press, 2013. (Discrete Mathematics and Its Applications). ISBN 9780849373640. Disponível em: https://books.google.com/books?id=eEDMBQAAQBAJ. Citado 2 vezes nas páginas 29, 33.

Anexos

ANEXO A – Declaração de Uso de Inteligência Artificial Generativa

Durante a preparação deste documento, eu, *Harrison de Lana Araújo e Silva*, estudante de graduação do curso de *Engenharia de Controle e Automação*, declaro o uso de IAGen *ChatGPT (OpenAI)*, versão *GPT-5.5*, como ferramenta de apoio à escrita científica. A ferramenta foi utilizada como recurso auxiliar na revisão gramatical, ortográfica e estilística dos textos, no aprimoramento da redação acadêmica, na organização e estruturação do documento, na adaptação de figuras e fluxogramas para fins ilustrativos e na revisão da linguagem empregada ao longo dos capítulos deste trabalho.

Após o uso desta ferramenta, a pessoa autora revisou e editou integralmente o conteúdo em conformidade com os princípios éticos para uso de IAGen (Resolução CONPEP nº 143) e com os acordos estabelecidos com a pessoa orientadora da pesquisa. Dessa forma, assumo total responsabilidade pelo conteúdo desta publicação.

Ressalta-se que a ferramenta foi utilizada exclusivamente como apoio ao processo de escrita, revisão, organização e ilustração do trabalho, não sendo empregada para a geração automática dos resultados da pesquisa, da implementação dos algoritmos, dos experimentos computacionais, das análises realizadas ou das conclusões apresentadas, as quais são de autoria e responsabilidade exclusiva do autor.