UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE EDUCAÇÃO SUPERIOR DO ALTO VALE DO ITAJAÍ – CEAVI
ENGENHARIA DE SOFTWARE

GABRIEL ANTONIO BERTOLDI PERINI

HEURÍSTICAS CONSTRUTIVAS E ADAPTATIVAS PARA O PROBLEMA DO CONJUNTO *K*-DOMINANTE MÍNIMO

IBIRAMA – SC

2025

GABRIEL ANTONIO BERTOLDI PERINI

HEURÍSTICAS CONSTRUTIVAS E ADAPTATIVAS PARA O PROBLEMA DO CONJUNTO *K*-DOMINANTE MÍNIMO

Trabalho de conclusão apresentado ao curso de Engenharia de Software do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), da Universidade do Estado de Santa Catarina (UDESC), como requisito parcial para a obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Marcelo de Souza

IBIRAMA – SC

2025

GABRIEL ANTONIO BERTOLDI PERINI

HEURÍSTICAS CONSTRUTIVAS E ADAPTATIVAS PARA O PROBLEMA DO CONJUNTO *K*-DOMINANTE MÍNIMO

Trabalho de conclusão apresentado ao curso de Engenharia de Software do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), da Universidade do Estado de Santa Catarina (UDESC), como requisito parcial para a obtenção do grau de bacharel em Engenharia de Software.

Banca Examinadora

Orientador:

Prof. Dr. Marcelo de Souza
UDESC

Membros:

Prof. Dr. Paulo Roberto Farah
UDESC

Prof. Tiago Funk
UDESC

Ibirama – SC, 01/07/2025

Dedico este trabalho aos meus queridos amigos e especialmente à minha irmã e meu pai, que genuinamente me fazem acreditar que sou a pessoa mais abençoada desse mundo pelo privilégio de tê-los ao meu lado.

AGRADECIMENTOS

Agradeço primeiramente aos amigos que encontrei nestes anos de curso, que me proporcionaram momentos muito divertidos, e de forma mais significativa àqueles amigos mais próximos de longa data, que são como irmãos para mim.

Um obrigado também à universidade e aos professores que me instruíram nessa jornada.

Gostaria expressar um agradecimento de um modo mais especial, aos meus professores Prof. Dra. Marília Guterres Ferreira, Prof. MSc. Mattheus da Hora França, que além de ótimos professores e profissionais, são pessoas incríveis e foram grandes parceiros durante toda a jornada. De modo similar gostaria de agradecer ao Glauco David Laicht, que foi um verdadeiro mentor para mim dentro do EMDPS, sempre muito compreensivo, me incentivando também a melhorar como profissional. Tive sorte de ter a oportunidade de trabalhar com todos eles nessa jornada.

Agradeço também ao meu orientador, Prof. Dr. Marcelo de Souza, com quem foi um prazer desenvolver este trabalho, tanto por sua paixão pelo tema quanto pelo seu cuidado e dedicação. Também, pela paciência de me atender nos horários mais inoportunos, sempre atencioso às diversas ideais que discutimos.

Por último e mais importante, minha mais sincera gratidão à meu pai e minha irmã, que são pessoas maravilhosas, sempre me apoiando e me incentivando em todas as etapas da vida. Sem eles nada disso teria sido possível, muito obrigado!

“Não importa quão devagar você vá, desde que você não pare.”
— Confúcio, filósofo chinês

RESUMO

Este trabalho propõe novas abordagens metaheurísticas para o o problema do conjunto k-dominante mínimo (MkDSP), uma generalização do clássico problema do conjunto dominante mínimo (MDSP) em grafos. Dado um grafo $G = (V, E)$ e um valor $k \in \mathbb{N}$, o (MkDSP) consiste em encontrar um conjunto mínimo $S \subseteq V$ onde cada vértices $v \in V$ pertence a S ou possui ao menos k vizinhos em S. Tratando-se de um problema NP-Difícil, encontrar soluções ótimas para instâncias de médio e grande porte demandam recursos computacionais consideráveis. Sendo assim, este trabalho propõe as metaheurísticas ALNS e LNS assim como um conjunto de operadores que serão combinados aos algoritmos citados para resolver o MkDSP. Além disso, é apresentado uma série de estudos comparativos, entre os métodos propostos neste trabalho e finalmente com o estado da arte da literatura.

Palavras-chave: *Adaptive Large Neighborhood Search*; Problema do Conjunto k-Dominante Mínimo; Otimização Combinatória; Teoria dos Grafos

ABSTRACT

This work proposes new metaheuristic approaches to solve the Minimum k-Dominating Set Problem (MkDSP), a generalization of the classical Minimum Dominating Set Problem (MDSP) in graphs. Given an undirected, unweighted graph $G = (V, E)$ and an integer $k \in \mathbb{N}$, the MkDSP consists in finding a minimum subset $S \subseteq V$ such that every vertex $v \in V$ either belongs to S or has at least k neighbors in S. As an *NP*-hard problem, finding optimal solutions for medium and large instances requires significant computational effort. herefore, this work proposes the metaheuristics *Large Neighborhood Search* (LNS) and *Adaptive Large Neighborhood Search* (ALNS), as well as a set of operators that will be combined with the aforementioned algorithms to solve the MkDSP. Additionally, a comprehensive set of experimental studies is conducted, both to compare the proposed methods and to evaluate their performance against the state-of-the-art solutions available in the literature.

Keywords: Adaptive Large Neighborhood Search; Minimum k-Dominating Set Problem; Combinatorial Optimization; Graph Theory

LISTA DE ILUSTRAÇÕES

Figura 1 – Possíveis conjuntos dominantes S_1 e S_2 para $G = (V,E)$ 18
Figura 2 – *conjunto 2-dominante mínimo S_3 = {a, c, e, g, i}.* 19
Figura 3 – Fluxo do procedimento *Iterated Racing* 21
Figura 4 – Aplicação da heurística H_V ao grafo G_1. 32
Figura 5 – Aplicação da heurística H_D ao grafo G_1. 33
Figura 6 – Aplicação da heurística $H\varphi_D$ ao grafo G_3. 35
Figura 7 – Aplicação da heurística $H\varphi_D$ ao grafo G_4. 37
Figura 8 – Aplicação da heurística $H\varphi_V$ ao grafo G_4. 37
Figura 9 – Progressão dos operadores de reconstrução para a instância *Bristol* considerando três métricas (número de chamadas, score acumulado e peso) com a configuração $Conf_K$ para diferentes valores de $k = 1,2,4$ 74
Figura 10 – Progressão dos operadores de reconstrução para a instância *Bristol* considerando três métricas (número de chamadas, score acumulado e peso) com as configurações $Conf_{K1}$, $Conf_{K2}$ e $Conf_{K4}$ para os respectivos valores de $k = 1,2,4$ 75

LISTA DE TABELAS

Tabela 1 – Métodos para resolver o problema de k-dominação com base em cinco características. 23
Tabela 2 – Nomenclatura dos símbolos usados no Algoritmo 3 43
Tabela 3 – Número de Vértices e Arestas para 20 Instâncias Selecionadas 46
Tabela 4 – Parâmetros configuráveis e seus respectivos domínios. 48
Tabela 5 – Configurações de parâmetros distintas aplicadas ao ALNS 49
Tabela 6 – Resultados das heurísticas construtivas determinísticas para $k = 1$. 50
Tabela 7 – Resultados das heurísticas construtivas determinísticas para $k = 2$. 51
Tabela 8 – Resultados das heurísticas construtivas determinísticas para $k = 4$. 52
Tabela 9 – Comparação entre diferentes operadores de reparação na meta-heurística LNS em $k = 1$. 54
Tabela 10 – Comparação entre diferentes operadores de reparação na meta-heurística LNS em $k = 2$. 55
Tabela 11 – Comparação entre diferentes operadores de reparação na meta-heurística LNS em $k = 4$. 56
Tabela 12 – Comparação entre ALNS, LNS (melhores resultados) e heurísticas construtivas (melhores resultados) para $k = 1$. 58
Tabela 13 – Comparação entre ALNS, LNS (melhores resultados) e heurísticas construtivas (melhores resultados) para $k = 2$. 59
Tabela 14 – Comparação entre ALNS, LNS (melhores resultados) e heurísticas construtivas (melhores resultados) para $k = 4$. 60
Tabela 15 – Resultados de ALNS com diferentes configurações em $k = 1$. 62
Tabela 16 – Resultados de ALNS com diferentes configurações em $k = 2$. 64
Tabela 17 – Resultados de ALNS com diferentes configurações em $k = 4$. 65
Tabela 18 – Comparação dos resultados VNS e ALNS para $k = 1$. 67
Tabela 19 – Comparação dos resultados VNS e ALNS para $k = 2$. 68
Tabela 20 – Comparação dos resultados VNS e ALNS para $k = 4$. 69

SUMÁRIO

1 INTRODUÇÃO **12**

1.1 Problema . . . 13

1.2 Objetivos . . . 13

1.2.1 Objetivo Geral . . . 14

1.2.2 Objetivos Específicos . . . 14

1.3 Justificativa . . . 14

1.4 Metodologia . . . 15

1.5 Estrutura do trabalho . . . 16

2 FUNDAMENTAÇÃO TEÓRICA **18**

2.1 Problema do Conjunto k-Dominante Mínimo - M*k*DSP . . . 18

2.2 Configuração de Parâmetros . . . 20

3 TRABALHOS CORRELATOS **22**

3.1 Análise da Literatura . . . 22

3.2 *Iterated Greedy* para o MDSP . . . 24

3.3 Metaheurísticas aplicadas ao M*k*DSP . . . 25

3.3.1 *Beam Search* com Construção Gulosa . . . 25

3.3.2 *Variable Neighborhood Search* . . . 26

4 ABORDAGEM PROPOSTA **28**

4.1 Heurísticas Construtivas Gulosas para o M*k*DSP . . . 28

4.1.1 Heurística Baseada em Vizinhos Remanescentes . . . 30

4.1.2 Heurística Baseada no Grau de Dominação . . . 32

4.1.3 Heurísticas Híbridas . . . 34

4.1.4 Heurística Aleatória . . . 38

4.2 Large Neighborhood Search . . . 38

4.2.1 Técnicas Estocásticas dedicadas aos Operadores de Reconstrução . . . 39

4.2.2 Critério de Aceitação da Solução . . . 40

4.3 Adaptative Large Neighborhood Search . . . 41

5 AVALIAÇÃO EXPERIMENTAL **45**

5.1 Instâncias . . . 46

5.2 Configuração de Parâmetros . . . 47

5.3 Resultados e Discussão 49
5.3.1 Avaliação das Heurísticas Gulosas Determinísticas 49
5.3.2 Avaliação do LNS 53
5.3.3 Avaliação do ALNS vs. LNS vs. Heurísticas 57
5.3.4 Avaliação do ALNS sob Diferentes Configurações 61
5.3.5 Comparação com o Estado da Arte 66
5.3.6 Análise Detalhada do ALNS 70

6 CONSIDERAÇÕES FINAIS **76**
6.1 Trabalhos Futuros 77

REFERÊNCIAS **79**

1 INTRODUÇÃO

Grafos são estruturas matemáticas usadas para modelar relações entre pares de objetos. Eles são capazes de representar uma variedade de problemas e cenários do mundo real. Formalmente, um grafo é definido como um par $G = (V, E)$, onde V é o conjunto de vértices e E é o conjunto de arestas que conectam esses vértices West et al. (2001).

Na teoria dos grafos, campo de estudo dessa estrutura, um dos problemas mais difundidos é o problema da dominação (também chamado de conjunto dominante). Sua relevância se confere tanto pela sua aplicabilidade em cenários do mundo real, quanto pela sua importância em campos de estudo teóricos. A dominação em grafos consiste em selecionar um subconjunto de vértices de maneira que todos os vértices do grafo estejam no subconjunto selecionado ou sejam adjacentes (conectados) a pelo menos um vértice daquele subconjunto. O desafio de encontrar o menor subconjunto que domine um grafo é conhecido como o problema do conjunto dominante mínimo (*minimun dominating set problem*, MDSP).

Este trabalho tem como foco de estudo uma generalização do MDSP, o problema do conjunto k-dominante mínimo (*minimun k-domination problem*, MkDSP). Essa variação se difere do problema clássico introduzindo uma nova entrada $k \in \mathbb{N}$. Um conjunto $D \subseteq V$ de um grafo é considerado k-dominante se todo vértice $v \in V$ está contido em D ou é vizinho de pelo menos k vértices de D. Sua versão de decisão, de maneira similar ao problema tradicional, acrescenta outro inteiro l como entrada, a fim de verificar se há um conjunto k-dominante de tamanho l para o grafo. Lan e Chang (2013) demonstram que essa versão de decisão do Problema da k-Dominação é NP-Completo, até mesmo para grafos divididos (*split graphs*). Já a versão de otimização, do mesmo modo que a variante clássica, é classificada como um problema NP-Difícil (FINK, 1985).

Diversas abordagens podem ser aplicadas para resolver problemas dessa complexidade. Métodos exatos, isto é, algoritmos determinísticos que produzem a solução ótima, podem ser custosos e demandar tempo de execução inviável para instâncias de maior tamanho. Em contrapartida, algoritmos heurísticos e metaheurísticas se apresentam como alternativas interessantes para encontrar boas soluções em tempo razoável, sacrificando a otimalidade do conjunto encontrado em troca do menor tempo de processamento.

Métodos heurísticos podem conter uma abundante quantidade de mecanismos, das mais variadas estratégias e procedimentos, mas que ainda assim visam encontrar uma solução aceitável. Esses procedimentos podem ser classificados em duas categorias. Heurísticas, que são algoritmos projetados com um problema específico em mente, fazendo uso de suas particularidades no desenvolvimento de sua estratégia. Metaheurísticas, por sua vez, são estratégias

abstratas, de alto nível, que podem ser aplicadas a qualquer problema e guiam a busca dentro de um complexo espaço de soluções. Esses algoritmos também são classificados quanto ao seu funcionamento geral. Abordagens de construção produzem soluções pela seleção e inclusão de elementos a partir de uma solução vazia. Abordagens de busca (ou perturbativas) exploram o espaço de soluções aplicando modificações em uma solução de base (ZÄPFEL; BRAUNE; BÖGL, 2010).

Este trabalho propõe uma metaheurística ainda não aplicada ao M*k*DSP, com o objetivo de encontrar boas soluções em baixo tempo de processamento. O método adotado é o ALNS (*Adaptive Large Neighbourhood Search*) complementado por um conjunto de heurísticas construtivas, fundamentais para seu funcionamento. O trabalho apresenta um estudo experimental dos algoritmos propostos, comparando o desempenho individual das heurísticas desenvolvidas e dos demais operadores que compõem o ALNS.

1.1 PROBLEMA

A literatura apresenta vários estudos dedicados a problemas de dominação e suas variantes, visto que dominação é um conceito fundamental dentro da teoria dos grafos. Consequentemente, existe uma vasta gama de métodos empregados ao MDSP. Ainda que o M*k*DSP disponha de algumas técnicas já propostas, o volume de estudos dedicados a ele é consideravelmente menor do que aquele voltado ao problema clássico.

No que diz respeito à diversidade das abordagens empregadas na busca por conjuntos *k*-dominantes, observa-se que a literatura se concentra em heurísticas construtivas e métodos exatos. Metaheurísticas, por sua vez, são aplicadas quase que exclusivamente ao MDSP (CORCORAN; GAGARIN, 2021). No contexto do M*k*DSP, essas técnicas ainda são pouco exploradas, com os trabalhos de Corcoran e Gagarin (2021) e (PREDOJEVIĆ; KARTELJ; DJUKANOVIĆ, 2023) sendo as únicas abordagens a empregarem metaheurísticas voltadas ao problema. Diante deste cenário, fica evidente uma carência na aplicação de metaheurísticas ao M*k*DSP. A aplicação desses algoritmos, além de colaborar com a variabilidade de técnicas presentes na literatura, tem o potencial de contribuir melhorando os resultados do estado da arte do M*k*DSP.

1.2 OBJETIVOS

Esta seção indica os objetivos geral e específicos do trabalho. São definidas também as contribuições buscadas com o estudo.

1.2.1 Objetivo Geral

Este trabalho propõe a implementação da metaheurística ALNS (*Adaptive Large Neighborhood Search*) para aproximar o conjunto k-dominante mínimo em instâncias de *benchmark* presentes na literatura. ALNS é uma metaheurística que adiciona um mecanismo adaptativo ao LNS (*Large Neighborhood Search*), cuja ideia central consiste em iterativamente destruir e reconstruir partes de uma solução. Nesse contexto, o desenvolvimento da metaheurística também prevê o projeto e construção dos componentes algorítmicos do ALNS, em especial as heurísticas construtivas. São incluídas as principais abordagens da literatura, bem como projetadas novas heurísticas.

1.2.2 Objetivos Específicos

I. Desenvolver a metaheurística ALNS, de modo que seja capaz de resolver o MkDSP para as instâncias designadas.

II. Propor novas abordagens de heurísticas construtivas, utilizadas como componentes internos no ALNS.

III. Realizar um estudo comparativo entre os componentes de construção e destruição do ALNS, a fim de identificar combinações com melhor desempenho.

IV. Estudar o ganho de desempenho do ALNS ao ajustar os valores dos seus parâmetros.

1.3 JUSTIFICATIVA

O estudo acerca do problema apresentado desperta interesse por dois motivos principais. O primeiro se refere aos desafios computacionais que envolvem alcançar um conjunto k-dominante mínimo, já que se trata de um problema *NP*-difícil, tornando necessário o desenvolvimento de algoritmos eficientes para aproximar o resultado ótimo. O segundo motivo é ligado à possibilidade de aplicação prática, particularmente para redes de larga escala, onde diversos cenários do mundo real podem ser modelados a partir dessa abordagem.

No contexto da complexidade computacional, ao afirmar que o MkDSP pertence à classe *NP*-difícil, compreende-se que não existem métodos capazes de resolvê-lo de forma eficiente, nem mesmo de verificar a otimalidade de uma solução em tempo razoável (ERICKSON, 2023), assumindo que $\mathcal{P} \neq \mathcal{NP}$. Determinar o conjunto k-dominante mínimo é, portanto, uma tarefa computacionalmente custosa. Um grafo $G = (V, E)$ qualquer possui um número finito de conjuntos dominantes, no total de $2^n - 1$. Para encontrar o conjunto dominante mínimo de G,

bastaria enumerar todo subconjunto $S \subseteq V$, sendo S um conjunto dominante, ordenando-os do maior ao menor com base na cardinalidade de S, selecionando o primeiro conjunto da lista. Apesar de sua simplicidade, este algoritmo possui uma complexidade de tempo exponencial ($\mathcal{O}(2^n)$ passos), o que leva a tempos de processamento impraticáveis para encontrar a solução ótima em grafos com uma quantidade significativa de vértices. Nesse sentido, a inerente complexidade de problemas dessa classe demanda algoritmos eficientes que produzam soluções satisfatórias em tempo viável. Além disso, os métodos apresentados neste trabalho podem ser proveitosos para problemas de complexidade similar, seja promovendo abordagens eficazes ou aumentando o entendimento geral acerca do problema.

Além dos desafios teóricos e computacionais, problemas de dominação também apresentam ampla aplicabilidade prática. Diversos sistemas do mundo real podem ser modelados como grafos, nos quais um conjunto dominante representa elementos com maior capacidade de monitorar, influenciar ou cobrir a rede. O problema clássico de dominação (MDSP, com $k = 1$) foi explorado por Nacher e Akutsu (2016), que o aplicaram à modelagem de redes biológicas com o objetivo de identificar os vértices com maior capacidade de controle sobre a rede, que apresenta implicações relevantes em terapias e diagnósticos. O MkDSP, por sua vez, também tem aplicações práticas. Gagarin e Corcoran (2018) o utilizaram na formulação de um modelo para otimizar o posicionamento de postos de carregamento de veículos elétricos em ambientes urbanos reais. Thai et al. (2007) propôs o problema k-m-CDS (conjunto m-dominante k-conectado) no contexto de redes sem fio ad hoc, com o objetivo de encontrar subconjuntos tolerantes a falhas, como a desconexão de dispositivos, em grafos de disco. Por fim, Khomami et al. (2018) aplicaram o *Minimum Positive Influence Dominating Set* (*MPIDS*), uma variação semelhante ao MkDSP, onde cada vértice deve ser "influenciado" positivamente por metade de seus vizinhos, buscando identificar indivíduos influentes em redes sociais.

1.4 METODOLOGIA

O estudo apresentado neste trabalho pode ser organizado em seis etapas principais, listadas e descritas a seguir.

I. Levantamento do estado-da-arte entre os métodos para aproximação do conjunto k-dominante mínimo em grafos não ponderados e não direcionados.

II. Coleta de dados dos trabalhos correlatos e das instâncias adequadas à proposta do trabalho, assim como as métricas usadas para avaliação dos algoritmos da literatura (Capítulos 3 e 5).

III. Elaboração dos algoritmos propostos, conforme descrito no Capítulo 4.

IV. Definição e configuração dos parâmetros associados aos algoritmos desenvolvidos, detalhada na Seção 5.2.

V. Aplicação dos algoritmos desenvolvidos nas instâncias coletadas, planejando e conduzindo uma rotina de experimentos a fim de obter as métricas de interesse.

VI. Análise comparativa entre os resultados dos algoritmos elaborados no trabalho, verificando o desempenho de cada um em relação aos demais, incluindo a melhor abordagem da literatura, como pode ser acompanhado ao longo da Seção 5.3.

As primeiras duas etapas concentram-se na exploração da literatura. Este trabalho propõe a implementação de novos métodos para o MkDSP em grafos não ponderados e não direcionados. Com esse escopo definido, torna-se possível criar novas instâncias para os testes ou identificar trabalhos prévios que se adequem ao problema, selecionando suas instâncias e os algoritmos que obtiveram os melhores resultados conhecidos. Neste estudo, optou-se pela segunda alternativa, fazendo uso de instâncias já presentes na literatura.

As etapas III, IV e V contemplam o desenvolvimento das heurísticas construtivas, bem como das metaheurísticas LNS e ALNS. A configuração de parâmetros será realizada por meio da ferramenta `irace` (mais detalhes sobre seu funcionamento serão apresentados no Capítulo 2). A execução da quinta etapa será iniciada com testes preliminares sem a configuração automatizada, a fim de analisar o desempenho isolado das heurísticas e, posteriormente, sua integração no ALNS. Em seguida, novos experimentos serão realizados com os parâmetros ajustados, buscando alcançar os melhores resultados possíveis.

Por fim, o estudo comparativo previsto na última etapa foi conduzido a partir de uma comparação direta com os melhores resultados presentes na literatura, utilizados como *benchmark* para as instâncias selecionadas. Segundo (WAINER et al., 2007), a avaliação de algoritmos é um estudo quantitativo e objetivo, podendo ser pautada a partir de alguns critérios fundamentais, como tempo de execução, qualidade dos resultados e consistência das soluções. As métricas extraídas a partir de uma série de execuções incluem tempo médio de execução, média dos resultados, desvio padrão e melhor resultado observado. Esses indicadores permitirão uma comparação direta com os resultados da literatura, a fim de averiguar a eficácia das técnicas elaboradas neste estudo em relação aos demais métodos previamente desenvolvidos.

1.5 ESTRUTURA DO TRABALHO

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica, que define formalmente o problema do conjunto k-dominante mínimo, assim

como detalha a ferramenta `irace`. O capítulo seguinte descreve as abordagens propostas na literatura, com uma análise detalhada daquelas que mais se assemelham às técnicas desenvolvidas neste trabalho. O Capítulo 4 descreve os algoritmos elaborados, incluindo seus pseudocódigos e explicações acerca de seu funcionamento e de seus componentes internos. O Capítulo 5 é dedicado à análise experimental, iniciando com experimentos comparativos entre as abordagens propostas neste trabalho. Em seguida, concentra-se na avaliação do desempenho do ALNS em relação aos métodos da literatura, finalizando com uma breve análise dos componentes-chave que integram o algoritmo ALNS. Por fim, o Capítulo 6 apresenta as considerações finais, apontando possíveis melhorias e sugestões para trabalhos futuros.

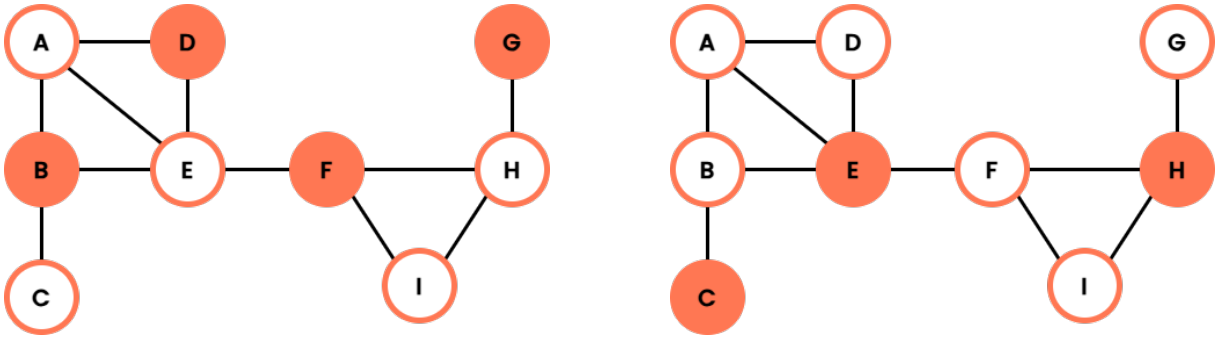
2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é apresentar os conceitos fundamentais que sustentam o desenvolvimento deste trabalho. Embora a introdução no Capítulo 1 já tenha oferecido uma breve definição, acompanhada de uma justificativa para o uso de algoritmos heurísticos e metaheurísticos, outros tópicos essenciais ainda precisam ser definidos. Alguns deles são cruciais para a compreensão geral do restante da tese, enquanto outros fornecem detalhes técnicos indispensáveis que serão retomados em momentos posteriores do trabalho. Portanto, as seções seguintes tratam da definição formal do problema do conjunto k-dominante mínimo e da configuração de parâmetros, ponto fundamental para assegurar que os algoritmos apresentados no Capítulo 4 possam alcançar seu melhor desempenho.

2.1 PROBLEMA DO CONJUNTO K-DOMINANTE MÍNIMO - MKDSP

Na teoria dos grafos, uma das classes de problemas mais estudadas e difundidas é a dos chamados problemas de dominação, cujo conceito central é o conjunto dominante. Dado um grafo $G = (V,E)$, um subconjunto $S \subseteq V$ é considerado um conjunto dominante se, para todo vértice $v \in V$, temos que $v \in S$ ou v possui ao menos um vizinho em S. A Figura 1 ilustra dois possíveis conjuntos dominantes S_1 e S_2 para um grafo G.

Figura 1 – Possíveis conjuntos dominantes S_1 e S_2 para $G = (V,E)$



(a) *conjunto minimamente dominante* $S_1 = \{b,d,f,g\}$

(b) *conjunto dominante mínimo* $S_2 = \{c,e,h\}$

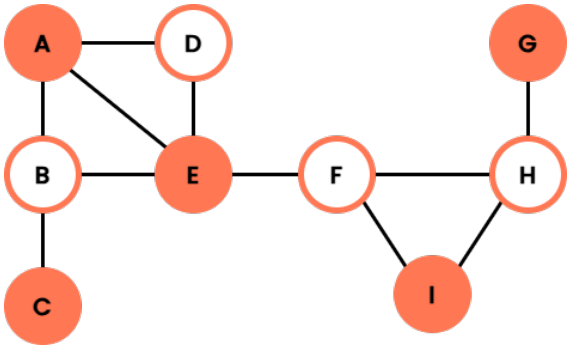
Fonte: Elaborado pelo autor, 2025.

O conjunto $S_1 = \{b,d,f,g\}$, retratado na Figura 1(a), é chamado de conjunto minimamente dominante. Ou seja, não existe um subconjunto próprio que seja um conjunto dominante, o que implica que a remoção de qualquer vértice de um conjunto minimamente dominante faz com que ele deixe de ser dominante. É notável também que $S_2 = \{c,e,h\}$ possui menor cardinalidade que o conjunto S_1. Na realidade, não há subconjunto menor que domine G. O conjunto

dominante com menor número de vértices dentre todos os possíveis para um grafo é denominado conjunto dominante mínimo, e sua cardinalidade é chamada de *domination number* (número de dominação). A busca pelo conjunto dominante mínimo é conhecida como problema do conjunto dominante mínimo (*minimum dominating set problem* — MDSP) e constitui a base dessa família de problemas.

O problema do conjunto k-dominante mínimo[1] (*minimum k-dominating set problem* — MkDSP) é uma generalização do MDSP, onde um conjunto é considerado k-dominante se cada vértice v de um $G = (V, E)$ faz parte ou é vizinho de ao menos k vértices do conjunto dominante. A Figura 2 apresenta um exemplo de conjunto 2-dominante mínimo para o mesmo grafo G introduzido anteriormente.

Figura 2 – *conjunto 2-dominante mínimo S_3 = {a, c, e, g, i}.*



Fonte: Elaborado pelo autor, 2025.

O MkDSP pode ser formalizado conforme segue.

> Seja $G = (V, E)$ um grafo simples não direcionado e $k \in \mathbb{N}$ um número fixo. Para $v \in V$, $N(v)$ é o conjunto de todos os vértices adjacentes de v no grafo G, ou seja, $N(v) = \{w \mid vw \in E\}$. O M$k$DSP é um problema de otimização cujo objetivo é encontrar um conjunto dominante com cardinalidade mínima. (PREDOJEVIĆ; KARTELJ; DJUKANOVIĆ, 2023, p. 2, tradução nossa)

Sendo assim, o MkDSP pode ser formulado como o seguinte problema de otimização (CORCORAN; GAGARIN, 2021):

$$\underset{S \subseteq V}{\text{minimiza}} \; |S|, \quad \text{sujeito a} \quad \forall v \in V \setminus S, \; |N(v) \cap S| \geq k.$$

Tratando-se de complexidade, a versão de decisão do MkDSP foi demonstrada como um problema NP-completo (LAN; CHANG, 2013), logo, sua versão de otimização é comprovadamente NP-difícil (FINK, 1985). Diversos desafios do mundo real podem ser modelados como o problema do conjunto k-dominante. Por essa razão, o MkDSP possui aplicações em diferentes áreas, incluindo alocação de postos Gagarin e Corcoran (2018), redes sociais Khomami et al.

[1] Também conhecido como Problema da k-Dominação

(2018), redes biológicas Nacher e Akutsu (2016) e redes sem fio Thai et al. (2007), conforme discutido na Seção 1.3.

2.2 CONFIGURAÇÃO DE PARÂMETROS

É comum que muitos algoritmos para problemas complexos de otimização tenham uma série de parâmetros configuráveis. Isso se deve ao fato de que não há configuração ótima para todo e qualquer tipo de aplicação, visto que tais parâmetros variam não só conforme o problema em questão, mas também com o tipo ou classe de instâncias abordadas. Em outras palavras, a configuração de seus parâmetros deve levar em consideração o contexto em que está inserido. O desempenho de algoritmos heurísticos depende frequentemente da escolha adequada desses parâmetros, que pode influenciar significativamente a qualidade das soluções encontradas. O processo de encontrar uma boa combinação desses parâmetros é conhecido como configuração de parâmetros (*parameter tuning*). Tradicionalmente, esse ajuste era feito manualmente, o que não é apenas demorado, como também propenso a vieses. O desafio de encontrar uma combinação de parâmetros que proporcione o melhor desempenho esperado sobre uma classe de instâncias pode ser considerado como um problema por si só.

A fim de contornar os obstáculos derivados do processo de configuração manual de algoritmos, surgiram métodos automáticos de configuração de parâmetros, que sistematizam esse processo. Entre as ferramentas de configuração automática de parâmetros mais usadas está o `irace` (LÓPEZ-IBÁÑEZ et al., 2016). Esse configurador é baseado no conceito de *racing*, que iterativamente compara diferentes configurações de parâmetros, avaliando-as sobre várias instâncias do problema, eliminando gradativamente amostras com desempenho inferior e focando a exploração em espaços com configurações de melhor desempenho.

O `irace` implementa uma abordagem denominada *Iterated Racing* (mais especificamente o algoritmo *Iterated F-Race*) composta por três etapas principais:

> (1) geração de novas configurações com base em uma distribuição específica, (2) seleção das melhores configurações entre as recém-geradas por meio de corridas (*racing*) e (3) atualização da distribuição de amostras, a fim de direcionar as próximas configurações para regiões promissoras do espaço de busca. Essas três etapas são repetidas até que um critério de parada seja atendido. (LÓPEZ-IBÁÑEZ et al., 2016, p. 3, tradução nossa)

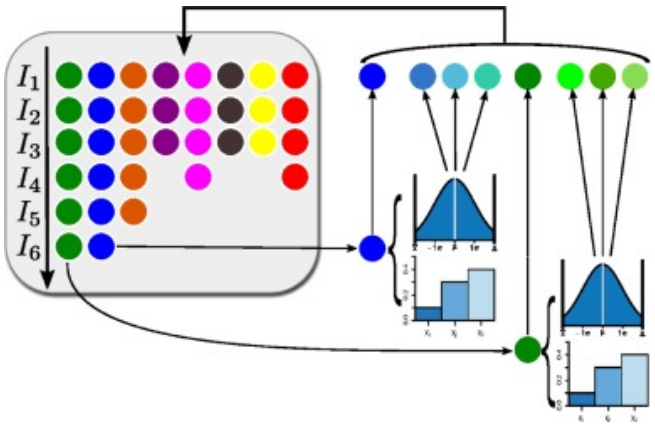
No processo de avaliação, o `irace` executa as configurações candidatas em uma sequência de instâncias, descartando progressivamente aquelas que apresentarem desempenho estatisticamente inferior. O mecanismo de eliminação se baseia em testes estatísticos (como o teste de Friedman) e visa concentrar os recursos computacionais nas configurações com maior potencial. As distribuições usadas para gerar novas configurações são adaptadas continuamente, incorpo-

rando os valores dos parâmetros das configurações bem-sucedidas, o que reforça a exploração de regiões promissoras no espaço de busca.

Juntamente à sua natureza iterativa, o `irace` conta com uma série de mecanismos adicionais que aumentam sua eficácia. O elitismo preserva as melhores configurações ao longo das iterações; um mecanismo de *restart* evita a estagnação da busca; por fim, execuções de configurações claramente inferiores são encurtadas a fim de poupar tempo e recursos computacionais (CÁCERES et al., 2017). Essas estratégias provam-se eficientes na configuração de parâmetros de algoritmos onde o espaço de parâmetros é grande e a avaliação é custosa.

A Figura 3 ilustra o fluxo geral do processo iterativo de configuração de parâmetros no `irace`. À esquerda, observa-se uma tabela onde cada coluna representa uma configuração de parâmetros, e cada linha corresponde a uma instância do problema. Os círculos coloridos indicam a execução de uma determinada configuração sobre uma instância, e diferentes cores representam diferentes configurações. Configurações de desempenho inferior são eliminadas com o tempo. À direita, observa-se o processo de atualização da amostragem, com as melhores configurações (verde e azul) influenciando a geração de novas amostras.

Figura 3 – Fluxo do procedimento *Iterated Racing*



Fonte: López-Ibáñez et al. (2016)

O amplo uso do `irace` na literatura é prova de sua versatilidade e efetividade, tendo aplicações na configuração de diversos algoritmos para diferentes problemas de otimização combinatória, como problemas de agendamento Reid et al. (2019) e roteamento Pugliese et al. (2022) e uma série de outras classes de problemas. No contexto deste trabalho, o `irace` foi utilizado para ajustar os parâmetros das heurísticas desenvolvidas para o M*k*DSP, sobretudo da metaheurística ALNS. Tendo em vista a complexidade do problema, assim como a variabilidade das soluções geradas por conta da natureza estocástica dos algoritmos desenvolvidos, a configuração automática de parâmetros se torna importante para garantir o melhor desempenho dos métodos propostos.

3 TRABALHOS CORRELATOS

Este capítulo é destinado à revisão dos métodos presentes na literatura que visam resolver ou aproximar o problema da k-dominação. Embora existam variações do problema e aplicações voltadas a diferentes classes de grafos (como brevemente discutido na Seção 1.3), este trabalho concentra-se na formulação do MkDSP aplicada a grafos não ponderados e não direcionados. Portanto, os métodos abordados nesta revisão seguirão estritamente essa orientação.

3.1 ANÁLISE DA LITERATURA

Por ser um problema de otimização clássico, o MDSP possui uma variedade de técnicas propostas, desde métodos exatos, variadas estratégias de construção heurísticas e metaheurísticas. Contudo, isso é válido apenas para o problema básico (i.e., com $k = 1$). Como será discutido adiante, o MkDSP não conta com a mesma quantidade e variedade de propostas. Para Corcoran e Gagarin (2021) os métodos presentes na literatura podem ser diferenciados a partir de cinco características: a primeira diz respeito se o algoritmo foi concebido para o problema da dominação clássica; a segunda verifica se a técnica é aplicável a casos onde $k > 1$; as últimas três características referem-se à abordagem do algoritmo, se foi utilizado um algoritmo guloso, uma metaheurística ou um método exato. A Tabela 1 apresenta um resumo das abordagens presentes na literatura com base nessas cinco características.

Nota-se que há uma quantidade consideravelmente menor de algoritmos que satisfazem o critério da segunda coluna da tabela, o que evidencia a escassez de métodos propostos especificamente para o MkDSP quando $k \geq 1$. Observa-se também que a maioria das contribuições para o MkDSP surgiu nas últimas duas décadas, o que indica que o desenvolvimento da área se encontra em seu estágio inicial, que vem diversificando as estratégias aplicadas ao problema nos últimos anos. Com o objetivo de oferecer um panorama mais detalhado sobre as técnicas empregadas, é realizada a seguir uma breve análise dos trabalhos que são aplicáveis ao MkDSP onde $k \geq 1$. Também é detalhado o funcionamento do algoritmo proposto por Casado et al. (2023), em razão das semelhanças entre este e o ALNS explorado neste trabalho.

Entre os métodos levantados na Tabela 1 para o MkDSP, é possível encontrar ao menos uma abordagem em cada uma das categorias listadas (Heurísticas Gulosas, Metaheurísticas e Métodos Exatos). Contudo, é importante destacar que as metaheurísticas constituem aplicações bastante recentes ao problema. Até o momento, apenas Corcoran e Gagarin (2021) e Predojević, Kartelj e Djukanović (2023) exploraram esse tipo de técnica, usando, respectivamente, os algo-

Tabela 1 – Métodos para resolver o problema de k-dominação com base em cinco características.

	$k=1$	$k>1$	Heurística Gulosa	Metaheurística	Método Exato
Parekh (1991)	✓		✓		
Sanchis (2002)	✓		✓		
Stephen Eubank et al. (2004)	✓		✓		
Chellali et al. (2012)	✓	✓	✓		
Hedar E Ismail (2010)	✓			✓	
Hedar E Ismail (2012)	✓			✓	
Ho et al. (2006)	✓			✓	
Nehéz et al. (2015)	✓				✓
Bird (2017)	✓				✓
Assadian (2019)	✓				✓
Couture et al. (2008)	✓	✓	✓		
Gagarin, Poghosyan e Zverovich (2013)	✓	✓	✓		
Gagarin e Corcoran (2018)	✓	✓	✓		✓
Corcoran e Gagarin (2021)	✓	✓	✓	✓	
Casado et al. (2023)	✓		✓	✓	
Predojević, Kartelj e Djukanović (2023)	✓	✓		✓	

Fonte: Adaptado de Corcoran e Gagarin (2021).

ritmos *Beam Search* e *Variable Neighborhood Search* para aproximar o conjunto k-dominante em instâncias que modelam redes urbanas reais.

No grupo de métodos exatos (e determinísticos) foi explorada apenas uma formulação exata via programação linear inteira. Segundo (GAGARIN; CORCORAN, 2018), apesar de que resultados ótimos tenham sido alcançados em grafos de até 50 vértices e $k = \{1,2,4\}$, a proposta apresentou baixa escalabilidade, apresentando baixo desempenho em grandes instâncias ou maiores valores de k.

No mesmo trabalho, Gagarin e Corcoran (2018) propuseram uma generalização do algoritmo guloso clássico ($k = 1$) para o MkDSP. A estratégia parte de um conjunto vazio, ao qual são adicionados vértices cujas vizinhanças contêm o maior número de vértices ainda não suficientemente dominados. Já Gagarin, Poghosyan e Zverovich (2013) apresentaram uma construção aleatória, iniciando com uma seleção parcial aleatória e adicionando novos vértices conforme a necessidade de dominação, aplicando diferentes funções de avaliação para a escolha dos vértices. Por fim, Couture et al. (2008) propõem uma abordagem iterativa para determinar o conjunto k-dominante mínimo, baseada na construção sucessiva de conjuntos independentes máximos (*maximal independent sets*) para cada valor de k. O método inicia com $k = 1$ e, a cada iteração, constrói um conjunto independente máximo cujos vértices são adicionados ao conjunto solução. Esse processo é repetido progressivamente, formando inicialmente um conjunto *1*-dominante mínimo, seguido por um *2*-dominante, e assim sucessivamente, até alcançar um conjunto k-dominante mínimo.

3.2 *ITERATED GREEDY* PARA O MDSP

Casado et al. (2023) aplicaram a metaheurística *Iterated Greedy* (IG) ao MDSP. O IG foi inicialmente proposto por Ruiz e Stützle (2007). Embora o foco deste trabalho seja o problema da k-dominação com $k \geq 1$, o algoritmo desenvolvido (ALNS) guarda similaridades com o IG. Isso se deve ao fato de o ALNS ser uma extensão da metaheurística *Large Neighborhood Search* (LNS), proposta por Shaw (1998), a qual compartilha princípios conceituais e estruturais com o IG. Ambas as metaheurísticas (IG e LNS) apresentam uma abordagem iterativa, seguindo o princípio comum de explorar o espaço de soluções por meio de um ciclo de destruição parcial e reconstrução, buscando escapar de ótimos locais e melhorar progressivamente a qualidade da solução obtida. A única diferença entre elas diz respeito à forma como a solução é reconstruída, IG faz uso de heurísticas gulosas, e mesmo que o LNS possa fazer uso delas, também se permite usar outros meios para explorar o espaço de soluções.

A metaheurística IG apresentada por Casado et al. (2023) conta com três principais heurísticas, responsáveis por etapas diferentes do algoritmo: construção inicial, destruição parcial e reconstrução da solução. Em uma fase de pré-processamento, a instância é analisada a fim de identificar os vértices "folha" (vértices ligados a um ou menos vértices que, consequentemente, devem estar no conjunto solução) a fim de reduzir o espaço de busca. A seguir, a fase de construção inicial visa gerar uma solução viável, i.e. um conjunto dominante. Essa construção inicial pode ser realizada pelos métodos *Greedy Insertion Procedure* (GIP) e *Greedy Deletion Procedure* (GDP). O GIP parte de um conjunto parcial contendo os vértices de suporte e insere iterativamente os vértices que dominam o maior número de vértices não dominados, enquanto o GDP parte de um conjunto contendo todos os vértices e remove progressivamente os que são menos essenciais. Ambos os métodos garantem a viabilidade da solução e aplicam, ao final, um procedimento para eliminar vértices redundantes e obter um conjunto minimamente dominante.

Após a construção da solução inicial, o IG entra em um ciclo de melhoria iterativa composta pelas fases de destruição parcial, reconstrução e refinamento. A destruição remove uma fração β dos vértices do conjunto atual, podendo seguir uma estratégia aleatória ou gulosa. Em seguida, a fase de reconstrução visa restaurar a dominância do conjunto, podendo adicionar vértices aleatoriamente ou com base em um critério guloso equivalente ao usado no GIP. É importante citar que a estrutura do IG faz uso de apenas um componente de destruição e reparação. Portanto, ao escolher um operador de destruição ou reparação, não é possível alterá-lo durante a execução. Por fim, a etapa de construção é sucedida por uma etapa de melhoria local baseada em movimentos de troca, através de uma busca local que preserva a viabilidade da solução. Uma última etapa de eliminação de redundâncias é realizada ao finalizar o processo de todo o algoritmo.

Os resultados dos experimentos reportados por Casado et al. (2023) mostram que a melhor configuração é aquela que usa o GIP para gerar a solução inicial, destruição aleatória e reconstrução gulosa. As soluções produzidas pelo algoritmo sob essa configuração são melhores, e o tempo de execução é menor. Esse algoritmo provou-se superior às demais metaheurísticas da literatura para as instâncias testadas.

3.3 METAHEURÍSTICAS APLICADAS AO M*k*DSP

Esta seção descreve as metaheurísticas empregadas na resolução do M*k*DSP. Como já mencionado, esse tipo de abordagem ainda é pouco explorado para o problema, representando uma contribuição recente na literatura. As metaheurísticas identificadas até o momento são o *Beam Search* e o *Variable Neighborhood Search*, propostas originalmente por Ow e Morton (1988) e Mladenović e Hansen (1997), respectivamente.

3.3.1 *Beam Search* com Construção Gulosa

Corcoran e Gagarin (2021) apresentam uma nova formulação matemática para o M*k*DSP, que indica o nível de cobertura de cada vértice em vez de apenas verificar se o vértice é dominado ou não. O nível de cobertura de um vértice é dado por

$$C(D,v) = \min\left(k,\ |N(v) \cap D|\right),$$

e a formulação consiste em maximizar o nível de cobertura, sendo

$$\arg\max_{D \subseteq V} \sum_{v \in V \setminus D} C(D,v) \quad \text{sujeito a} \quad \forall v \in V \setminus D,\ |N(v) \cap D| \geq k.$$

Com base nessa formulação, os autores desenvolveram uma heurística gulosa que constrói iterativamente um conjunto k-dominante selecionando, a cada passo, o vértice que maximiza a função objetivo, isto é

$$\arg\max_{u \in V \setminus D} \sum_{v \in V \setminus (D \cup \{u\})} C(D \cup \{u\}, v).$$

Dessa forma, garante-se a escolha de vértices que ampliem a cobertura do grafo, priorizando regiões ainda pouco cobertas no grafo, de modo a evitar a seleção de vértices que reforçam áreas já dominadas.

A fim de superar limitações da construção gulosa pura (que pode ficar presa em ótimos locais), o artigo propõe a metaheurística *Beam Search*, uma variante da busca em largura. A

partir de uma solução inicial e de um parâmetro de largura b (denominado *beam width*), são geradas novas soluções candidatas, avaliadas por uma função heurística (neste caso, a métrica baseada na cobertura). Em seguida, apenas as b melhores soluções são mantidas. Isso ocorre a cada nível da árvore de busca, gerando novas soluções com um vértice adicional. Vale ressaltar que para $b = 1$, o *Beam Search* é equivalente à heurística construtiva gulosa.

Corcoran e Gagarin (2021) exploraram três valores distintos para o parâmetro b (1, 2 e 4), nomeados $BS1$, $BS2$ e $BS4$, respectivamente. Os experimentos, realizados em instâncias de grafos que modelam cidades inglesas e algumas capitais europeias, indicaram que a heurística *Beam Search* superou os demais métodos da literatura até então. Dentre os valores testados, $BS4$ apresentou o melhor desempenho em qualidade de solução, embora com um tempo de execução consideravelmente maior. Já $BS2$ demonstrou ser um bom compromisso entre qualidade e tempo de processamento, enquanto $BS1$ também se mostrou eficiente em certas condições, oferecendo resultados quase instantâneos.

3.3.2 *Variable Neighborhood Search*

Predojević, Kartelj e Djukanović (2023) propuseram a metaheurística *Variable Neighborhood Search* (VNS) para resolver o problema da k-dominação, buscando superar as limitações dos algoritmos previamente discutidos na literatura (em especial o *Beam Search*). A abordagem amplia o espaço de busca ao explorar tanto soluções viáveis quanto inviáveis, utilizando uma função de avaliação que penaliza proporcionalmente a violação da restrição de k-dominação na função objetivo, ao mesmo tempo em que minimiza o tamanho do conjunto dominante. A função objetivo é dada por F, sendo

$$F(D) = (1 + viols(D)) \cdot (1 + penalty \cdot |D|),$$

$$viols(D) = \sum_{v \in V \setminus D} \big(k - C(D, v)\big).$$

O termo $viols(D)$ conta o número de violações acumuladas da restrição de k-dominação. Ou seja, conta o número de vértices fora de D que não estão dominados. Essa quantidade é multiplicada por $penalty$, que define a penalização pela violação da restrição. Valores menores de $penalty$ tendem a preferir soluções viáveis, mesmo que maiores. Contudo, é necessário garantir que este valor não seja tão baixo a ponto de inflar a solução somente para viabilizá-la.

A metaheurística VNS explora diferentes vizinhanças por meio de um mecanismo de *shaking* (destruição parcial), que remove d vértices da solução atual (variando entre $d_{\min}$ e

$d_{\max}$), permitindo explorar a mesma vizinhança ou escapar dela caso não traga melhorias. Após essa destruição, o algoritmo realiza uma busca local em duas fases: primeiro, adiciona vértices para restabelecer a viabilidade; em seguida, remove vértices redundantes para reduzir o tamanho do conjunto sem comprometer a dominância. Por fim, a solução é então avaliada pela função objetivo F, dando início a uma nova iteração.

A configuração dos parâmetros mencionados anteriormente foi realizada com o auxílio da ferramenta *Grid Search* e os experimentos usaram as mesmas instâncias desenvolvidas por Corcoran e Gagarin (2021). Os resultados mostraram que o VNS obteve melhorias significativas na qualidade das soluções, principalmente para $k = 2$ e $k = 4$, estabelecendo novos resultados de estado da arte.

4 ABORDAGEM PROPOSTA

Como discutido anteriormente, o MkDSP é um desafio complexo que pertence à classe de problemas *NP*-difícil. Exceto para instâncias de pequeno porte, técnicas exatas encontram grande dificuldade em apresentar a solução ótima em tempo razoável (CORCORAN; GAGARIN, 2021). Por esse motivo, estratégias heurísticas são amplamente empregadas para desafios dessa complexidade, buscando soluções de qualidade aceitável em prazos razoáveis. Os diferentes algoritmos propostos para o MkDSP foram detalhadamente revisados no Capítulo 3.

Este capítulo tem como objetivo apresentar a abordagem proposta para aproximar o conjunto k-dominante mínimo em grafos não ponderados e não direcionados. Dentre os algoritmos desenvolvidos, destaca-se a metaheurística ALNS (*Adaptive Large Neighborhood Search*), cuja estrutura faz uso de diversos operadores e componentes para explorar o espaço de busca. Muitos desses operadores são heurísticas que também foram implementadas e avaliadas isoladamente, tópico tratado no Capítulo 5. Para maior clareza, a descrição dos algoritmos desenvolvidos segue a ordem abaixo.

I. Componentes heurísticos que integram o ALNS;

II. Metaheurística LNS (*Large Neighborhood Search*), incluindo os operadores de destruição e reconstrução;

III. Metaheurística ALNS (*Adaptive Large Neighborhood Search*), apresentando seus operadores e mecanismos adaptativos.

4.1 HEURÍSTICAS CONSTRUTIVAS GULOSAS PARA O MKDSP

Uma heurística construtiva gulosa é um método de construção de soluções que iterativamente escolhe localmente a opção mais promissora, segundo algum critério. Em geral, heurísticas construtivas são simples, rápidas e geralmente não garantem a solução ótima, mas produzem resultados aceitáveis em baixo tempo de processamento. As heurísticas apresentadas nesta seção foram originalmente propostas por Rangel (2023), que desenvolveu uma série de abordagens gulosas para o MkDSP. Neste trabalho, foram realizadas apenas pequenas adaptações pontuais, focadas na forma em que as métricas de avaliação de cada vértice eram atualizadas, sem alterar o funcionamento essencial dos métodos originais.

Todos os algoritmos gulosos apresentados a seguir compartilham de uma estrutura básica, com uma fase de inicialização padronizada, conforme apresentado pelo Algoritmo 1. Cada heurística recebe como entrada um grafo $G = (V, E)$ e um número inteiro positivo k, e computa um

Algoritmo 1: Estrutura[1] das Heurísticas Construtivas Gulosas para o MkDSP

Input : Grafo $G = (V, E)$ e constante k.
Output: Conjunto S capaz de k-dominar G.

foreach $v \in V$ **do**
  $K[v] \leftarrow k$
  $I[v] \leftarrow \big(M_1[v], M_2[v], \ldots, M_n[v]\big)$; `// Inicialize métricas do vetor I`
end
$S \leftarrow \emptyset$
$R \leftarrow V$; `// Conjunto de vértices ainda não k-dominados`
while $R \neq \emptyset$ **do**
  Encontre $v = \arg\max_{u \in R} F(u)$
  $S \leftarrow S \cup \{v\}$
  foreach $u \in N(v)$ **do**
    if $u \in R$ **then**
      $K[v] \leftarrow K[v] - 1$
      $UpdateMetrics(u)$; `// Atualize demais métricas de u`
      if $K[v] = 0$ **then**
        $R \leftarrow R \setminus \{u\}$; `// Vértice u dominado`
        $UpdateMetrics(N(u))$; `// Atualize métricas dos vizinhos de u`
      end
    end
  end
  $R \leftarrow R \setminus \{v\}$
end
return S

conjunto k-dominante S para G. O algoritmo inicializa o conjunto solução S como o conjunto vazio, e vértices são iterativamente adicionados a S até que se forme um conjunto k-dominante. O vértice selecionado a cada passo é determinado pela função de avaliação específica de cada heurística.

Além disso, dois vetores K e I são inicializados após a leitura do grafo G e do inteiro k. O primeiro tem como propósito guardar a quantidade de vizinhos que ainda restam para que o vértice v esteja k-dominado, i.e. $K[v] = (k - |N(v) \cap S|)$. O vetor I por sua vez, armazena informações relevantes para cada vértice $v \in V$, como as métricas utilizadas para definir o vértice mais apto de acordo com a função de avaliação. Nesse contexto, $I[v] = \big(M_1^v, M_2^v, \ldots, M_n^v\big)$ pode conter diversas métricas M_i para o vértice v.

Após a fase de pré-processamento das métricas, é inicializado um conjunto S vazio, que representa os vértices selecionados para formar o conjunto k-dominante de G. Simultaneamente, é definido um conjunto R, que contém os vértices ainda não dominados, definido como

$$R = \{v \in V \setminus S \mid |N(v) \cap S| < k\}.$$

[1] Pseudocódigo 1 adaptado de Rangel (2023).

O critério de parada é atingido quando $R = \emptyset$, indicando que não restam vértices não dominados no grafo. Além disso, um vértice v é considerado elegível para ser adicionado ao conjunto solução se e somente se $v \in R$, i.e. se não pertence ao conjunto S e possui menos de k vizinhos em S.

Outro elemento central compartilhado por todas as heurísticas é o vetor K, que armazena uma métrica comum a todos os métodos, chamada grau de dominação, que indica quantos vértices ainda são necessários para que um vértice v esteja devidamente dominado. Inicialmente, este valor é igual ao inteiro k para todos os vértices e é atualizado pelo procedimento invocado na linha 13 do Algoritmo 1, cada vez que um vértice é adicionado ao conjunto solução. Deste modo, $K[v]$ para um vértice $v \in V$ é dado por

$$K[v] = k - |N(v) \cap S|.$$

As principais diferenças entre as heurísticas construtivas estão nas demais métricas armazenadas no vetor I e nas definições das funções F e *UpdateMetrics* que correspondem, respectivamente, à função de avaliação adotada e à forma como as métricas são atualizadas ao longo do processo. Esses elementos específicos serão discutidos nas subseções seguintes.

4.1.1 Heurística Baseada em Vizinhos Remanescentes

Esta primeira heurística construtiva prioriza os vértices com a maior quantidade de vizinhos ainda não dominados. Tratando-se de heurísticas construtivas, pode ser considerada a estratégia mais intuitiva para o problema da k-dominação, uma vez que os vértices com maior grau de vizinhança são os mais críticos e influentes em qualquer grafo. Essa abordagem é equivalente ao método apresentado por Gagarin e Corcoran (2018), discutido na Seção 3, que por sua vez é uma generalização da heurística padrão para o problema da dominação, originalmente proposta por (PAREKH, 1991).

A heurística em questão é denominada H_{V}, sendo sua função de avaliação (representada por F no Algoritmo 1) formulada como $|N(v) \cap R|$, que, por sua vez, será chamada de *grau de vizinhança* daqui em diante. Desse modo, é iterativamente escolhido o vértice que maximiza a expressão

$$\arg\max_{u \in R} |N(u) \cap R|.$$

Essa heurística utiliza apenas o grau de vizinhança como métrica para sua função de avaliação. Portanto, $I[v] = |N(v) \cap R|$ para cada vértice $v \in V$. Esse valor é atualizado para um vértice u segundo a regra $I[u] \leftarrow I[u] - 1$ em dois momentos: (1) quando um de seus vizinhos é

selecionado para o conjunto solução; e (2) quando esse vizinho é dominado (conforme as linhas 14 e 17 do Algoritmo 1).

A Figura 4 ilustra, passo a passo, a aplicação da heurística H_{V} ao grafo $G_1 = (V,E)$, em que $V = \{a,b,c,d,e,f\}$ e $E = \{(a,b),(a,c),(a,d),(b,e),(c,f),(d,f),(e,f)\}$. Os vértices destacados em laranja representam os vértices que compõem o conjunto solução. Aqueles com borda laranja, mas sem preenchimento, indicam vértices dominados. Os vértices com borda acinzentada representam aqueles ainda não dominados.

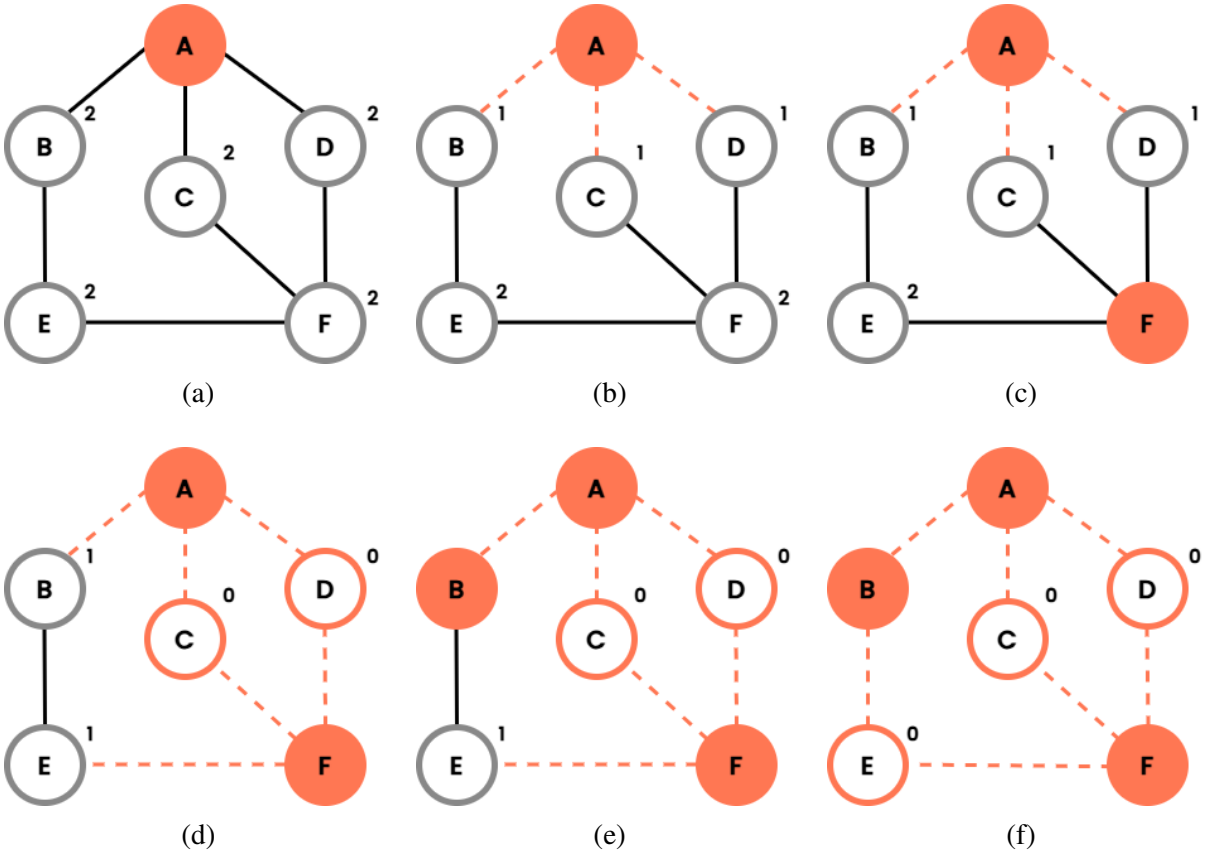
As arestas pretas indicam conexões ativas entre vértices, enquanto as arestas tracejadas representam conexões que já foram desconsideradas pelo vetor de métricas (vetor *I*). Complementando, quando arestas tracejadas também forem coloridas em laranja, isso denota relação de "influência", isto é, o vértice alaranjado contribui para a dominação de seu vizinho. Finalizando, vértices tracejados e acinzentados simplesmente indicam que não há relação de influência entre os dois vértices, apenas que a aresta não está sendo considerada, relação comum entre vértices dominados e vértices ainda não dominados.

Por fim, os pequenos números ao lado de cada vértice indicam quantos de seus vizinhos no conjunto solução ainda são necessários para que determinado vértice seja dominado. Assim como os demais exemplos da Seção 4.1 apresentados adiante, a Figura 4 define o requisito de dominação como $k = 2$, sendo assim, todos os vértices iniciam com esse número os acompanhando.

Na primeira iteração (Figura 4a), o algoritmo poderia escolher os vértices a ou f, pois há um empate no número de vizinhos não dominados. Como os algoritmos gulosos apresentados nesta seção são determinísticos, não possuindo qualquer elemento estocástico, o critério de desempate segue a ordem de visitação dos vértices, que neste exemplo, corresponde à ordem alfabética. Nesse cenário, o vértice a é selecionado primeiro. Quando o vértice a é inserido na solução, ele passa a contribuir para a dominação de seus vizinhos b, c e d, diminuindo o número de vértices adicionais necessários para que esses sejam dominados, como ilustrado na Figura 4b.

Após essa atualização, os graus de vizinhança dos vértices são ajustados, ficando da seguinte forma: $\{(a:0),(b:1),(c:1),(d:1),(e:2),(f:3)\}$. O vértice a deixa de pertencer ao conjunto dos vértices não dominados e, portanto, não é mais elegível para seleção. Mesmo que fosse, a prioridade passaria a ser do vértice f, que agora possui o maior grau de vizinhança. Uma vez que f é selecionado e domina seus vizinhos (Figura 4c) os vértices c e d, que encontravam-se sob a "influência" de a, passam a ser completamente dominados. Restam então, apenas os vértices b e e, ambos com apenas um vizinho restante. Seguindo a ordem de visitação, o vértice b é escolhido, como retrata a Figura 4e, resultando no conjunto solução $S = \{a,b,f\}$.

Figura 4 – Aplicação da heurística H_{V} ao grafo G_1.



Fonte: Adaptado de (RANGEL, 2023)

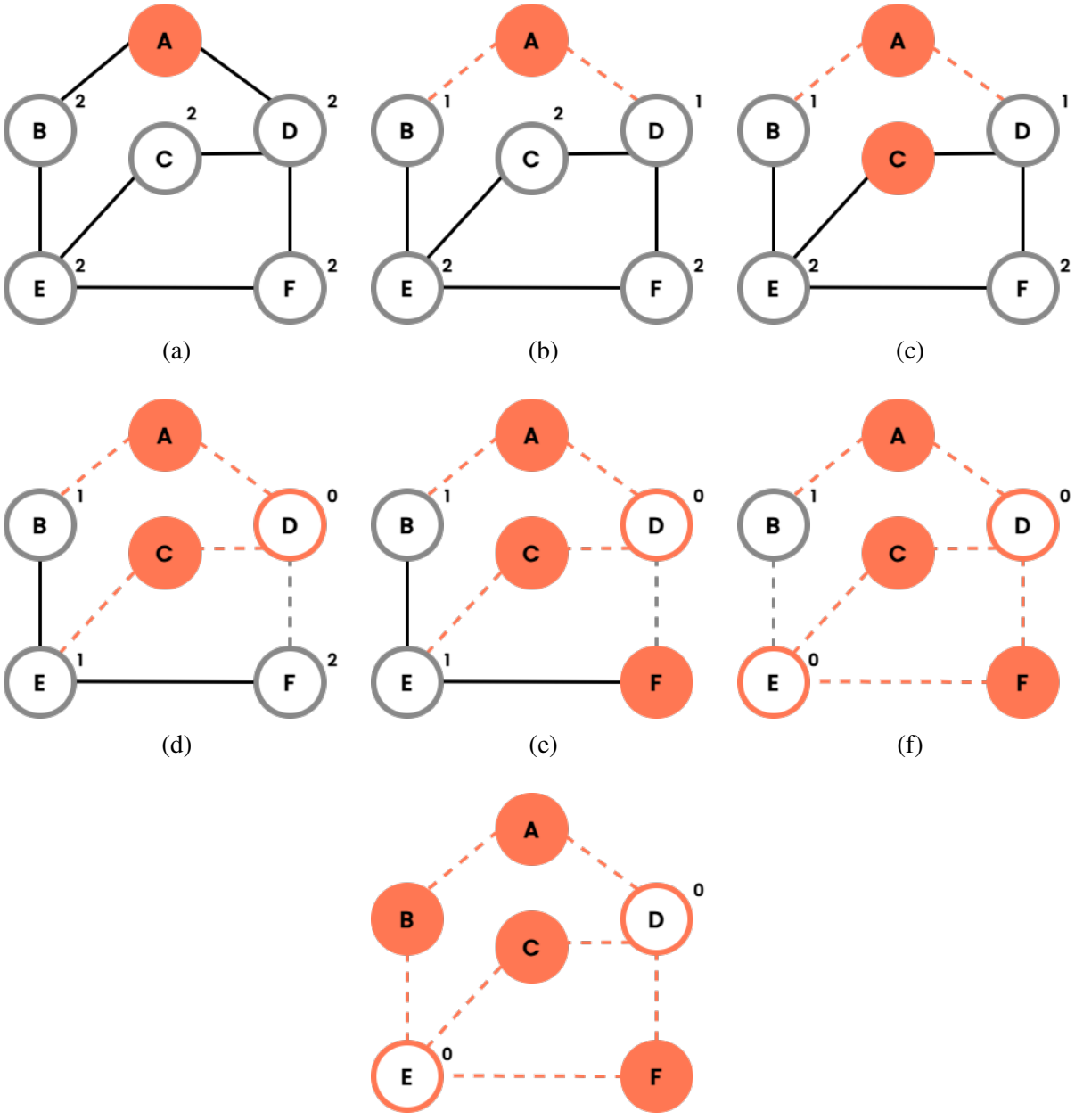
4.1.2 Heurística Baseada no Grau de Dominação

A heurística baseada no grau de dominação seleciona, a cada iteração, o vértice menos dominado, i.e. que exige um maior número de vizinhos no conjunto solução. A estratégia central consiste em ampliar a cobertura em regiões do grafo ainda pouco exploradas, priorizando vértices que permanecem menos atendidos, buscando assim uma expansão mais distribuída e eficiente do conjunto solução. A função de avaliação dessa heurística é selecionar o vértice com o maior valor no vetor K. Essa função pode ser formalizada como

$$\arg\max_{u \in R} \; k - |N(u) \cap S|.$$

Neste trabalho, essa heurística é denominada H_{D}. Como a única métrica usada para calcular a função de avaliação está contida no vetor K, que é atualizado sempre que um vértice é inserido no conjunto solução S, o vetor auxiliar I permanece vazio durante toda a execução. A Figura 5 demonstra a aplicação do método no grafo G_2, muito similar à estrutura do grafo G_1 utilizado anteriormente. Os valores indicados ao lado dos vértices representam a quantidade de

Figura 5 – Aplicação da heurística H_{D} ao grafo G_1.



Fonte: Adaptado de (RANGEL, 2023)

vizinhos que dominam cada vértice no estado atual da solução, servindo também para indicar aquele com maior prioridade para ser incluído no conjunto solução.

Como todos os vértices iniciam com o mesmo grau de dominação ($k = 2$), a Figura 5a apresenta o vértice a como o selecionado na primeira iteração, já que é o primeiro em ordem alfabética. Na sequência, os vértices c e f são escolhidos, pois não são vizinhos dos vértices previamente selecionados, como ilustram, respectivamente, as Figuras 5c e 5e. Por fim, o vértice b é adicionado, resultando no conjunto solução final $S = \{a, c, f, b\}$.

4.1.3 Heurísticas Híbridas

A partir dos critérios apresentados nas duas heurísticas anteriores, foram desenvolvidas outras duas heurísticas híbridas, que combinam os conceitos de grau de dominação e grau de vizinhança em funções de avaliação que diversificam as estratégias abordadas até o momento. Para introduzir as equações que definem essas heurísticas, é necessário esclarecer como as métricas de interesse são armazenadas. O vetor K permanece responsável pelo grau de dominação, enquanto o vetor auxiliar I é composto por duas métricas M_α e M_β, que armazenam os valores do grau de vizinhança e o resultado da função de avaliação calculada para cada vértice a cada iteração, respectivamente.

Seja V o grau de vizinhança de um vértice v, definido como $|N(v) \cap R|$, D seu grau de dominação, dado por $k - |N(v) \cap S|$, e $|R|$ a quantidade de vértices não dominados. Tendo isso em vista, as heurísticas híbridas usam as seguintes funções de avaliação:

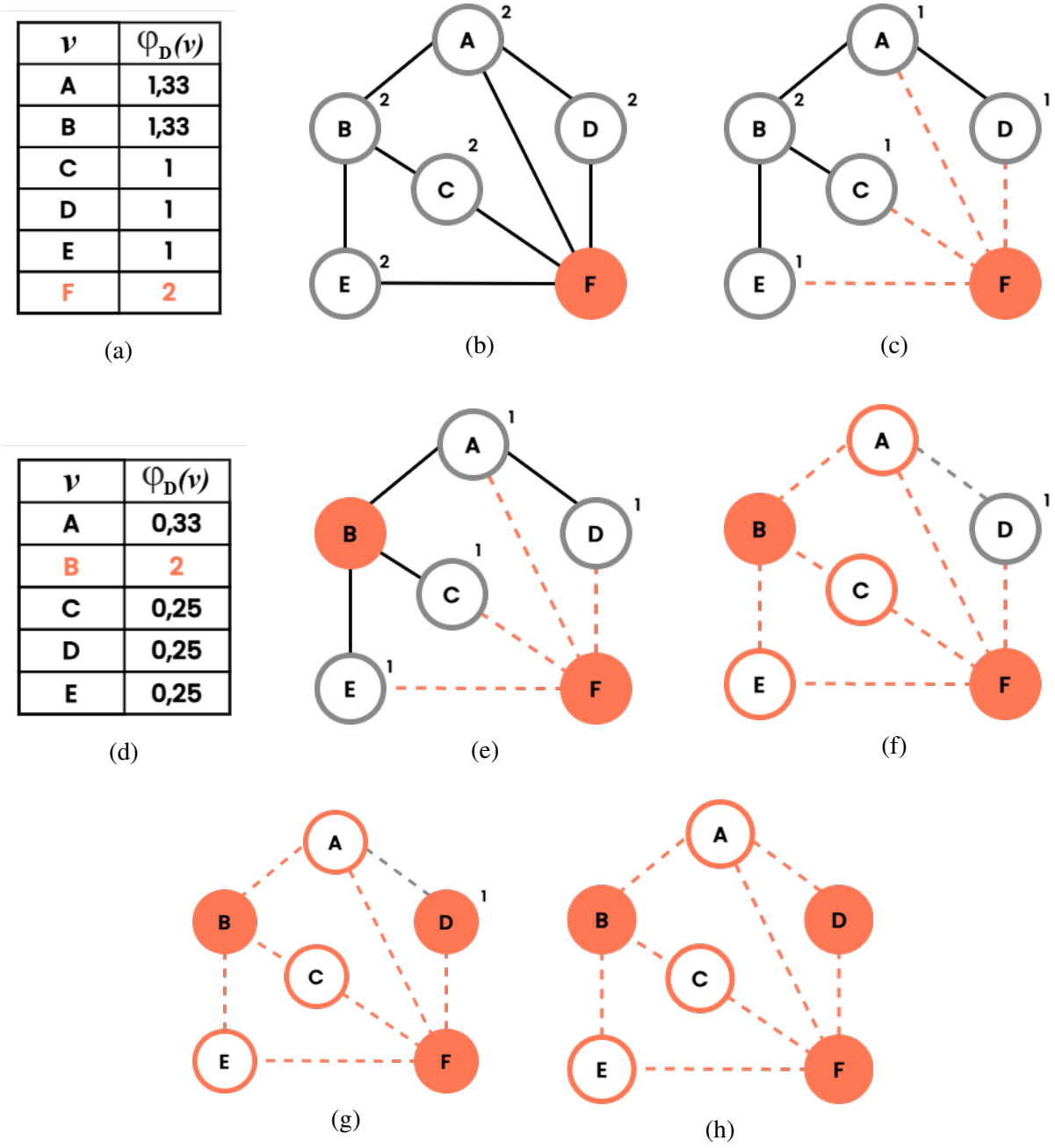
$$\varphi_D(v) = \frac{D[v]^2}{|R| - V[v]},$$

$$\varphi_V(v) = \frac{V[v]^2}{|R| - D[v]}.$$

Essas funções de avaliação também foram propostas por Rangel (2023). A heurística $H\varphi_D$, que usa a função $\varphi_D(v)$, prioriza vértices críticos, ou seja, aqueles que ainda estão distantes de alcançar k vizinhos no conjunto solução. No entanto, essa heurística introduz um critério de desempate, atribuindo maior prioridade aos vértices com maior grau de vizinhança, ponderado no denominador da equação. Já a heurística $H\varphi_V$, que usa a função $\varphi_V(v)$, equilibra ambos os fatores de forma a criar uma estratégia mais balanceada, buscando vértices que não apenas estejam pouco dominados, mas que também tenham uma ampla vizinhança ainda não suficientemente atendida.

A Figura 6 esclarece o funcionamento da heurística $H\varphi_D$, apresentando os vértices selecionados em cada passo da execução do algoritmo, acompanhados de tabelas que exibem o resultado da equação φ_D a cada vértice do grafo. Na primeira iteração, o vértice f é selecionado, uma vez que todos os vértices possuem o mesmo grau de dominação inicial. Contudo, f se destaca por ter o maior número de vizinhos no conjunto dos vértices não dominados e, consequentemente, é o vértice com melhor avaliação pela função de avaliação, como aponta a Figura 6a. Ao ser adicionado à solução, sua influência se estende aos vértices de sua vizinhança, $N(f) = \{a, c, d, e\}$, que passam a ser menos atrativos segundo a métrica φ_D, já que estão mais próximos de serem totalmente dominados (Figura 6c).

Figura 6 – Aplicação da heurística $H\varphi_D$ ao grafo G_3.



Fonte: Adaptado de (RANGEL, 2023)

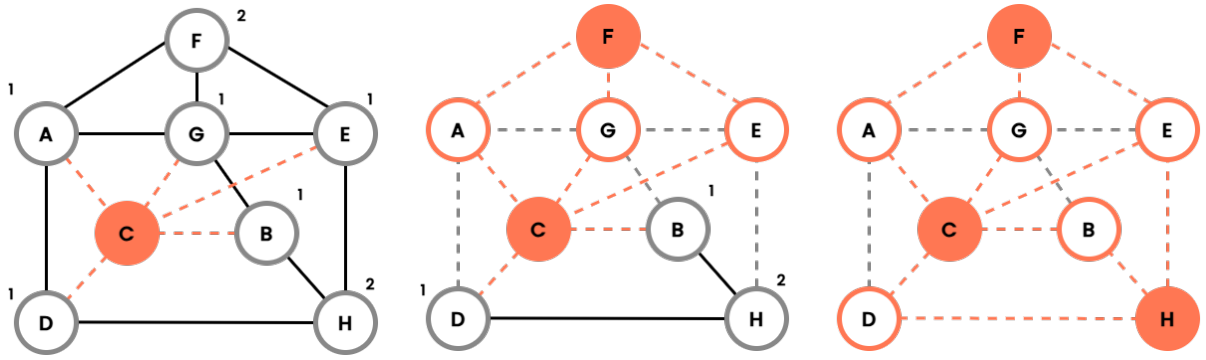
A iteração seguinte recupera o grau de dominação e vizinhança atualizados de todos os vértices (estes valores são atualizados de maneira idêntica à heurística H_D). Também é atualizada a cardinalidade do conjunto R, pois o vértice f já não compõe mais o conjunto de vértices não dominados. A partir disso, a função de avaliação é recalculada para todo vértice $v \in R$, os valores podem ser conferidos a partir da Figura 6d. A segunda iteração aponta o vértice b como o mais indicado, adicionando-o à solução. Por fim, restando apenas um último vértice não dominado (Figura 6f), d é selecionado, resultando no conjunto dominante $S = \{f, b, d\}$.

A fim de proporcionar uma visão mais concreta da diferença do comportamento dessas duas estratégias, é apresentado o grafo G_4 em duas figuras diferentes, sendo resolvido pela heurística $H\varphi_D$ na Figura 7 e $H\varphi_V$ na Figura 8.

A heurística $H\varphi_D$ seleciona, na primeira iteração (Figura 7a), o vértice c, pois a função de avaliação φ_D prioriza vértices com menor grau de dominação, aplicando como critério de desempate o grau de vizinhança. Todos os vértices iniciam com o mesmo grau de dominação, equivalente ao valor de $k = 2$, como nos exemplos anteriores, de modo que c e g ganham prioridade devido ao número de vizinhos não dominados e, novamente, aplica-se o desempate pela ordem alfabética. A Figura 7a também ilustra o impacto da adição de c ao conjunto solução, que posteriormente favorece a seleção do vértice f, o qual, juntamente com h, apresenta o menor grau de dominação, mas se sobressai pelo maior grau de vizinhança. Os valores atribuídos pela função φ_D a f e h, ainda na Figura 7a, são respectivamente, $\frac{2^2}{6-3} = 1{,}33$ e $\frac{2^2}{6-2} = 1$, o que justifica a escolha de f. Por fim, a Figura 7b ainda ilustra h como vértice menos dominado (atente-se que a relação das arestas $\{(a,d),(e,h)$ não é de dominância) logo, h é o vértice mais indicado com uma boa margem.

A Figura 8 apresenta a aplicação da heurística $H\varphi_V$, que inverte a função de avaliação da heurística construtiva abordada acima, priorizando o grau de vizinhança e utilizando o grau de dominação como critério de desempate. A seleção também se inicia pelo vértice c, pois, assim como no cenário anterior, o grau de dominação é igual para todos os vértices, e o grau de vizinhança de c e g destaca-os dos demais. A primeira diferença ocorre na Figura 8b, pois, no lugar do vértice f, que foi selecionado pela heurística anterior, φ_V escolhe o vértice g. Isso se deve à preferência por vértices com maior grau de dominação, já que, conforme mostrado na Figura 8a, g ainda possui um grau de dominação superior aos demais, mesmo após um de seus vizinhos ter sido inserido na solução. Sua avaliação é então dada por $\frac{4^2}{6-1} = 3{,}2$, valor superior aos demais. O algoritmo prossegue selecionando o vértice h, o que não é surpreendente, dado que apenas d e h possuem eles mesmos como vizinhos não dominados, com h sendo escolhido devido ao critério de desempate. É interessante observar que o vértice f encontra-se isolado, pois, mesmo que o vértice g exerça influência sobre ele, as arestas (a,f) e (e,f) não contribuem para sua dominação. Assim, f possui grau de dominação 0, ou seja, nenhum vizinho ainda não dominado. Dessa forma, as Figuras 8c e 8d apresentam, respectivamente, a adição de h e, por fim, do vértice isolado f.

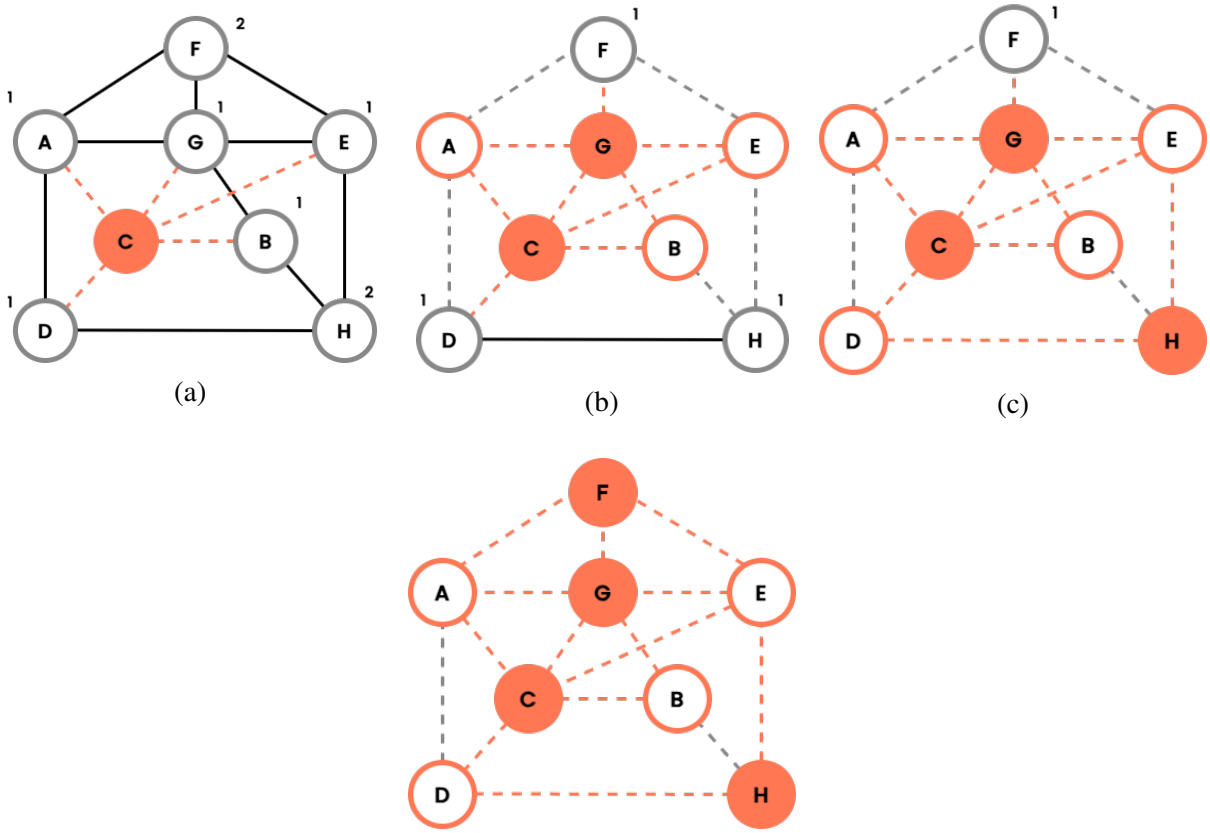
Figura 7 – Aplicação da heurística $H\varphi_D$ ao grafo G_4.



(a) (b) (c)

Fonte: Adaptado de (RANGEL, 2023)

Figura 8 – Aplicação da heurística $H\varphi_V$ ao grafo G_4.



(a) (b) (c)

(d)

Fonte: Adaptado de (RANGEL, 2023)

4.1.4 Heurística Aleatória

Por fim, foi implementada uma heurística de construção aleatória, cujo objetivo é servir como um *baseline* de comparação para as demais heurísticas, além de atuar como um operador dentro do *framework* algorítmico do ALNS, contribuindo para aumentar a diversidade das soluções geradas. Assim como os demais algoritmos, a heurística aleatória (denominada H_R) inicia com um conjunto solução vazio. Sua principal diferença está no fato de não utilizar qualquer métrica para orientar o processo de seleção dos vértices. O algoritmo seleciona vértices de forma aleatória e os adiciona ao conjunto solução até que todos os vértices do grafo estejam devidamente k-dominados.

4.2 LARGE NEIGHBORHOOD SEARCH

O Capítulo 3 apresentou uma breve introdução ao *Large Neighborhood Search* (LNS). Esta seção tem como objetivo detalhar o funcionamento de cada componente desse algoritmo, bem como descrever sua implementação na abordagem proposta neste trabalho.

Muitas metaheurísticas aplicam procedimentos que realizam melhorias incrementais sobre soluções existentes, aplicando operadores que modificam apenas um ou poucos elementos por vez, como buscas locais. Consequentemente, essas metaheurísticas costumam explorar o espaço de soluções a partir de vizinhanças locais, dado que seus componentes promovem alterações de pequeno porte e graduais. O *Large Neighborhood Search* (LNS), proposto inicialmente por Shaw (1998), rompe com essa limitação ao explorar vizinhanças de grande escala, permitindo movimentos consideráveis no espaço de soluções. A principal vantagem ao considerar a exploração do espaço de soluções a partir de grandes vizinhanças está na capacidade de transitar com maior facilidade entre diferentes regiões promissoras do espaço de soluções, especialmente em problemas altamente restritos. Em contrapartida, a principal desvantagem é a alta complexidade computacional demandada para modificar amplos segmentos de uma solução (MARA et al., 2022).

O princípio fundamental do LNS consiste na aplicação sequencial de dois operadores principais: destruição (*destroy*) e reconstrução (*repair*). Na etapa de destruição, uma parte significativa da solução corrente é removida, de forma controlada, criando uma solução parcial. Em seguida, na etapa de reconstrução, elementos são inseridos, buscando gerar uma nova solução viável, potencialmente melhor do que a anterior. O Algoritmo 2 detalha a estrutura da metaheurística LNS. A linha 3 ilustra o processo de destruição e reparação, removendo uma parte da solução inicial S e, em seguida, aplicando o operador de reconstrução a fim de gerar uma nova solução viável.

Algoritmo 2: Estrutura LNS com critério de Metropolis

Input: Um conjunto solução viável S, β, α, T_0, γ, T_F
$S^* \leftarrow S$, $S' \leftarrow S$;
repeat
  $S' \leftarrow repair(destroy(S))$;
  if *accept(S', S)* **then**
    $S \leftarrow S'$;
  end
  if $f(S') < f(S^*)$ **then**
    $S^* \leftarrow S'$;
  end
  if $rand(0,1) < \exp\left(\frac{f(S')-f(S)}{T}\right)$ **then**
    $S \leftarrow S'$;
  end
  $T \leftarrow \gamma T$;
until *critério de parada atingido*;
return S^*

Este trabalho propõe seis operadores, sendo um de destruição e cinco de reconstrução. O operador de destruição consiste em uma heurística destrutiva aleatória, cujo funcionamento é similar ao da heurística construtiva aleatória apresentada na seção anterior. Nesse processo, uma fração dos vértices presentes na solução atual é removida aleatoriamente, permitindo que o operador de reconstrução explore uma nova vizinhança que, potencialmente, leve a soluções superiores. O grau de destruição é controlado pelo parâmetro α, onde $\alpha \in \mathbb{Q}, 0 \leq \alpha \leq 1$, que define a proporção dos vértices a serem eliminados do conjunto atual. Por sua vez, os operadores de reconstrução correspondem às heurísticas construtivas gulosas apresentadas nas seções anteriores, com pequenas adaptações que incorporam mecanismos estocásticos ao processo de construção.

4.2.1 Técnicas Estocásticas dedicadas aos Operadores de Reconstrução

As seções dedicadas às heurísticas construtivas gulosas destacaram sua natureza determinística. Ao integrá-las ao *framework* do LNS, é necessário incorporar um mecanismo estocástico. Com exceção da heurística aleatória H_R, todas as demais heurísticas implementadas são determinísticas, característica que se mostra ineficaz para a exploração do espaço de soluções. Isso porque, mesmo com o ciclo de remoção e reconstrução aplicado sobre a solução atual, as variações seriam bastante limitadas, dado o comportamento determinístico dos operadores.

Para superar essa limitação e introduzir diversidade às soluções geradas pelas heurísticas, foi adotada a abordagem RCL (*Quality-Based Restricted Candidate List Construction*), origi-

nalmente proposta para a metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) por Feo e Resende (1995). Nesse método, um parâmetro de controle $\beta \in \mathbb{Q}, 0 \leq \beta \leq 1$ é usado para definir um limite de qualidade, a partir do qual é formada uma lista restrita de candidatos (RCL) composta por todos os elementos cujo valor da função de avaliação seja superior ou igual a esse limite. Isso significa que, mesmo que vários candidatos tenham recebido a mesma avaliação, desde que esse valor atenda ao limite, todos estarão incluídos na RCL, sem qualquer preferência adicional. Em seguida, um candidato é selecionado aleatoriamente a partir da RCL, abdicando da gulosidade por maior aleatoriedade no processo de construção da solução.

Seja C o conjunto de vértices candidatos a compor o conjunto solução, e $g(c)$ a função de avaliação que atribui uma medida de qualidade a cada candidato $c \in C$. A construção da RCL pode ser formalizada da seguinte maneira (FEO; RESENDE, 1995). Seja $g_{\min} = \min\{g(c) \mid c \in C\}$ e $g_{\max} = \max\{g(c) \mid c \in C\}$ os menores e maiores valores para a função de avaliação, respectivamente. O limite L para inclusão de candidatos à RCL é dado por $L = g_{\max} - \beta \cdot (g_{\max} - g_{\min})$. Finalmente, a RCL é construída como

$$\text{RCL} = \{c \in C \mid g(c) \geq L\}.$$

Dessa forma, cada heurística construtiva apresentada previamente passa a ser considerada pseudo-gulosa quando incorporada como operador de reconstrução no LNS, fazendo uso do parâmetro β para determinar o relaxamento do limite L e, consequentemente, o quão guloso será o algoritmo.

4.2.2 Critério de Aceitação da Solução

A estrutura padrão da metaheurística LNS adota um critério de aceitação clássico, no qual uma nova solução é aceita para a próxima iteração apenas se apresentar qualidade superior à solução atual. Como será detalhado adiante, o ALNS expande alguns dos conceitos introduzidos no LNS, incorporando novos mecanismos e componentes. Um deles se refere ao critério de aceitação, que passa a utilizar o critério de Metropolis. Esse critério foi integrado ao LNS neste trabalho com o objetivo de facilitar análises comparativas, que serão discutidas no Capítulo 5.

Originalmente proposto por Kirkpatrick, Jr e Vecchi (1983) no contexto da metaheurística *Simulated Annealing*, esse critério permite aceitar soluções piores do que a atual, com base em uma probabilidade controlada por um parâmetro de temperatura. Esse mecanismo tem como objetivo escapar de ótimos locais, favorecendo uma exploração mais ampla do espaço de soluções.

O critério de Metropolis é aplicado conforme as linhas 4 a 9 do Algoritmo 2. Caso a nova solução apresente melhor qualidade que a melhor solução encontrada até o momento, i.e.

$f(S') < f(S^*)$, então S^* é imediatamente atualizada para S'. Caso contrário, S' pode ser aceito como nova solução atual com uma probabilidade definida por

$$e^{-\frac{f(S')-f(S)}{T}}.$$

Essa probabilidade assegura dois pontos fundamentais: (1) soluções superiores à atual são sempre aceitas; e (2) a probabilidade de aceitar uma solução inferior é proporcional à temperatura atual e à diferença de qualidade entre as duas soluções. Sendo assim, T representa a temperatura atual e, quanto maior seu valor, maior a chance de aceitar soluções inferiores. O valor inicial da temperatura T é dado pelo parâmetro T_0. A temperatura decresce ao longo da execução do algoritmo, sendo atualizada a cada iteração pela multiplicação com a taxa de resfriamento (*cooling rate*) γ, onde $0 < \gamma < 1$, até atingir um valor final de temperatura, dado pelo parâmetro T_F. Essa abordagem permite que uma maior diversidade de soluções seja aceita nas iterações iniciais, favorecendo a exploração de diferentes regiões do espaço de busca, ao passo que as últimas iterações focam em soluções de maior qualidade.

4.3 ADAPTATIVE LARGE NEIGHBORHOOD SEARCH

O *Adaptive Large Neighborhood Search* (ALNS) surge como uma evolução direta do LNS, incorporando ao seu funcionamento um mecanismo adaptativo que permite um controle dinâmico das vizinhanças durante a busca. Proposto inicialmente por Ropke e Pisinger (2006), o ALNS mantém a ideia central do LNS, alternando operadores de destruição e reconstrução para explorar grandes vizinhanças. Contudo, o ALNS permite múltiplos operadores de cada tipo, ampliando a variedade de estratégias (e vizinhanças) exploradas. Além disso, a principal motivação por trás do desenvolvimento do ALNS reside na constatação de que a eficácia de um operador de vizinhança pode variar significativamente ao longo da execução, dependendo do estado atual da solução ou da fase da busca Pisinger e Ropke (2007).

O ALNS introduz um sistema de seleção adaptativa que avalia continuamente o desempenho de cada operador ao longo das iterações. Isso torna o algoritmo capaz de priorizar, a partir de probabilidades ligadas aos operadores, aqueles que historicamente demonstram maior eficácia no problema em questão. Além disso, outro mecanismo importante no *framework* do ALNS é o uso do critério de Metropolis como critério de aceitação.

O Algoritmo 3 descreve o funcionamento do ALNS. Os símbolos usados nessa descrição estão detalhados na Tabela 2. Assim como o LNS, o ciclo do algoritmo ALNS consiste na aplicação iterativa de dois operadores de destruição e reparação. A cada iteração, um único par dos operadores de destruição e reconstrução é selecionado a partir de uma roleta proporcional aos

Algoritmo 3: Funcionamento do algoritmo ALNS.

Input: Um conjunto solução viável S, Ω^-, Ω^+, η_s, T_0, γ, T_F, α, β

1 $S^* \leftarrow S$, $S' \leftarrow S$, $i \leftarrow 1$, $T \leftarrow T_0$;
2 Inicialize w^-, w^+, p^-, e p^+;
3 **repeat**
4 Selecione um operador de destruição e um operador de reparação de Ω^- e Ω^+ utilizando *roulette wheel selection*;
5 $S' \leftarrow repair(destroy(S))$;
6 **if** *accept(S′, S)* **then**
7 $S \leftarrow S'$;
8 **end**
9 **if** $f(S') < f(S^*)$ **then**
10 $S^* \leftarrow S'$;
11 **end**
12 **if** $rand(0,1) < \exp\left(\frac{f(S')-f(S)}{T}\right)$ **then**
13 $S \leftarrow S'$;
14 **end**
15 **if** $i = \eta_s$ **then**
16 Atualize w^-, w^+, p^-, e p^+;
17 $i \leftarrow 0$;
18 **end**
19 $T \leftarrow \gamma T$, $i \leftarrow i+1$;
20 **until** *critério de parada atingido*;
21 **return** S^*

pesos atuais de desempenho desses operadores, onde operadores com melhor desempenho histórico possuem maior probabilidade de serem escolhidos. Inicialmente, parte da solução atual é destruída, e em seguida o operador de reparação reconstrói essa solução parcial, produzindo uma nova solução candidata. Essa solução é prontamente aceita caso ela apresente melhoria em relação à atual ($S' \leftarrow S$), e, caso demonstre ser superior à melhor solução encontrada até o momento, ocupa o lugar desta, tornando-se a nova solução incumbente ($S^* \leftarrow S$). Se, porventura, a solução não for superior nem à solução atual nem à incumbente, ela ainda pode ser aceita de acordo com o critério de Metropolis. O processo de aceitação da solução descrito acima reflete as linhas 6 a 14 do Algoritmo 3. Ao final de cada iteração, é verificado se o valor η_s foi atingido, o que corresponde ao fim de um segmento, que corresponde a um limite de iterações, e sinaliza que a atualização dos pesos dos operadores deve ser executada (linhas 15 a 18). Por fim, a iteração finaliza com a atualização do valor da temperatura T, de acordo com o grau de resfriamento γ, e com o incremento da iteração, conforme descrito na linha 19 do Algoritmo 3. Todo o processo descrito acima, desde a seleção dos operadores até o incremento do valor de iteração i, é repetido até que o critério de parada seja alcançado, retornando assim o conjunto solução S, que corresponde à melhor solução encontrada.

Tabela 2 – Nomenclatura dos símbolos usados no Algoritmo 3

Símbolo	Descrição
S	Solução viável
S'	Solução recém obtida
S^*	Melhor solução conhecida
Ω^-	Conjunto de operadores de destruição
Ω^+	Conjunto de operadores de reparo
w_i^-	Peso associado ao operador de destruição i
w_i^+	Peso associado ao operador de reconstrução i
p_i^-	Probabilidade associada ao operador de destruição i
p_i^+	Probabilidade associada ao operador de reconstrução i
P^-	Probabilidade dos operadores de destruição
P^+	Probabilidade dos operadores de reconstrução
η_s	Taxa de atualização dos pesos
T_0	Temperatura inicial
T_F	Temperatura final
T	Temperatura atual
γ	Grau de resfriamento

O mecanismo adaptativo é o componente principal do ALNS, portanto será apresentado detalhadamente os processos que conduzem a escolha dos operadores em tempo de execução. A seleção dos operadores de destruição e reconstrução é realizada por meio do método de roleta. As probabilidades de escolha, representadas por $p^- = \{p_1^-, \ldots, p_{|\Omega^-|}^-\}$ e $p^+ = \{p_1^+, \ldots, p_{|\Omega^+|}^+\}$, são calculadas com base nos pesos correspondentes w_i^- e w_i^+, que refletem o desempenho histórico de cada operador, sendo inicialmente iguais para todos e atualizados a cada η_s iterações (MARA et al., 2022). A equação abaixo apresenta o cálculo da probabilidade para um único operador p_i^- e p_i^+, que pode ser descrita como a fração entre o peso do operador Ω_i pelo peso total dos operadores de reconstrução Ω^+ ou destruição Ω^- :

$$p_i^- = \frac{w_i^-}{\sum_{j=1}^{|\Omega^-|} w_j^-} \qquad \text{e} \qquad p_i^+ = \frac{w_i^+}{\sum_{j=1}^{|\Omega^+|} w_j^+}.$$

Os pesos são atualizados ao final de cada segmento de tamanho η_s, definidos conforme Ropke e Pisinger (2006) por

$$w_{i,j+1} = w_{i,j}(1-r) + r \cdot \frac{\pi_i}{\theta_i}$$

onde

- $w^{-}_{i,j}$ é o peso associado ao operador de destruição i no segmento j;
- $w^{+}_{i,j}$ é o peso associado ao operador de reparo i no segmento j;
- r é o fator de reação (*reaction factor*), com $r \in \mathbb{Q}$, $0 < r < 1$;
- π_i é a pontuação do operador i obtida durante o último segmento; e
- θ_i é o número de tentativas de aplicar o operador i durante o último segmento.

Dessa forma, o desempenho de um operador é atribuído à proporção entre a sua pontuação acumulada e o número de vezes em que foi aplicado no segmento j. Esse valor é multiplicado pelo fator de reação r e somado ao peso anterior do operador. O fator de reação exerce influência direta no comportamento do componente adaptativo. Valores próximos de 1 fazem com que o modelo priorize mais os resultados recentes, enquanto valores próximos de 0 atribuem maior importância ao desempenho histórico dos operadores. A partir dos pesos atualizados, o arranjo de probabilidades para todos os operadores de destruição ou reparação em um segmento j pode ser obtido através do seguinte cálculo:

$$P^{-}_{j} = \frac{w^{-}_{i,j}}{\sum_{i \in \Omega^{-}} w^{-}_{i,j}} \qquad \text{e} \qquad P^{+}_{j} = \frac{w^{+}_{i,j}}{\sum_{i \in \Omega^{+}} w^{+}_{i,j}}.$$

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo descreve a avaliação dos algoritmos desenvolvidos neste trabalho. São apresentados os detalhes de implementação, as instâncias adotadas, os parâmetros e sua configuração, e os resultados.

Todos os algoritmos foram implementados na linguagem `Python`. As heurísticas construtivas são baseadas nas ideias de Rangel (2023) e estendidas com pequenas adaptações, conforme o Algoritmo 1. As metaheurísticas seguem os pseudocódigos apresentados no Capítulo 4. Os algoritmos foram estruturados em classes, de modo a fazer bom uso dos princípios de herança e polimorfismo para modularizar os diversos componentes e operadores heurísticos do ALNS. Essa abordagem teve o intuito de suavizar possíveis alterações no código, evitando a necessidade de reformulações estruturais, facilitando também a criação de novos componentes e operadores à medida que novas estratégias eram formuladas. Essa organização também provou-se útil na sistematização dos experimentos.

Adicionalmente, foi desenvolvido um módulo destinado à coleta de métricas durante execuções do ALNS. Esse módulo permite especificar quais métricas de interesse devem ser registradas em diferentes pontos do código, contribuindo para a eficiência na coleta, organização e análise dos resultados experimentais. Além disso, foram desenvolvidos testes unitários com o intuito de assegurar que cada componente do ALNS opere conforme o esperado. Também foram implementados testes de integração simples, envolvendo os operadores de destruição e reconstrução, para garantir que as métricas associadas aos vértices mantenham seus valores corretos durante a execução dos algoritmos. Por fim, a representação dos grafos foi realizada com o suporte da biblioteca `Networkx` (HAGBERG; SCHULT; SWART, 2008).

Todos os métodos propostos neste trabalho foram executados cinco vezes para cada instância, considerando os valores de $k = \{1, 2, 4\}$. O critério de parada adotado para as metaheurísticas LNS e ALNS foi de 5000 iterações. Os experimentos foram conduzidos em um computador equipado com processador Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz, 8 GB de memória RAM, sistema operacional Windows 10 Home (64 bits) e linguagem `Python` na versão 3.13.0.

A coleta e a organização dos dados foram realizadas com o auxílio da biblioteca `Pandas` (MCKINNEY et al., 2010), amplamente utilizada para manipulação e análise de dados em `Python`. Esse processo envolveu a geração de três arquivos ao término de cada experimento: um arquivo `JSON`, contendo as configurações da execução; um `CSV`, com o desempenho geral do algoritmo; e, para o caso do ALNS, um `XLSX` que armazena suas principais métricas, como a progressão das soluções, a evolução dos pesos e o número de chamadas de cada operador.

5.1 INSTÂNCIAS

As instâncias de referência utilizadas para avaliar os métodos propostos neste trabalho foram introduzidas por Corcoran e Gagarin (2021). Esse conjunto é composto por 20 instâncias de pequeno e médio porte, nas quais cada uma representa a rede de ruas reais de diferentes cidades britânicas, modeladas por meio de grafos de acessibilidade. Referente à definição da acessibilidade, Gagarin e Corcoran (2018) adotaram o seguinte critério para a construção das instâncias: uma aresta (u, v) é criada sempre que a distância do caminho mais curto entre os vértices u e v for inferior a t metros. O número de vértices e arestas de cada instância pode ser conferido na Tabela 3.

Tabela 3 – Número de Vértices e Arestas para 20 Instâncias Selecionadas

Instância	Dimensões do grafo	
	$\|V\|$	$\|E\|$
Bath	910	18560
Belfast	1700	62617
Brighton	976	35012
Bristol	1569	47522
Cardiff	1127	23155
Coventry	1175	26689
Exeter	1250	31997
Glasgow	1137	24323
Leeds	1647	56511
Leicester	1531	48219
Liverpool	1273	42564
Manchester	1991	77286
Newcastle	1109	26614
Nottingham	1739	51595
Oxford	479	8396
Plymouth	1122	35070
Sheffield	1582	50534
Southampton	796	19942
Sunderland	1346	42013
York	1044	23774

Com o intuito de fazer uma análise comparativa direta com os métodos da literatura, este trabalho adotou os mesmos valores de k previamente explorados nas instâncias apresentadas na Tabela 3 (1, 2 e 4). A resolução de uma instância para o problema da k-dominação varia significativamente de acordo com o requisito de dominação. Mesmo ao considerar o mesmo grafo, diferentes valores de k resultam em desafios computacionais distintos. Portanto, os três

valores de k, combinados com os 20 grafos apresentados na Tabela 3, totalizam 60 instâncias distintas para o MkDSP.

Os valores de *benchmark* aos quais os algoritmos desenvolvidos neste trabalho serão comparados correspondem ao desempenho da metaheurística VNS, proposta por Predojević, Kartelj e Djukanović (2023), que representa o estado da arte em termos dos melhores resultados obtidos para as 20 instâncias selecionadas nos valores de k previamente indicados. As instâncias foram obtidas em Predojević, Kartelj e Djukanović (2023) e estão disponíveis no repositório [1], assim como os demais algoritmos e artefatos deste trabalho.

5.2 CONFIGURAÇÃO DE PARÂMETROS

Como apresentado na Seção 2.2, é comum que vários algoritmos possuam uma série de parâmetros configuráveis. Isso ocorre pelas necessidades implícitas em seus componentes, que dependem de certos valores fundamentais, capazes de ditar seu comportamento e cujos valores devem ser definidos antes da execução. Um exemplo simples é o parâmetro β, utilizado nas heurísticas pseudo-gulosas, o qual determina seu grau de gulosidade.

Para que o processo de configuração automática por meio do `irace` seja realizado, é necessário estabelecer três componentes fundamentais: o conjunto de instâncias de treinamento, que são as amostras sobre as quais os parâmetros serão ajustados; o espaço de parâmetros, onde estão definidos os parâmetros, seus tipos (numéricos, categóricos ou ordinais) e domínios; e o cenário de configuração, que inclui as definições gerais do experimento (LÓPEZ-IBÁÑEZ et al., 2016).

Foram selecionadas como instâncias de treinamento 10 das 20 instâncias apresentadas na Tabela 3, sendo elas *Bath*, *Brighton*, *Cardiff*, *Glasgow*, *Leicester*, *Manchester*, *Nottingham*, *Plymouth*, *Southampton* e *York*. Vale lembrar que o MkDSP depende do valor de k. Logo, para cada instância é atribuído um valor k correspondente a cada um dos valores determinados para os experimentos, totalizando 30 instâncias de treinamento. A configuração de parâmetros foi feita com o `irace` na versão 4.2.0 com 3000 como valor de *budget*.

A Tabela 4 apresenta o espaço de parâmetros configuráveis para o algoritmo ALNS, que será o único algoritmo configurado dentre os métodos propostos. A coluna à esquerda nomeia os parâmetros, a coluna *tipo* apresenta os tipos, a coluna *domínio* apresenta o intervalo de valores de cada parâmetro, e a coluna *restrição* define regras de definição de valores entre os parâmetros.

O parâmetro β define o grau de gulosidade para os operadores de construção. O parâmetro α define o grau de destruição, determinando assim a quantidade de vértices que serão removidos

[1] <https://github.com/AhPerai/MkDSP_ALNS>

Tabela 4 – Parâmetros configuráveis e seus respectivos domínios.

Parâmetro	**Tipo**	**Domínio**	**Restrição**
β	Racional	(0.0, 1.0)	
α	Racional	(0.1, 0.9)	
T_0	Inteiro	(1, 100)	
T_F	Racional	(0.1, 10)	$T_F \leq T_0$
γ	Racional	(0.80, 0.9999)	
η_s	Inteiro	(10, 1000)	
r	Racional	(0.0, 1.0)	
R^*	Inteiro	(0, 100)	
R'	Inteiro	(0, 100)	$R' \leq R^*$
$R^\vee$	Inteiro	(0, 100)	$R^\vee \leq R^*$

pelos operadores de destruição. Os parâmetros T_0, T_F e γ estipulam os valores usados no critério de Metropolis, temperatura inicial, temperatura final e grau de resfriamento, respectivamente. Os demais parâmetros se referem ao componente *Segmented Roulette Wheel* com η_s como o tamanho do segmento r representando o fator de reação. Além destes, também devem ser configurados os valores de recompensa para um operador de acordo com a solução obtida, R^*, R^+ e $R^\vee$ representam a recompensa ao alcançar um ótimo resultado global, um resultado superior à solução incumbente e, por fim, um resultado aceito pelo critério de Metropolis.

A metaheurística ALNS recebeu cinco configurações de parâmetros distintas para os experimentos que serão abordados a seguir, sendo a primeira manual e outras quatro configurações automáticas produzidas pelo `irace`. A primeira configuração, denominada $Conf_M$, é uma configuração manual aplicada nos experimentos iniciais. A segunda provém de uma configuração automática que usou todas as instâncias de treinamento, i.e. com os três valores distintos de k. Essa configuração é usada para obter um algoritmo genérico para qualquer valor de k e é chamada $Conf_K$. Por outro lado, as outras três configurações foram produzidas pelo `irace`, mas usando as instâncias com um valor específico de k no treinamento. Dessa forma, essas configurações especializam o algoritmo para cada valor de k, e são chamadas $Conf_{\mathrm{K1}}$, $Conf_{\mathrm{K2}}$ e $Conf_{\mathrm{K4}}$ valores 1, 2 e 4, respectivamente.

A razão para trazer uma configuração distinta para cada valor diferente de k se dá por duas motivações: (1) explorar as diferentes configurações de parâmetros e verificar como cada uma se especializa a um valor de k; e (2) averiguar se essas configurações apresentam desempenhos superiores para o valor alvo de k, em comparação com as demais. Essas configurações extras contribuem do ponto de vista experimental ao trabalho, visto que "uma implementação específica é definida não só por decisões estruturais, [...] mas também, especialmente no que

Tabela 5 – Configurações de parâmetros distintas aplicadas ao ALNS

Parâmetro	$Conf_M$	$Conf_K$	$Conf_{\mathrm{K1}}$	$Conf_{\mathrm{K2}}$	$Conf_{\mathrm{K4}}$
α	0,15	0,0057	0,0183	0,0093	0,0044
β	0,5	0,2666	0,3830	0,2967	0,8888
T_0	25	62	81	58	20
T_F	1	0,1187	0,2106	0,1187	7,7517
γ	0,9975	0,9434	0,8020	0,8729	0,9926
η_s	25	303	223	655	591
r	0,5	0,8744	0,5192	0,8033	0,5363
R^*	33	72	58	45	62
R'	16	12	45	19	13
$R^\vee$	9	32	50	18	7

diz respeito à metaheurística, pela configuração específica considerada" (BIRATTARI; KACPRZYK, 2009, p.176, tradução nossa). A Tabela 5 apresenta as cinco diferentes configurações e os valores dos parâmetros definidos por cada uma.

5.3 RESULTADOS E DISCUSSÃO

Os resultados de cada série de comparações estão dispostos em três tabelas, uma para cada valor de $k = \{1,2,4\}$. As métricas, assim como o arranjo das colunas apresentadas para cada experimento, podem variar de acordo com a natureza da abordagem analisada e serão especificadas em cada subseção. Ainda assim, há elementos semelhantes em todas as tabelas. A primeira coluna é reservada para as instâncias, enquanto o cabeçalho indica o nome dos algoritmos avaliados, seguido das métricas de avaliação. Há um conjunto de métricas comuns presente na maioria das tabelas, representado pelos símbolos $|S^*|$, que indica o melhor resultado encontrado entre todas as execuções, $\overline{|S|}$, correspondente à média dos resultados obtidos, $\sigma(|S|)$, o desvio padrão, e $\bar{t}$, o tempo médio de execução. Por fim, os melhores valores são destacados em negrito, a fim de facilitar a análise e a leitura dos resultados. Apenas as colunas $|S^*|$ e $\overline{|S|}$ receberam esse destaque, indicando o melhor valor absoluto encontrado e a melhor média entre os métodos avaliados.

5.3.1 Avaliação das Heurísticas Gulosas Determinísticas

As Tabelas 6, 7 e 8 apresentam os resultados das heurísticas de construção baseadas no grau de vizinhança, grau de dominação, híbrida com foco no grau de dominação, híbrida com

Tabela 6 – Resultados das heurísticas construtivas determinísticas para $k = 1$.

Instância	H_V		H_D		$H\varphi_D$		$H\varphi_V$		H_R	
	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$
Bath	**50**	0,1	63	0,1	**50**	0,1	**50**	0,1	63,8	0,1
Belfast	**60**	0,2	71	0,2	**60**	0,2	**60**	0,3	78,8	0,2
Brighton	**30**	0,1	39	0,1	**30**	0,1	**30**	0,1	37,8	0,1
Bristol	**61**	0,2	74	0,2	**61**	0,2	**61**	0,2	75,6	0,2
Cardiff	**60**	0,1	72	0,1	**60**	0,1	**60**	0,1	75,6	0,1
Coventry	**56**	0,1	83	0,1	**56**	0,1	**56**	0,1	73,2	0,1
Exeter	**58**	0,1	77	0,1	**58**	0,1	**58**	0,1	76,6	0,1
Glasgow	**68**	0,1	90	0,1	**68**	0,1	**68**	0,1	83,2	0,1
Leeds	**62**	0,3	74	0,2	**62**	0,3	**62**	0,3	74,6	0,2
Leicester	**56**	0,2	73	0,1	**56**	0,2	**56**	0,2	78,4	0,2
Liverpool	**40**	0,1	63	0,1	**40**	0,1	**40**	0,2	59,4	0,1
Manchester	**57**	0,3	77	0,2	**57**	0,3	**57**	0,3	76,8	0,3
Newcastle	**59**	0,2	76	0,1	**59**	0,1	**59**	0,1	75,6	0,1
Nottingham	**62**	0,3	74	0,2	**62**	0,2	**62**	0,2	85,8	0,2
Oxford	**31**	0,0	38	0,0	**31**	0,0	**31**	0,0	39,4	0,1
Plymouth	**46**	0,1	53	0,1	**46**	0,1	**46**	0,1	60,4	0,1
Sheffield	**65**	0,3	79	0,3	**65**	0,3	**65**	0,3	82,6	0,2
Southampton	**33**	0,1	42	0,1	**33**	0,1	**33**	0,1	47,0	0,1
Sunderland	**49**	0,1	55	0,1	**49**	0,1	**49**	0,1	62,8	0,1
York	**47**	0,1	70	0,1	**47**	0,1	**47**	0,1	70,2	0,1

foco no grau de vizinhança e aleatória, nessa ordem. Devido à natureza determinística das quatro primeiras heurísticas, uma única execução é suficiente para avaliar seu desempenho. Portanto, apenas as métricas de melhor resultado e tempo médio de execução são reportadas. Para a heurística aleatória, foram realizadas dez execuções, nas quais foram coletadas a média dos resultados e o tempo médio de execução. O melhor resultado não é exibido na tabela, visto que o intuito principal dessa heurística é servir como um *baseline*, permitindo avaliar a efetividade das demais abordagens.

A Tabela 6 corresponde ao primeiro experimento elaborado e traz informações relevantes, sobretudo no que diz respeito ao comportamento das heurísticas H_V, H_{φ_D} e H_{φ_V} quando $k = 1$, isto é, quando o MkDSP se equivale ao problema clássico de dominação. Observa-se que todas essas heurísticas empatam em todas as instâncias, apresentando os melhores resultados para este primeiro experimento, apesar de empregarem funções de avaliação baseadas em atributos distintos do problema. As heurísticas φ_D e φ_V definem as funções de avaliação das heurísticas

Tabela 7 – Resultados das heurísticas construtivas determinísticas para $k = 2$.

Instância	H_V		H_D		$H\varphi_D$		$H\varphi_V$		H_R	
	$\lvert S^*\rvert$	$\bar{t}$	$\lvert S^*\rvert$	$\bar{t}$	$\lvert S^*\rvert$	$\bar{t}$	$\lvert S^*\rvert$	$\bar{t}$	$\lvert S^*\rvert$	$\bar{t}$
Bath	97	0,1	125	0,1	101	0,1	**95**	0,1	122,6	0,1
Belfast	114	0,2	141	0,2	118	0,2	**113**	0,2	143,8	0,3
Brighton	**58**	0,1	74	0,1	61	0,1	**58**	0,1	74,0	0,1
Bristol	114	0,2	136	0,2	**112**	0,2	**112**	0,2	142,2	0,2
Cardiff	113	0,1	145	0,1	123	0,1	**112**	0,1	142,8	0,1
Coventry	106	0,1	147	0,1	107	0,1	**104**	0,1	139,8	0,1
Exeter	111	0,1	140	0,1	123	0,1	**109**	0,1	146,8	0,1
Glasgow	130	0,1	158	0,1	134	0,1	**124**	0,1	154,2	0,1
Leeds	121	0,3	145	0,2	130	0,3	**118**	0,3	145,8	0,2
Leicester	108	0,2	143	0,2	120	0,2	**106**	0,2	147,6	0,2
Liverpool	85	0,1	120	0,1	**84**	0,2	**84**	0,2	108,0	0,1
Manchester	**103**	0,3	147	0,3	118	0,3	**103**	0,3	143,4	0,3
Newcastle	109	0,1	145	0,1	112	0,1	**107**	0,1	140,8	0,1
Nottingham	115	0,2	153	0,2	115	0,2	**113**	0,2	151,4	0,2
Oxford	62	0,0	75	0,0	**59**	0,0	60	0,0	75,8	0,0
Plymouth	88	0,1	108	0,1	93	0,1	**87**	0,1	111,8	0,1
Sheffield	114	0,3	153	0,2	126	0,3	**113**	0,3	153,0	0,2
Southampton	66	0,1	83	0,1	72	0,1	**63**	0,1	86,2	0,1
Sunderland	96	0,2	112	0,1	**94**	0,2	96	0,2	118,6	0,2
York	**91**	0,1	128	0,1	99	0,1	**91**	0,1	127,0	0,1

H_{φ_D} e H_{φ_V}. A primeira prioriza vértices menos dominados, utilizando o grau de vizinhança como critério de desempate, com a segunda formulação invertendo a estratégia. Contudo, no cenário em que o requisito de dominação é apenas um vértice ($k = 1$), o critério de desempate de H_{φ_V} torna-se irrelevante, uma vez que não há variação no grau de dominação, isto é, todos os vértices ainda não dominados terão o mesmo valor $k = 1$ ou obrigatoriamente estarão dominados. De maneira similar, em H_{φ_D}, a priorização de vértices menos dominados também perde sentido, pois todos os vértices não dominados possuem custo uniforme (igual a 1). Uma vez que um vértice seja dominado, não fará mais parte do conjunto e sua função de avaliação se vê reduzida apenas ao seu critério de desempate. Portanto, essas heurísticas se tornam equivalentes à H_V, o que explica os resultados iguais.

As Tabelas 7 e 8 apresentam resultados semelhantes entre si e permitem identificar padrões relevantes no desempenho das heurísticas avaliadas. Para $k = 2$, a heurística H_{φ_V} se destaca na maioria das instâncias, embora H_{φ_D} e H_V eventualmente alcancem resultados supe-

Tabela 8 – Resultados das heurísticas construtivas determinísticas para $k = 4$.

Instância	H_V		H_D		$H\varphi_D$		$H\varphi_V$		H_R	
	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$	$\|S^*\|$	$\bar{t}$
Bath	181	0,1	219	0,1	192	0,1	**173**	0,1	214,0	0,1
Belfast	201	0,2	266	0,2	207	0,3	**198**	0,3	250,8	0,2
Brighton	105	0,1	132	0,1	119	0,1	**103**	0,1	131,4	0,1
Bristol	199	0,2	256	0,2	209	0,2	**194**	0,2	257,6	0,2
Cardiff	201	0,1	251	0,1	223	0,1	**190**	0,1	250,4	0,1
Coventry	196	0,1	255	0,1	212	0,2	**194**	0,1	249,6	0,1
Exeter	206	0,1	258	0,1	223	0,2	**201**	0,2	261,6	0,1
Glasgow	231	0,1	267	0,1	239	0,1	**219**	0,2	271,2	0,1
Leeds	223	0,3	274	0,2	242	0,4	**213**	0,3	260,6	0,2
Leicester	204	0,2	265	0,2	234	0,2	**193**	0,2	269,2	0,2
Liverpool	159	0,2	211	0,1	161	0,2	**156**	0,2	193,6	0,2
Manchester	201	0,3	265	0,3	228	0,4	**199**	0,4	261,0	0,3
Newcastle	203	0,1	246	0,1	206	0,2	**197**	0,1	242,4	0,1
Nottingham	217	0,2	289	0,2	228	0,3	**209**	0,2	278,8	0,2
Oxford	113	0,0	134	0,0	110	0,0	**107**	0,0	132,0	0,1
Plymouth	164	0,1	195	0,1	176	0,2	**157**	0,1	194,0	0,1
Sheffield	211	0,3	276	0,2	235	0,4	**205**	0,3	272,6	0,2
Southampton	121	0,1	161	0,1	139	0,1	**117**	0,1	158,0	0,1
Sunderland	183	0,2	221	0,2	193	0,2	**178**	0,2	222,4	0,2
York	168	0,1	236	0,1	181	0,2	**162**	0,1	226,6	0,1

riores. Esse comportamento, no entanto, não se repete para $k = 4$, onde há uma dominância total de H_{φ_V} em todas as instâncias. Em ambos os cenários, H_V se destaca como a segunda melhor heurística, seguida de H_{φ_D}. A heurística H_D apresenta o pior desempenho entre as quatro heurísticas determinísticas, com uma margem considerável. De fato, a heurística aleatória H_R supera H_D em mais de uma instância, de modo que os resultados sugerem uma vantagem para o primeiro conforme o valor de k aumenta, superando o segundo método por completo quando $k = 4$, onde H_R oferece melhores resultados para a maioria das instâncias. Considerando que H_R foi projetada para atuar como *baseline*, o fato de superar uma heurística determinística revela um desempenho particularmente insatisfatório por parte de H_D. As possíveis causas para esse comportamento serão discutidas nas seções seguintes.

5.3.2 Avaliação do LNS

Os próximos experimentos têm como objetivo comparar cada combinação entre os operadores de reconstrução e o operador de destruição aleatório, visando explorar dois pontos centrais. O primeiro refere-se ao desempenho das heurísticas construtivas, agora integradas ao LNS, avaliando se os resultados observados na seção anterior se mantêm ou se a introdução do mecanismo estocástico, associado à destruição parcial da solução, altera a qualidade de cada heurística. O segundo aspecto busca fornecer dados para experimentos futuros. Em virtude do mecanismo adaptativo do ALNS, torna-se relevante investigar se existe correlação entre o melhor operador em um contexto isolado (LNS) e aquele que tende a ser mais favorecido durante a execução do ALNS. As Tabelas 9, 10 e 11 apresentam cada algoritmo LNS usando as diferentes heurísticas construtivas pseudo-gulosas, identificadas no nome da metaheurística. Por exemplo, o algoritmo $LNSH_V$ representa a metaheurística LNS utilizando o operador de reconstrução baseado na heurística H_V. Todos os métodos foram avaliados com base nas métricas $|S^*|$ (melhor solução), $\overline{|S|}$ (média das soluções), $\sigma(|S|)$ (desvio padrão) e $\bar{t}$ (tempo médio de execução).

Foram realizadas três execuções para cada método. É importante destacar que a solução inicial viável S foi construída, em todos os casos, pela heurística aleatória, servindo como ponto de partida para o ciclo de destruição e reconstrução. Por fim, cabe mencionar que o LNS também possui alguns parâmetros configuráveis. No entanto, como o foco deste trabalho reside no estudo do ALNS, optou-se por adotar uma configuração manual para o LNS, utilizando os valores definidos na configuração $Conf_M$ da Tabela 5. Essa escolha visa permitir a comparação entre ALNS e as demais metaheurísticas LNS. Naturalmente, foram considerados apenas os parâmetros presentes em $Conf_M$ que são aplicáveis ao LNS, sendo β, α, T_0, T_F e γ.

O $LNSH\varphi_V$ destacou-se como o operador de reconstrução superior em todas as instâncias, para todos os valores de k. No caso de $k = 1$, o segundo melhor desempenho foi do $LNSH\varphi_D$, seguido pelo $LNSH_V$, com pouca diferença entre as três abordagens, dada sua similaridade para essa variante do problema, conforme discutido na seção anterior. Para os demais valores de k, o $LNSH_V$ manteve-se consistentemente na segunda posição, indicando que funções de avaliação baseadas no grau de vizinhança apresentaram desempenho superior em todos os casos. Por outro lado, o operador $LNSH\varphi_D$ não se mostrou estritamente superior ao $LNSH_D$, que já havia apresentado resultados insatisfatórios nos primeiros experimentos. Não há uma vantagem evidente entre os algoritmos quando comparados diretamente, uma vez que ambos apresentaram, em algum momento, resultados superiores ao outro ao se analisar a coluna $|S^*|$, embora $LNSH_{\varphi_D}$ tenha demonstrado um desempenho mais consistente em termos de média entre as execuções.

Tabela 9 – Comparação entre diferentes operadores de reparação na meta-heurística LNS em $k = 1$.

Instância	$LNSH_V$				$LNSH_D$				$LNSH\varphi_D$				$LNSH\varphi_V$				$LNSH_R$			
	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$
Bath	48	48,7	0,5	52,5	59	59,7	0,9	15,2	47	48,0	0,8	68,6	**45**	**46,0**	0,8	72,6	59	59,3	0,5	9,9
Belfast	61	61,7	0,5	159,2	72	73,0	1,4	26,5	60	60,7	0,9	170,3	**57**	**58,7**	1,3	173,5	71	72,0	0,8	17,2
Brighton	29	29,3	0,5	82,6	32	33,0	0,8	14,1	28	28,7	0,5	86,3	**26**	**27,0**	0,8	88,5	34	34,3	0,5	8,7
Bristol	54	55,0	0,8	126,0	71	72,0	0,8	24,0	54	55,0	0,8	142,5	**51**	**52,3**	1,3	144,3	71	71,7	0,5	20,5
Cardiff	58	58,3	0,5	81,5	72	72,7	0,9	18,5	58	59,3	0,9	83,6	**54**	**55,3**	0,9	82,5	73	73,7	0,5	12,3
Coventry	52	53,7	1,3	77,3	70	71,0	0,8	19,1	52	53,3	0,9	87,0	**50**	**50,7**	0,5	90,5	69	69,7	0,5	13,0
Exeter	57	58,3	0,9	89,0	68	71,7	2,6	20,5	54	56,0	2,2	92,1	**54**	**54,7**	0,5	92,4	74	74,0	0,0	13,2
Glasgow	68	69,0	0,8	73,7	83	83,7	0,5	20,6	66	67,3	1,3	91,2	**63**	**64,0**	0,8	94,8	80	80,7	0,5	12,9
Leeds	60	61,0	0,8	138,0	73	73,7	0,9	24,9	58	59,3	1,3	144,3	**56**	**57,3**	0,9	149,4	74	74,7	0,5	16,5
Leicester	56	57,7	1,3	115,7	73	73,7	0,5	24,4	56	57,3	0,9	137,8	**54**	**54,7**	0,5	134,6	74	74,3	0,5	15,5
Liverpool	42	43,0	0,8	107,9	52	53,0	0,8	19,7	42	42,3	0,5	101,7	**38**	**39,7**	1,3	103,2	52	52,7	0,9	12,4
Manchester	59	59,7	0,5	162,2	70	71,3	1,3	32,9	56	56,7	0,5	171,6	**55**	**55,7**	0,5	183,0	70	71,0	0,8	18,3
Newcastle	59	59,7	0,5	70,6	68	71,0	2,2	19,2	58	58,3	0,5	83,1	**56**	**56,7**	0,5	83,2	68	72,0	2,8	12,0
Nottingham	60	61,7	1,3	131,0	78	78,7	0,9	31,7	60	60,3	0,5	138,7	**57**	**58,0**	0,8	139,6	75	77,7	1,9	17,6
Oxford	30	30,0	0,0	27,7	36	36,7	0,5	8,2	29	29,3	0,5	29,4	**28**	**28,0**	0,0	30,2	36	36,3	0,5	5,3
Plymouth	44	44,7	0,9	79,2	53	54,0	0,8	17,1	43	43,3	0,5	81,9	**42**	**42,3**	0,5	87,7	54	54,3	0,5	13,1
Sheffield	63	63,0	0,0	114,1	76	77,0	0,8	26,3	62	63,0	0,8	122,2	**59**	**59,3**	0,5	125,3	77	78,0	0,8	17,6
Southampton	33	33,7	0,5	51,5	40	42,0	1,4	11,3	32	32,3	0,5	55,4	**30**	**30,7**	0,5	57,0	39	40,0	0,8	10,7
Sunderland	50	50,3	0,5	96,0	59	59,7	0,9	21,4	49	49,7	0,9	98,7	**47**	**47,7**	0,5	104,3	57	58,3	0,9	13,6
York	48	49,0	0,8	66,2	64	66,3	1,7	17,8	48	49,0	0,8	70,2	**45**	**45,7**	0,5	73,4	62	65,3	2,4	11,1

Tabela 10 – Comparação entre diferentes operadores de reparação na meta-heurística LNS em $k = 2$.

Instância	$LNSH_V$				$LNSH_D$				$LNSH\varphi_D$				$LNSH\varphi_V$				$LNSH_R$			
	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$
Bath	96	97,3	1,3	81,7	107	116,3	6,8	38,3	109	111,0	1,6	106,0	**93**	**93,3**	0,5	96,6	115	117,7	1,9	20,5
Belfast	121	122,7	1,3	180,8	133	138,3	3,9	57,3	139	140,3	1,3	230,7	**117**	**117,7**	0,9	214,6	136	138,7	3,1	35,3
Brighton	58	58,0	0,0	103,2	69	71,3	1,7	21,2	63	65,3	1,7	111,1	**52**	**54,3**	1,7	112,5	69	69,3	0,5	15,1
Bristol	113	113,7	0,5	158,6	130	130,7	0,9	52,3	136	137,7	1,7	197,8	**104**	**107,0**	2,2	179,1	130	135,0	5,1	30,3
Cardiff	117	117,7	0,9	100,1	139	141,0	1,6	39,4	134	134,7	0,5	138,2	**113**	**114,0**	0,8	118,5	137	140,7	2,6	25,0
Coventry	108	109,0	0,8	109,7	137	140,3	2,5	42,1	132	133,7	1,3	141,4	**100**	**103,3**	2,4	129,4	133	136,7	4,5	25,8
Exeter	117	118,7	1,7	119,5	138	143,7	4,9	43,3	137	138,3	1,3	172,6	**114**	**114,0**	0,0	154,0	138	141,0	4,2	31,1
Glasgow	131	131,7	0,5	103,2	154	155,7	1,3	40,5	144	144,7	0,5	158,8	**126**	**126,3**	0,5	127,8	148	149,0	1,4	24,7
Leeds	120	120,7	0,5	175,7	141	145,3	4,8	50,7	140	142,3	2,1	251,0	**112**	**115,3**	2,4	234,6	139	141,7	3,1	33,0
Leicester	119	121,3	2,1	168,9	146	147,3	1,9	53,6	142	142,7	0,9	299,7	**114**	**115,0**	1,4	249,1	140	146,3	4,5	30,0
Liverpool	85	85,3	0,5	150,0	108	109,3	1,3	46,1	101	104,0	2,2	182,2	**82**	**82,3**	0,5	206,6	104	106,7	2,1	24,4
Manchester	117	117,3	0,5	271,7	144	146,3	2,6	72,5	135	137,3	2,6	349,1	**111**	**112,3**	0,9	324,3	138	139,0	0,8	34,3
Newcastle	116	116,7	0,5	159,3	138	144,0	4,2	50,8	132	133,7	1,7	162,9	**111**	**111,7**	0,9	150,1	126	135,0	6,4	21,8
Nottingham	128	128,7	0,5	239,0	151	154,0	2,9	77,7	153	154,7	1,7	312,9	**119**	**119,3**	0,5	269,6	154	156,3	2,1	33,6
Oxford	58	59,3	1,3	42,4	74	74,7	0,5	16,7	67	68,3	0,9	54,5	**56**	**56,3**	0,5	53,9	68	70,7	2,1	8,4
Plymouth	89	89,7	0,5	131,6	104	107,0	2,5	38,7	103	103,7	0,5	179,2	**85**	**85,3**	0,5	170,1	100	107,0	5,0	19,4
Sheffield	126	127,0	0,8	200,4	148	150,3	2,1	71,4	151	151,0	0,0	289,4	**121**	**122,0**	0,8	247,7	148	149,0	0,8	28,3
Southampton	69	70,0	0,8	83,7	84	85,0	1,4	27,2	79	80,0	0,8	115,1	**65**	**65,7**	0,5	124,5	83	84,3	1,3	13,1
Sunderland	103	103,7	0,5	168,0	121	122,7	1,3	50,7	114	115,0	0,8	196,8	**96**	**98,0**	1,6	208,9	118	119,0	0,8	22,9
York	100	101,3	0,9	117,5	121	127,3	4,5	44,4	119	122,0	2,2	157,8	**96**	**96,3**	0,5	130,7	127	128,0	1,4	20,0

Tabela 11 – Comparação entre diferentes operadores de reparação na meta-heurística LNS em $k = 4$.

Instância	$LNSH_V$				$LNSH_D$				$LNSH\varphi_D$				$LNSH\varphi_V$				$LNSH_R$			
	$\lvert S^*\rvert$	$\overline{\lvert S\rvert}$	$\sigma(\lvert S\rvert)$	$\bar{t}$	$\lvert S^*\rvert$	$\overline{\lvert S\rvert}$	$\sigma(\lvert S\rvert)$	$\bar{t}$	$\lvert S^*\rvert$	$\overline{\lvert S\rvert}$	$\sigma(\lvert S\rvert)$	$\bar{t}$	$\lvert S^*\rvert$	$\overline{\lvert S\rvert}$	$\sigma(\lvert S\rvert)$	$\bar{t}$	$\lvert S^*\rvert$	$\overline{\lvert S\rvert}$	$\sigma(\lvert S\rvert)$	$\bar{t}$
Bath	184	186,0	1,6	148,6	209	215,0	4,9	84,0	204	209,3	4,1	221,5	**177**	**178,7**	1,3	180,7	212	217,3	4,1	31,1
Belfast	225	226,3	1,3	356,9	251	254,3	3,4	136,4	250	253,0	2,9	496,5	**218**	**219,3**	0,9	425,1	250	255,7	4,9	61,1
Brighton	109	109,0	0,0	192,4	129	130,7	2,4	44,5	126	129,0	3,6	203,1	**105**	**105,7**	0,9	209,5	128	131,0	2,5	35,3
Bristol	217	218,0	0,8	263,8	254	254,3	0,5	111,5	242	252,3	7,4	389,7	**208**	**210,3**	1,7	333,8	252	254,7	2,1	66,0
Cardiff	217	219,0	1,6	184,0	248	250,3	1,7	88,3	245	248,0	2,2	318,9	**214**	**214,7**	0,5	224,0	252	254,0	1,6	52,9
Coventry	208	211,7	2,9	199,5	247	248,3	0,9	78,7	237	243,7	6,2	290,8	**206**	**206,7**	0,5	242,1	242	248,7	6,8	55,7
Exeter	222	223,7	1,3	216,7	251	256,3	4,5	81,5	258	259,7	1,3	323,0	**216**	**217,3**	1,3	291,0	258	261,7	3,9	58,5
Glasgow	237	237,7	0,5	193,9	257	262,3	3,8	74,7	257	258,3	1,3	310,1	**231**	**232,3**	0,9	241,9	265	265,3	0,5	53,7
Leeds	227	229,7	1,9	290,7	253	259,3	6,9	100,3	255	258,0	2,2	426,8	**223**	**224,0**	0,8	365,8	255	257,3	1,7	73,4
Leicester	233	233,3	0,5	261,0	266	269,3	2,5	92,7	264	269,3	3,9	360,4	**222**	**224,0**	1,6	337,0	250	262,0	9,1	66,6
Liverpool	166	167,0	0,8	227,8	189	197,3	7,9	65,9	196	199,7	2,9	192,9	**158**	**159,3**	0,9	279,0	190	194,0	5,0	54,5
Manchester	221	223,3	2,1	398,8	260	262,0	2,2	116,0	253	257,3	3,3	377,3	**215**	**216,0**	0,8	458,8	249	257,7	6,3	91,6
Newcastle	213	214,0	0,8	180,0	237	240,3	2,5	67,8	241	245,0	3,3	178,4	**204**	**206,7**	2,5	147,3	238	241,7	2,6	50,9
Nottingham	247	248,3	1,3	317,9	276	282,7	6,2	125,0	278	284,0	7,8	323,8	**234**	**235,3**	1,3	269,3	272	281,7	7,8	75,1
Oxford	113	113,7	0,5	65,4	128	129,0	0,8	23,0	128	128,0	0,0	57,8	**110**	**110,7**	0,5	49,6	129	129,0	0,0	17,1
Plymouth	170	171,7	1,3	182,2	191	192,0	0,8	56,6	190	192,0	1,6	172,1	**163**	**164,3**	1,3	147,4	195	198,3	2,4	44,8
Sheffield	235	238,7	2,9	284,0	264	266,0	2,8	103,3	273	274,0	0,8	261,6	**230**	**230,7**	0,5	226,6	263	269,0	4,9	63,8
Southampton	130	131,7	1,3	118,7	147	151,7	4,1	47,8	149	151,7	2,1	96,4	**128**	**128,3**	0,5	89,7	152	154,0	1,4	27,7
Sunderland	195	196,3	1,3	215,1	219	222,0	2,2	79,7	225	226,7	1,3	205,2	**188**	**190,3**	1,7	187,5	219	222,0	2,9	50,6
York	194	194,7	0,5	153,4	221	224,0	2,5	73,9	226	228,0	1,6	150,2	**185**	**187,0**	1,6	133,9	222	228,3	4,9	42,1

Apesar disso, considerando os resultados anteriores em conjunto com os mais recentes, constata-se que H_D se apresenta como a pior heurística, sendo consistentemente superada, inclusive pela heurística aleatória em diversos casos. Isso ocorre principalmente devido à maneira como sua função de avaliação foi concebida. Ao priorizar isoladamente os vértices menos dominados, a exploração de áreas menos dominadas é feita de forma ineficaz, pois a estratégia evita vértices já parcialmente dominados, sem se importar com a região como um todo. Na prática, isso resulta em uma seleção semi-aleatória, que percorre os vértices desde os mais dominados (cuja vizinhança se torna menos atrativa para a métrica) até os menos dominados. Esse efeito é agravado pelo fato de que, na iteração inicial, todos os vértices possuem o mesmo grau de dominação. Por exemplo, quando $k = 4$, a primeira escolha é essencialmente aleatória, seguida por seleções entre vértices com grau de dominação 3, depois 2 e, por fim, 1, caracterizando uma seleção aleatória em etapas, efetivamente reduzida à escolha aleatória quando $k = 1$. Para finalizar, mesmo com a introdução do mecanismo pseudo-guloso RCL, os baixos valores de $k = \{1,2,4\}$ explorados neste trabalho não são suficientes para alterar significativamente esse comportamento. No Capítulo 6 são discutidas alternativas para melhorar a eficácia do grau de dominação.

5.3.3 Avaliação do ALNS vs. LNS vs. Heurísticas

Nesta seção, são comparados os métodos previamente desenvolvidos à metaheurística ALNS sob a configuração $Conf_M$, a configuração inicial manual adotada para comparação direta com os demais métodos, especialmente com o LNS. Foram considerados apenas os melhores resultados obtidos pelas heurísticas construtivas determinísticas e pelas variantes do LNS, com o objetivo de verificar se qualquer uma dessas abordagens superaria o ALNS em alguma instância. Para cada instância, foram realizadas cinco execuções do ALNS, utilizando o critério de parada de 5000 iterações, que assim como o LNS, teve sua solução inicial gerada pela heurística aleatória. Todos os métodos foram avaliados com base nas métricas $|S^*|$ (melhor solução encontrada), $\overline{|S|}$ (média das soluções), $\sigma(|S|)$ (desvio padrão) e $\bar{t}$ (tempo médio de execução).

Os resultados da Tabela 12 ($k = 1$) apresentam um cenário esperado. O $ALNS_{Conf_M}$ se destaca em todas as instâncias, seguido por LNS_{Best} e, por fim, pela heurística H_{Best}. Na Tabela 13, embora o $ALNS_{Conf_M}$ continue a apresentar o melhor desempenho geral, observa-se que, em algumas instâncias, os melhores resultados encontrados pelas heurísticas construtivas superam aqueles obtidos pelo LNS_{Best}. É importante ressaltar que, mesmo quando o operador de reconstrução equivalente a uma heurística foi empregado, a solução inicial no LNS foi gerada pela heurística aleatória. Ou seja, os resultados do LNS não partem do resultado base provido pelos métodos construtivos.

Tabela 12 – Comparação entre ALNS, LNS (melhores resultados) e heurísticas construtivas (melhores resultados) para $k = 1$.

Instância	$ALNS_{Conf_M}$				LNS_{Best}				H_{Best}	
	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	Algo.	$\|S^*\|$	Algo.
Bath	**41**	**43,0**	1,1	39,8	45	46,0	0,8	$LNSH\varphi_V$	50	H_V
Belfast	**50**	**51,6**	1,0	100,9	57	58,7	1,3	$LNSH\varphi_V$	60	H_V
Brighton	**25**	**26,2**	0,8	51,6	26	27,0	0,8	$LNSH\varphi_V$	30	H_V
Bristol	**46**	**47,0**	0,6	89,0	51	52,3	1,3	$LNSH\varphi_V$	61	H_V
Cardiff	**48**	**49,0**	1,3	52,6	54	55,3	0,9	$LNSH\varphi_V$	60	H_V
Coventry	**45**	**46,0**	0,6	60,0	50	50,7	0,5	$LNSH\varphi_V$	56	H_V
Exeter	**48**	**49,2**	1,0	64,5	54	54,7	0,5	$LNSH\varphi_V$	58	H_V
Glasgow	**58**	**59,0**	0,6	54,6	63	64,0	0,8	$LNSH\varphi_V$	68	H_V
Leeds	**51**	**52,6**	0,8	103,2	56	57,3	0,9	$LNSH\varphi_V$	62	H_V
Leicester	**45**	**47,4**	1,4	88,8	54	54,7	0,5	$LNSH\varphi_V$	56	H_V
Liverpool	**35**	**35,8**	0,8	74,8	38	39,7	1,3	$LNSH\varphi_V$	40	H_V
Manchester	**51**	**51,8**	1,0	136,2	55	55,7	0,5	$LNSH\varphi_V$	57	H_V
Newcastle	**52**	**53,0**	0,9	56,1	56	56,7	0,5	$LNSH\varphi_V$	59	H_V
Nottingham	**52**	**53,6**	1,2	108,2	57	58,0	0,8	$LNSH\varphi_V$	62	H_V
Oxford	**26**	**27,4**	0,8	20,2	28	28,0	0,0	$LNSH\varphi_V$	31	H_V
Plymouth	**38**	**39,2**	0,8	94,5	42	42,3	0,5	$LNSH\varphi_V$	46	H_V
Sheffield	**52**	**53,0**	0,6	129,4	59	59,3	0,5	$LNSH\varphi_V$	65	H_V
Southampton	**28**	**28,8**	0,8	51,0	30	30,7	0,5	$LNSH\varphi_V$	33	H_V
Sunderland	**43**	**44,8**	1,5	98,7	47	47,7	0,5	$LNSH\varphi_V$	49	H_V
York	**40**	**42,2**	1,5	67,1	45	45,7	0,5	$LNSH\varphi_V$	47	H_V

Tabela 13 – Comparação entre ALNS, LNS (melhores resultados) e heurísticas construtivas (melhores resultados) para $k = 2$.

Instância	$ALNS_{Conf_M}$				LNS_{Best}				H_{Best}	
	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	Algo.	$\lvert S^* \rvert$	Algo.
Bath	**86**	**87,6**	1,0	79,6	93	93,3	0,5	$LNSH\varphi_V$	95	$H\varphi_V$
Belfast	**102**	**106,8**	2,7	197,3	117	117,7	0,9	$LNSH\varphi_V$	113	$H\varphi_V$
Brighton	**50**	**51,8**	1,0	85,3	52	54,3	1,7	$LNSH\varphi_V$	58	$H\varphi_V$
Bristol	**99**	**100,0**	1,1	147,6	104	107,0	2,2	$LNSH\varphi_V$	112	$H\varphi_V$
Cardiff	**102**	**104,4**	1,4	94,0	113	114,0	0,8	$LNSH\varphi_V$	112	$H\varphi_V$
Coventry	**93**	**95,0**	1,7	100,4	100	103,3	2,4	$LNSH\varphi_V$	104	$H\varphi_V$
Exeter	**100**	**102,6**	1,7	109,8	114	114,0	0,0	$LNSH\varphi_V$	109	$H\varphi_V$
Glasgow	**114**	**118,0**	2,9	94,8	126	126,3	0,5	$LNSH\varphi_V$	124	$H\varphi_V$
Leeds	**104**	**105,8**	1,0	167,6	112	115,3	2,4	$LNSH\varphi_V$	118	$H\varphi_V$
Leicester	**98**	**102,4**	2,6	144,8	114	115,0	1,4	$LNSH\varphi_V$	106	$H\varphi_V$
Liverpool	**74**	**75,6**	1,2	117,7	82	82,3	0,5	$LNSH\varphi_V$	84	$H\varphi_D$
Manchester	**102**	**103,8**	1,3	219,1	111	112,3	0,9	$LNSH\varphi_V$	103	$H\varphi_V$
Newcastle	**103**	**103,6**	0,5	92,9	111	111,7	0,9	$LNSH\varphi_V$	107	$H\varphi_V$
Nottingham	**107**	**109,6**	1,4	177,3	119	119,3	0,5	$LNSH\varphi_V$	113	$H\varphi_V$
Oxford	**54**	**54,2**	0,4	31,9	56	56,3	0,5	$LNSH\varphi_V$	59	$H\varphi_D$
Plymouth	**78**	**79,8**	1,2	102,3	85	85,3	0,5	$LNSH\varphi_V$	87	$H\varphi_V$
Sheffield	**109**	**111,8**	1,9	152,9	121	122,0	0,8	$LNSH\varphi_V$	113	$H\varphi_V$
Southampton	**60**	**60,8**	0,8	61,0	65	65,7	0,5	$LNSH\varphi_V$	63	$H\varphi_V$
Sunderland	**91**	**92,2**	1,0	121,0	96	98,0	1,6	$LNSH\varphi_V$	94	$H\varphi_D$
York	**87**	**87,8**	1,0	84,2	96	96,3	0,5	$LNSH\varphi_V$	91	H_V

Isso indica que, em determinadas instâncias, a exploração conduzida pelo LNS não foi suficientemente eficaz para superar os resultados das heurísticas construtivas determinísticas. É possível que um número maior de iterações pudesse melhorar o desempenho das combinações de LNS em relação às heurísticas. Em contrapartida, é notável que a combinação desses mesmos operadores dentro do *framework* do ALNS demonstrou ser mais eficiente na exploração do espaço de soluções, superando, mesmo sob uma configuração manual, os resultados obtidos pelas abordagens puramente construtivas.

Por fim, os resultados da Tabela 14 indicam que ambas as metaheurísticas enfrentaram maiores dificuldades na exploração do espaço de soluções quando $k = 4$, de modo que os resultados obtidos pelas heurísticas construtivas em parte das instâncias superaram até mesmo o desempenho do ALNS. Consequentemente, o LNS apresentou consistentemente os piores desempenhos, tanto nos melhores valores encontrados quanto nas médias das execuções. Apesar disso, confirma-se que o ALNS é o algoritmo que, de maneira geral, obteve os melhores resulta-

Tabela 14 – Comparação entre ALNS, LNS (melhores resultados) e heurísticas construtivas (melhores resultados) para $k = 4$.

Instância	$ALNS_{Conf_M}$				LNS_{Best}				H_{Best}	
	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	Algo.	$\|S^*\|$	Algo.
Bath	**166**	**169,4**	2,5	119,0	177	178,7	1,3	$LNSH\varphi_V$	173	$H\varphi_V$
Belfast	202	**205,4**	2,2	269,0	218	219,3	0,9	$LNSH\varphi_V$	**198**	$H\varphi_V$
Brighton	**99**	**100,6**	0,8	134,7	105	105,7	0,9	$LNSH\varphi_V$	103	$H\varphi_V$
Bristol	**194**	**196,6**	1,9	239,5	208	210,3	1,7	$LNSH\varphi_V$	**194**	$H\varphi_V$
Cardiff	195	**199,8**	2,5	157,1	214	214,7	0,5	$LNSH\varphi_V$	**190**	$H\varphi_V$
Coventry	**188**	**194,4**	3,8	169,9	206	206,7	0,5	$LNSH\varphi_V$	194	$H\varphi_V$
Exeter	**200**	**203,8**	2,1	184,9	216	217,3	1,3	$LNSH\varphi_V$	201	$H\varphi_V$
Glasgow	**219**	**221,4**	1,9	160,0	231	232,3	0,9	$LNSH\varphi_V$	**219**	$H\varphi_V$
Leeds	**201**	**207,0**	3,2	261,2	223	224,0	0,8	$LNSH\varphi_V$	213	$H\varphi_V$
Leicester	202	**207,8**	2,9	235,2	222	224,0	1,6	$LNSH\varphi_V$	**193**	$H\varphi_V$
Liverpool	**148**	**150,0**	1,4	184,6	158	159,3	0,9	$LNSH\varphi_V$	156	$H\varphi_V$
Manchester	202	**205,0**	2,2	351,3	215	216,0	0,8	$LNSH\varphi_V$	**199**	$H\varphi_V$
Newcastle	**195**	**198,0**	2,1	149,9	204	206,7	2,5	$LNSH\varphi_V$	197	$H\varphi_V$
Nottingham	218	**219,0**	0,9	273,8	234	235,3	1,3	$LNSH\varphi_V$	**209**	$H\varphi_V$
Oxford	**105**	**106,8**	1,3	44,1	110	110,7	0,5	$LNSH\varphi_V$	107	$H\varphi_V$
Plymouth	**156**	**157,0**	0,9	139,1	163	164,3	1,3	$LNSH\varphi_V$	157	$H\varphi_V$
Sheffield	214	**217,6**	2,1	219,9	230	230,7	0,5	$LNSH\varphi_V$	**205**	$H\varphi_V$
Southampton	120	**122,4**	1,6	86,9	128	128,3	0,5	$LNSH\varphi_V$	**117**	$H\varphi_V$
Sunderland	179	**181,2**	1,3	174,1	188	190,3	1,7	$LNSH\varphi_V$	**178**	$H\varphi_V$
York	171	**173,4**	2,2	112,3	185	187,0	1,6	$LNSH\varphi_V$	**162**	$H\varphi_V$

dos ao longo das instâncias avaliadas para os diferentes valores de k. Diante disso, as próximas seções têm por objetivo apresentar os experimentos conduzidos com a configuração automática dos parâmetros, buscando aprimorar o desempenho do ALNS e alcançar os melhores resultados possíveis.

5.3.4 Avaliação do ALNS sob Diferentes Configurações

Esta seção encerra a etapa de exploração e experimentação dos métodos desenvolvidos neste trabalho, com foco específico na metaheurística ALNS. Enquanto os experimentos da Seção 5.3.3 foram conduzidos usando os parâmetros da configuração manual $Conf_M$, os próximos experimentos empregam configurações automáticas obtidas por meio da ferramenta de ajuste de parâmetros `irace`. Cada uma das três tabelas, referentes aos valores $k = \{1,2,4\}$, apresenta os resultados obtidos a partir de três configurações distintas do ALNS, também listadas na Tabela 5. As primeiras oito colunas (desconsiderando a coluna de instâncias) são destinadas às métricas de melhor solução encontrada ($|S^*|$), média das soluções ($\overline{|S|}$), desvio padrão ($\sigma(|S|)$) e tempo médio ($\bar{t}$) das execuções do algoritmo ALNS fazendo uso da configuração manual $ALNS_{Conf_M}$ e configuração automática $ALNS_{Conf_K}$. Esta última corresponde à configuração automática obtida com o uso das instâncias de treinamento para todos os valores de k. Ou seja, é a configuração automática padrão, aplicável a todas as instâncias para $k = \{1,2,4\}$. Já as colunas restantes são reservadas a configurações específicas para algum valor de k (apresentado na tabela), obtidas também com o `irace` utilizando as mesmas instâncias de treinamento, mas restritas a um valor único de k. Cada algoritmo foi executado cinco vezes, nas habituais 5000 iterações.

As diferentes configurações de parâmetros propostas proporcionam não apenas um estudo comparativo, mas também um entendimento mais aprofundado sobre o comportamento do ALNS, seus operadores e componentes. Ao introduzir uma configuração geral e outra específica para cada valor de k, torna-se possível observar tendências nos parâmetros, como a variação do grau de destruição ou dos parâmetros associados ao critério de Metropolis. Essa análise pode revelar limitações ou até prioridades na forma como o espaço de soluções é explorado, especialmente sob diferentes requisitos de dominação. Portanto, as análises subsequentes não focam apenas nos resultados atingidos por cada algoritmo, mas também nas relações entre os parâmetros nos respectivos valores de k de cada tabela.

A Tabela 15 evidencia melhorias significativas em relação aos resultados obtidos com a configuração manual quando comparados às configurações automáticas $Conf_{K1}$ e especialmente $Conf_K$. Observa-se uma clara superioridade desta última, inclusive em relação à configuração específica para o valor de k desta tabela. Esses resultados podem, à primeira vista, parecer con-

Tabela 15 – Resultados de ALNS com diferentes configurações em $k = 1$.

Instância	$ALNS_{ConfM}$				$ALNS_{ConfK}$				$ALNS_{ConfK1}$			
	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$	$\lvert S^* \rvert$	$\overline{\lvert S \rvert}$	$\sigma(\lvert S \rvert)$	$\bar{t}$
Bath	41	43,0	1,1	39,8	**39**	**39,0**	0,0	26,9	**39**	**39,0**	0,0	29,9
Belfast	50	51,6	1,0	100,9	**41**	41,4	0,5	68,6	**41**	**41,2**	0,4	87,7
Brighton	25	26,2	0,8	51,6	**22**	**22,0**	0,0	34,6	**22**	**22,0**	0,0	42,9
Bristol	46	47,0	0,6	89,0	**37**	**37,0**	0,0	51,9	**37**	37,6	0,5	72,9
Cardiff	48	49,0	1,3	52,6	**41**	**41,8**	0,8	26,6	**41**	42,2	0,8	41,2
Coventry	45	46,0	0,6	60,0	**39**	**39,2**	0,4	30,6	**39**	39,8	0,4	47,2
Exeter	48	49,2	1,0	64,5	**40**	**40,6**	0,5	32,7	41	41,0	0,0	45,5
Glasgow	58	59,0	0,6	54,6	**53**	53,6	0,5	33,4	**53**	**53,4**	0,5	39,3
Leeds	51	52,6	0,8	103,2	**43**	**43,0**	0,0	61,9	**43**	43,6	0,8	76,0
Leicester	45	47,4	1,4	88,8	**38**	**38,2**	0,4	50,1	39	39,4	0,5	62,4
Liverpool	35	35,8	0,8	74,8	**29**	**29,0**	0,0	43,5	**29**	29,6	0,8	56,4
Manchester	51	51,8	1,0	136,2	**39**	**40,8**	1,0	80,0	40	41,6	1,0	102,6
Newcastle	52	53,0	0,9	56,1	**47**	47,2	0,4	30,0	**46**	**46,6**	0,5	38,4
Nottingham	52	53,6	1,2	108,2	**44**	**44,0**	0,0	67,7	**44**	44,2	0,4	77,8
Oxford	**26**	27,4	0,8	20,2	**26**	**26,0**	0,0	10,8	**26**	26,2	0,4	13,6
Plymouth	38	39,2	0,8	94,5	**31**	**31,2**	0,4	40,7	**31**	31,8	0,4	48,8
Sheffield	52	53,0	0,6	129,4	**43**	**44,2**	0,8	53,5	44	45,4	0,8	68,8
Southampton	28	28,8	0,8	51,0	**25**	**25,0**	0,0	25,2	**25**	**25,0**	0,0	27,9
Sunderland	43	44,8	1,5	98,7	**38**	**38,0**	0,0	50,0	**38**	38,2	0,4	55,1
York	40	42,2	1,5	67,1	**35**	36,2	0,8	29,5	**35**	**35,6**	0,5	34,6

traintuitivos. Entre as possíveis explicações para o desempenho inferior da configuração $Conf_{K1}$ poderia ser apontado um fenômeno conhecido como *overtuning* (BIRATTARI; KACPRZYK, 2009). *Overtuning*, no contexto da configuração automática de parâmetros para algoritmos heurísticos, caracteriza-se por uma situação em que o processo de ajuste gera uma configuração excessivamente especializada para o conjunto de instâncias utilizado durante o treinamento, comprometendo sua capacidade de generalização para outras instâncias. Ainda assim, não se observa uma correlação clara entre as instâncias não incluídas no conjunto de treinamento, composto por *Bath*, *Brighton*, *Cardiff*, *Glasgow*, *Leicester*, *Manchester*, *Nottingham*, *Plymouth*, *Southampton* e *York*, e um desempenho inferior de $ALNS_{Conf_{K1}}$ em relação a $ALNS_{Conf_K}$. Na prática, é apresentado apenas um único vértice de diferença nas soluções ótimas encontradas por $ALNS_{Conf_K}$ e não obtidas por $ALNS_{Conf_{K1}}$.

Dentre as principais diferenças nos parâmetros apresentados por ambas as configurações, destacam-se: β, T_0, γ, r e, especialmente, os valores de recompensa R^*, R' e $R^\vee$. Primeiramente, o parâmetro β, responsável por indicar o grau de gulosidade, assumiu valores extremamente baixos em todas as configurações, sendo que $Conf_{K1}$ apresentou o valor mais elevado. Esse comportamento é inesperado, uma vez que valores tão baixos tendem a limitar a exploração do espaço de soluções. Este trabalho não oferece uma explicação conclusiva que justifique a seleção desse valor pelo `irace`, embora algumas sugestões sobre o tema sejam discutidas no Capítulo 6. Por outro lado, os demais parâmetros apresentam comportamentos mais alinhados às expectativas.

A configuração $ALNS_{Conf_K}$ adota uma estratégia mais conservadora e controlada, realizando destruições moderadas na solução, utilizando uma temperatura inicial relativamente alta, aliada a uma taxa de resfriamento lenta (γ), o que favorece uma exploração mais ampla nas etapas iniciais da busca. Soma-se a isso um valor elevado para R^*, que privilegia operadores responsáveis pelos melhores resultados, enquanto R' é significativamente reduzido e $R^\vee$ assume um valor mais expressivo. Este arranjo sugere uma estratégia que prioriza a capacidade de escapar de ótimos locais, em detrimento de melhorias pontuais.

Por sua vez, a configuração $ALNS_{Conf_{K1}}$ apresenta um grau de gulosidade moderadamente menos restritivo e uma temperatura inicial bastante elevada, que é balanceada por um resfriamento mais agressivo. Isso permite uma exploração mais relaxada nas iterações iniciais, aceitando uma maior quantidade de soluções mesmo que piores, porém menos prolongada ao longo da execução. Além disso, o fator de reação (r) assume um valor mediano, sem favorecer de forma clara nem o desempenho histórico nem os resultados mais recentes dos operadores, enquanto $Conf_K$ dá maior ênfase ao desempenho mais recente. Finalmente, $Conf_{K1}$ chama a atenção pelo arranjo mais equilibrado entre as recompensas, com $R^* = 58$, $R' = 45$ e $R^\vee = 50$, o que indica que tanto melhorias incrementais quanto a simples aceitação de soluções viáveis

Tabela 16 – Resultados de ALNS com diferentes configurações em $k = 2$.

Instância	$ALNS_{ConfM}$				$ALNS_{ConfK}$				$ALNS_{ConfK2}$			
	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$
Bath	86	87,6	1,0	79,6	**75**	**75,6**	0,5	41,7	**75**	76,0	0,9	36,8
Belfast	102	106,8	2,7	197,3	**86**	89,2	2,1	103,5	**86**	**89,0**	2,5	92,3
Brighton	50	51,8	1,0	85,3	**43**	**43,4**	0,5	47,5	44	45,0	0,9	45,5
Bristol	99	100,0	1,1	147,6	**79**	**82,4**	2,3	70,6	82	83,8	1,2	75,7
Cardiff	102	104,4	1,4	94,0	**86**	**88,0**	1,7	50,2	87	88,6	1,4	53,3
Coventry	93	95,0	1,7	100,4	80	**81,0**	0,9	54,7	**79**	82,6	1,9	49,2
Exeter	100	102,6	1,7	109,8	**85**	**86,4**	2,3	57,2	**85**	88,6	2,2	52,2
Glasgow	114	118,0	2,9	94,8	**102**	**103,6**	1,5	51,6	**102**	**103,6**	1,0	46,8
Leeds	104	105,8	1,0	167,6	88	90,4	1,7	92,4	**85**	**89,2**	2,9	83,9
Leicester	98	102,4	2,6	144,8	**85**	**88,2**	2,3	79,5	86	89,8	2,3	68,9
Liverpool	74	75,6	1,2	117,7	**62**	**62,6**	0,8	70,9	**62**	63,8	1,2	59,4
Manchester	102	103,8	1,3	219,1	**89**	**90,0**	1,1	129,1	**89**	**90,0**	0,6	106,9
Newcastle	103	103,6	0,5	92,9	**88**	**90,2**	1,3	45,0	92	92,8	0,8	46,6
Nottingham	107	109,6	1,4	177,3	**91**	**93,0**	1,4	75,3	94	94,8	1,2	86,3
Oxford	54	54,2	0,4	31,9	**49**	**49,4**	0,8	15,5	**49**	49,6	0,8	17,5
Plymouth	78	79,8	1,2	102,3	**66**	**68,8**	1,9	43,6	68	**68,8**	0,4	52,8
Sheffield	109	111,8	1,9	152,9	**92**	**95,4**	3,1	63,1	97	99,0	1,1	76,7
Southampton	60	60,8	0,8	61,0	**51**	**51,4**	0,5	29,5	**51**	51,8	0,4	32,1
Sunderland	91	92,2	1,0	121,0	**78**	**78,8**	0,8	66,3	**78**	79,8	1,0	57,2
York	87	87,8	1,0	84,2	**72**	**73,6**	1,0	39,8	**72**	74,2	1,2	38,3

(ainda que piores) contribuíram de forma semelhante para o ajuste dos operadores, em vez do foco exclusivo na superação da melhor solução global.

A Tabela 16 apresenta um cenário semelhante ao observado para $k = 1$. O algoritmo $ALNS_{Conf_K}$ destaca-se como a configuração de melhor desempenho, superando $ALNS_{Conf_{K2}}$ por uma margem pequena nas instâncias em que isso ocorre. Além disso, não há uma correlação evidente entre *overtuning* e a obtenção de melhores resultados nas instâncias em que $ALNS_{ConfK2}$ eventualmente supera $ALNS_{Conf_K}$. Há pouca variação nos valores dos parâmetros fornecidos pelo `irace` para $Conf_K$ e $Conf_{K2}$, sendo este o par de configurações mais similar entre todas as avaliadas. As principais diferenças concentram-se nas recompensas, com $R^* : (Conf_K = 72, Conf_{K2} = 45)$, $R' : (Conf_K = 12, Conf_{K2} = 19)$ e $R^\vee : (Conf_K = 32, Conf_{K2} = 18)$, bem como no parâmetro η_s, que passa de 303 em $Conf_K$ para 655 em $Conf_{K2}$.

Em relação às recompensas, o $ALNS_{ConfK}$ prioriza operadores que contribuem diretamente para a melhoria da melhor solução global, mas também valorizando operadores que

Tabela 17 – Resultados de ALNS com diferentes configurações em $k = 4$.

Instância	$ALNS_{ConfM}$				$ALNS_{ConfK}$				$ALNS_{ConfK4}$			
	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$
Bath	166	169,4	2,5	119,0	**162**	167,6	4,2	57,5	164	**164,4**	0,5	215,2
Belfast	202	205,4	2,2	269,0	198	209,4	10,0	140,4	**186**	**186,2**	0,4	528,1
Brighton	99	100,6	0,8	134,7	**92**	97,2	3,1	57,8	94	**94,6**	0,5	216,6
Bristol	194	196,6	1,9	239,5	200	222,8	13,0	110,2	**182**	**182,8**	1,0	404,7
Cardiff	195	199,8	2,5	157,1	211	216,6	4,3	91,7	**183**	**184,4**	0,8	270,7
Coventry	188	194,4	3,8	169,9	197	217,2	13,5	82,2	**180**	**181,6**	1,0	302,4
Exeter	200	203,8	2,1	184,9	201	210,0	12,6	93,5	**192**	**192,2**	0,4	327,6
Glasgow	219	221,4	1,9	160,0	230	233,2	2,5	97,7	**209**	**209,8**	1,2	320,0
Leeds	201	207,0	3,2	261,2	211	218,8	11,7	136,7	**192**	**194,6**	1,9	502,1
Leicester	202	207,8	2,9	235,2	209	214,8	3,9	96,0	**182**	**183,4**	0,8	443,5
Liverpool	148	150,0	1,4	184,6	147	150,0	3,0	68,3	**141**	**141,6**	0,8	320,7
Manchester	202	205,0	2,2	351,3	200	214,0	12,9	141,6	**186**	**188,0**	1,3	605,1
Newcastle	195	198,0	2,1	149,9	189	202,8	8,5	73,7	**182**	**184,4**	1,2	264,7
Nottingham	218	219,0	0,9	273,8	225	234,0	10,6	108,3	**199**	**201,6**	1,6	521,1
Oxford	105	106,8	1,3	44,1	**95**	**97,8**	2,0	24,6	99	100,4	0,8	86,9
Plymouth	156	157,0	0,9	139,1	149	162,6	9,0	64,7	**141**	**142,4**	1,2	272,0
Sheffield	214	217,6	2,1	219,9	216	236,4	15,4	105,5	**194**	**194,6**	0,8	438,5
Southampton	120	122,4	1,6	86,9	121	125,8	3,6	44,0	**112**	**112,8**	0,4	157,3
Sunderland	179	181,2	1,3	174,1	171	175,2	2,9	77,2	**167**	**168,2**	0,8	364,6
York	171	173,4	2,2	112,3	170	180,6	13,1	54,1	**157**	**158,2**	0,8	246,0

favorecem a diversificação por meio da aceitação de soluções inferiores, ainda que em menor escala. Já o $ALNS_{ConfK2}$ atribui um peso mais equilibrado às recompensas, favorecendo de forma mais moderada operadores que geram soluções ótimas globais, sem privilegiar melhorias incrementais ou simples aceitações via o critério de Metropolis. Por fim, o aumento expressivo no valor de η_s em $Conf_{K2}$ faz com que os operadores mantenham seus pesos inalterados por um intervalo significativamente maior. Isso faz com que a convergência demore mais para acontecer e, quando ocorre, é de forma mais abrupta, já que acumula muito mais informações antes de atualizar os pesos. Na prática, isso torna a adaptação do algoritmo mais lenta, mas também mais sensível ao desempenho acumulado dos operadores ao longo de cada segmento.

Os resultados da Tabela 17 rompem com o padrão observado nos experimentos anteriores. A Seção 5.3.3 já havia apontado uma limitação dos operadores em explorar o espaço de soluções quando $k = 4$, cenário no qual os resultados das heurísticas construtivas determinísticas superaram o $ALNS_{ConfM}$ em várias instâncias. Em outras palavras, uma simples construção a

partir de um conjunto vazio se mostrou mais eficiente do que o ciclo de destruição e reconstrução das metaheurísticas LNS e ALNS. Esse é um ponto bastante relevante que ajuda a entender o desempenho da configuração $Conf_{K4}$, que apresentou os melhores resultados na maioria das instâncias.

O algoritmo $ALNS_{ConfK4}$ apresenta uma configuração consideravelmente distinta das demais, com alterações expressivas em diversos parâmetros: $\alpha = 0.889$, $T_0 = 20$, $T_F = 7.751$, $\gamma = 0.992$ e $\eta_s = 591$. Além disso, há uma ênfase na recompensa por encontrar a melhor solução global, com valor de 62, enquanto as recompensas atribuídas a melhorias parciais ou a soluções aceitas são consideravelmente menores, assumindo os valores 13 e 7, respectivamente. Novamente, o alto valor de η_s retarda a convergência para um operador mais eficiente. Esse comportamento possivelmente favorece uma maior diversidade na seleção de operadores durante as primeiras 600 iterações, permitindo que os operadores mais eficientes sejam priorizados nas etapas seguintes. Por fim, os parâmetros associados ao critério de Metropolis indicam uma estratégia de resfriamento bastante gradual. Embora a temperatura inicial seja relativamente baixa, o alto valor de γ prolonga o processo, resultando em um critério de aceitação que permanece moderadamente permissivo por boa parte da execução do algoritmo, e finaliza em uma temperatura consideravelmente mais elevada do que nas demais configurações. Esse cenário habilita uma exploração maior do espaço de soluções em estágios finais da execução, se comparados aos demais algoritmos ALNS sob as diferentes configurações.

Conclui-se que há uma limitação dos operadores de reconstrução desenvolvidos para explorar o espaço de soluções do MkDSP nas instâncias avaliadas com $k = 4$, uma vez que diferentes configurações não foram capazes de superar os resultados obtidos por uma construção determinística. Dessa forma, observa-se que a configuração que apresentou os melhores resultados buscou, em certo sentido, emular o comportamento do método construtivo simples, adotando um alto grau de destruição e um critério de aceitação mais relaxado ao longo da execução. Sobretudo, no que diz respeito às configurações comparadas nesta seção, pode-se afirmar que $ALNS_{ConfK}$ se destaca como a melhor configuração para o ALNS quando $k = \{1,2\}$, enquanto $ALNS_{ConfK4}$ apresentou desempenho superior para $k = 4$.

5.3.5 Comparação com o Estado da Arte

As seções anteriores foram dedicadas aos experimentos e comparações entre os métodos desenvolvidos neste trabalho, com o objetivo de extrair os melhores resultados e compreender o funcionamento dos operadores e metaheurísticas propostos. Nesse processo, a metaheurística ALNS demonstrou ser o algoritmo mais eficaz na aproximação do conjunto k-dominante mínimo nas instâncias avaliadas para $k = \{1,2,4\}$. Diante disso, esta seção tem como obje-

Tabela 18 – Comparação dos resultados VNS e ALNS para $k = 1$.

Instância	*Tamanho*		*ALNS*					*VNS*		
	$\|V\|$	$\|E\|$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	Conf.	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$
Bath	910	18560	39	39,0	0,0	26,9	$Conf_K$	**38,0**	0,0	661,7
Belfast	1700	62617	41	41,2	0,4	87,7	$Conf_{K_1}$	**39,0**	0,0	1800,1
Brighton	976	35012	22	22,0	0,0	34,6	$Conf_K$	**21,0**	0,0	1334,0
Bristol	1569	47522	**37**	**37,0**	0,0	**51,9**	$Conf_K$	**37,0**	0,0	1800,1
Cardiff	1127	23155	41	41,8	0,8	26,6	$Conf_K$	**39,0**	0,0	968,2
Coventry	1175	26689	39	39,2	0,4	30,6	$Conf_K$	**38,0**	0,0	1098,1
Exeter	1250	31997	40	40,6	0,5	32,7	$Conf_K$	**38,0**	0,0	1365,8
Glasgow	1137	24323	53	53,4	0,5	39,3	$Conf_{K_1}$	**50,1**	0,3	920,6
Leeds	1647	56511	43	43,0	0,0	61,9	$Conf_K$	**40,0**	0,0	1800,1
Leicester	1531	48219	**38**	38,2	0,4	**50,1**	$Conf_K$	**38,0**	0,0	1800,1
Liverpool	1273	42564	29	29,0	0,0	43,5	$Conf_K$	**28,0**	0,0	1800,1
Manchester	1991	77286	39	40,8	1,0	80,0	$Conf_K$	**38,3**	0,7	1800,2
Newcastle	1109	26614	46	46,6	0,5	38,4	$Conf_{K_1}$	**44,0**	0,0	1146,4
Nottingham	1739	51595	**44**	**44,0**	0,0	**67,7**	$Conf_K$	**44,0**	0,0	1799,1
Oxford	479	8396	26	26,0	0,0	10,8	$Conf_K$	**24,0**	0,0	263,1
Plymouth	1122	35070	**31**	31,2	0,4	**40,7**	$Conf_K$	**31,0**	0,0	1398,8
Sheffield	1582	50534	43	44,2	0,8	53,5	$Conf_K$	**42,0**	0,0	1800,2
Southampton	796	19942	**25**	25,0	0,0	**25,2**	$Conf_K$	**25,0**	0,0	1503,9
Sunderland	1346	42013	38	38,0	0,0	50,0	$Conf_K$	**36,0**	0,0	1559,3
York	1044	23774	35	35,6	0,5	34,6	$Conf_{K_1}$	**32,0**	0,0	856,8

tivo comparar o desempenho do ALNS com o VNS Predojević, Kartelj e Djukanović (2023), que representa o estado da arte para as instâncias utilizadas neste trabalho. Busca-se avaliar se os resultados obtidos pelas melhores configurações do ALNS são competitivos em relação aos apresentados na literatura.

As tabelas comparativas 18, 19 e 20 estão organizadas da seguinte forma. As três primeiras colunas são destinadas às instâncias e suas dimensões, representadas pela cardinalidade do conjunto de vértices e do conjunto de arestas. As cinco colunas subsequentes correspondem aos resultados do ALNS, exibindo as métricas $|S^*|$ (melhor solução encontrada), $\overline{|S|}$ (média das soluções), $\sigma(|S|)$ (desvio padrão) e $\bar{t}$ (tempo médio de execução), além da coluna $Conf.$, que indica a configuração utilizada para gerar os resultados. Por fim, as três últimas colunas apresentam os dados obtidos por Predojević, Kartelj e Djukanović (2023) para o VNS, contendo a média das soluções ($\overline{|S|}$), o desvio padrão ($\sigma(|S|)$) e o tempo médio de execução ($\bar{t}$).

O ALNS sob a configuração $ALNS_{Conf_K}$ iguala os melhores resultados obtidos pelo VNS em cinco instâncias: *Bristol*, *Leicester*, *Nottingham*, *Plymouth* e *Southampton* conforme apre-

Tabela 19 – Comparação dos resultados VNS e ALNS para $k = 2$.

Instância	*Tamanho*		*ALNS*					*VNS*		
	$\|V\|$	$\|E\|$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	Conf.	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$
Bath	910	18560	75	75,6	0,5	41,7	$Conf_K$	**71,1**	0,3	720,7
Belfast	1700	62617	86	89,0	2,5	92,3	$Conf_{K_2}$	**76,3**	0,5	1800,3
Brighton	976	35012	43	43,4	0,5	47,5	$Conf_K$	**40,1**	0,3	1789,7
Bristol	1569	47522	79	82,4	2,3	70,6	$Conf_K$	**73,8**	0,4	1800,1
Cardiff	1127	23155	86	88,0	1,7	50,2	$Conf_K$	**78,3**	0,5	900,5
Coventry	1175	26689	80	81,0	0,9	54,7	$Conf_K$	**73,0**	0,0	1002,5
Exeter	1250	31997	85	86,4	2,3	57,2	$Conf_K$	**77,0**	0,0	1544,2
Glasgow	1137	24323	102	103,6	1,5	51,6	$Conf_K$	**94,0**	0,5	1068,4
Leeds	1647	56511	85	89,2	2,9	83,9	$Conf_K$	**79,5**	0,5	1800,1
Leicester	1531	48219	85	88,2	2,3	79,5	$Conf_K$	**75,0**	0,0	1800,1
Liverpool	1273	42564	62	62,6	0,8	70,9	$Conf_K$	**57,0**	0,5	1800,1
Manchester	1991	77286	89	90,0	1,1	129,1	$Conf_{K_2}$	**77,9**	0,3	1800,1
Newcastle	1109	26614	88	90,2	1,3	45,0	$Conf_K$	**83,6**	0,5	1020,5
Nottingham	1739	51595	91	93,0	1,4	75,3	$Conf_K$	**84,7**	0,5	1800,2
Oxford	479	8396	49	49,4	0,8	15,5	$Conf_K$	**47,0**	0,0	298,0
Plymouth	1122	35070	66	68,8	1,9	43,6	$Conf_K$	**61,3**	0,5	1694,2
Sheffield	1582	50534	92	95,4	3,1	63,1	$Conf_K$	**84,6**	0,5	1747,7
Southampton	796	19942	51	51,4	0,5	29,5	$Conf_K$	**49,2**	0,4	807,2
Sunderland	1346	42013	78	78,8	0,8	66,3	$Conf_K$	**73,0**	0,0	1049,0
York	1044	23774	72	73,6	1,0	39,8	$Conf_K$	**68,0**	0,0	573,0

sentado na Tabela 18. Além disso, permanece a apenas um vértice do resultado ideal em outras seis instâncias e apresenta um desvio máximo de três vértices em apenas duas. De forma geral, ao comparar os resultados obtidos por ambos os algoritmos em cada instância, observa-se que o ALNS alcança, em média, 96% da qualidade dos resultados fornecidos pelo VNS. Vale destacar ainda que o tempo computacional dedicado às 5000 iterações do ALNS representa uma fração do tempo, com uma média de 3% do tempo empregado pelo algoritmo da literatura, o que confere uma vantagem expressiva à abordagem desenvolvida, que possivelmente poderia atingir resultados ainda melhores caso operasse com um número maior de iterações. Nesse sentido, os resultados da coluna $\bar{t}$ também são destacados em negrito para o algoritmo com menor tempo nos casos de empate mencionados anteriormente, evidenciando essa vantagem e servindo como critério de desempate para esses cenários.

A Tabela 19 apresenta a comparação entre os melhores resultados do ALNS e do VNS para $k = 2$. Enquanto os resultados da Tabela 18 revelam um cenário moderadamente competitivo entre os dois algoritmos, os resultados para $k = 2$ indicam que o VNS se destaca como

Tabela 20 – Comparação dos resultados VNS e ALNS para $k = 4$.

Instância	*Tamanho*		*ALNS*					*VNS*		
	$\|V\|$	$\|E\|$	$\|S^*\|$	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$	Conf.	$\overline{\|S\|}$	$\sigma(\|S\|)$	$\bar{t}$
Bath	910	18560	162	167,6	4,2	57,5	$Conf_K$	**140,1**	0,7	644,8
Belfast	1700	62617	186	186,2	0,4	528,1	$Conf_{K_4}$	**148,3**	0,7	1800,1
Brighton	976	35012	94	94,6	0,5	216,6	$Conf_{K_4}$	**78,0**	0,5	1800,1
Bristol	1569	47522	182	182,8	1,0	404,7	$Conf_{K_4}$	**146,6**	1,1	1797,8
Cardiff	1127	23155	183	184,4	0,8	270,7	$Conf_{K_4}$	**157,5**	0,8	660,8
Coventry	1175	26689	180	181,6	1,0	302,4	$Conf_{K_4}$	**149,2**	0,9	827,3
Exeter	1250	31997	192	192,2	0,4	327,6	$Conf_{K_4}$	**158,1**	0,7	943,2
Glasgow	1137	24323	209	209,8	1,2	320,0	$Conf_{K_4}$	**175,2**	0,9	745,6
Leeds	1647	56511	192	194,6	1,9	502,1	$Conf_{K_4}$	**152,8**	0,8	1800,1
Leicester	1531	48219	182	183,4	0,8	443,5	$Conf_{K_4}$	**149,3**	0,7	1759,0
Liverpool	1273	42564	141	141,6	0,8	320,7	$Conf_{K_4}$	**112,8**	0,6	1800,1
Manchester	1991	77286	186	188,0	1,3	605,1	$Conf_{K_4}$	**155,2**	0,6	1800,1
Newcastle	1109	26614	182	184,4	1,2	264,7	$Conf_{K_4}$	**152,4**	0,5	951,3
Nottingham	1739	51595	199	201,6	1,6	521,1	$Conf_{K_4}$	**164,2**	0,8	1800,2
Oxford	479	8396	95	97,8	2,0	24,6	$Conf_K$	**89,0**	0,0	254,4
Plymouth	1122	35070	141	142,4	1,2	272,0	$Conf_{K_4}$	**115,6**	0,5	1638,0
Sheffield	1582	50534	194	194,6	0,8	438,5	$Conf_{K_4}$	**161,4**	0,8	1800,0
Southampton	796	19942	112	112,8	0,4	157,3	$Conf_{K_4}$	**97,6**	0,5	1129,0
Sunderland	1346	42013	167	168,2	0,8	364,6	$Conf_{K_4}$	**141,0**	0,5	1438,3
York	1044	23774	157	158,2	0,8	246,0	$Conf_{K_4}$	**130,4**	0,5	784,5

o melhor algoritmo. Ainda assim, embora os resultados estejam mais distantes, o método proposto neste trabalho mantém um desempenho satisfatório em instâncias como *Bath*, *Leeds*, *Newcastle*, *Oxford*, *Southampton*, *Sunderland* e *York*, alcançando entre 94% e 96% dos resultados obtidos pelo VNS. Por outro lado, observa-se um desempenho inferior em instâncias como *Belfast* e *Manchester*, que não chegam a 90% dos valores do VNS. De maneira geral, o ALNS atinge, na média, apenas 92% dos resultados alcançados pelo VNS. Novamente, vale destacar que o tempo computacional requerido pelo ALNS para obter esses resultados é menor que o tempo empregado pelo VNS, com apenas algumas poucas instâncias excedendo 5% desse tempo, o que ainda sugere que o ALNS poderia aprimorar seus resultados dedicando maior tempo de execução.

A Tabela 20 apresenta a última comparação entre os dois algoritmos. Os resultados das Tabelas 18 e 19 já indicavam uma perda de desempenho do ALNS em relação ao VNS, o que é novamente confirmado na tabela atual. No entanto, essa diferença torna-se ainda mais acentuada quando comparada aos resultados anteriores. Enquanto os valores obtidos pela abordagem

proposta alcançavam pelo menos 90% dos resultados do VNS (salvo dois casos pontuais já citados), para $k = 4$ não há qualquer instância que atinja essa marca. Na verdade, os resultados concentram-se na faixa dos 80% (ainda que nunca abaixo disso), com média geral de 83%. Isso evidencia não apenas uma diferença expressiva entre os dois algoritmos, com clara superioridade do VNS, mas também uma discrepância significativa em relação às comparações anteriores. Além dos resultados, observa-se que o tempo necessário para concluir as 5000 iterações aumentou drasticamente em comparação ao cenário com $k = 2$. Este aumento se reflete também na relação com o tempo dedicado pelo VNS, que permanece praticamente constante para todos os valores de k. Como consequência, o tempo médio empregado pelo ALNS representa agora 26% do tempo total do VNS, chegando a 43% em instâncias maiores, como *Glasgow*. Assim, a métrica de tempo, que antes sugeria uma vantagem relevante para o ALNS, torna-se menos apelativa, além de evidenciar uma possível limitação do algoritmo quanto à sua escalabilidade frente ao aumento do valor de k.

5.3.6 Análise Detalhada do ALNS

Diferentemente dos experimentos comparativos discutidos nas seções anteriores, esta seção tem como objetivo analisar o funcionamento dos operadores integrados ao ALNS. Nesse contexto, pretende-se verificar se o desempenho observado para os operadores isolados é mantido ao serem combinados ao *framework* do ALNS. Adicionalmente, as informações apresentadas nesta seção proporcionam uma visão clara de como o mecanismo adaptativo se comporta para $k = \{1, 2, 4\}$, considerando também os efeitos de diferentes configurações. Deste modo, em trabalhos futuros, novas técnicas podem ser desenvolvidas com base no entendimento do funcionamento do mecanismo adaptativo, visando à criação de algoritmos que apresentem melhor desempenho.

Para cada operador de construção, foram gerados gráficos de linha para o número de chamadas por segmento, a evolução dos pesos por segmento e a pontuação agregada. Essas métricas permitem inferir uma série de informações relevantes, como a variabilidade no uso dos operadores, os segmentos em que ocorreram maiores alterações na solução incumbente ou na melhor solução global, além da identificação dos operadores de maior destaque em diferentes fases da busca. Os gráficos foram gerados para quatro configurações previamente apresentadas: $Conf_K$, $Conf_{K1}$, $Conf_{K2}$ e $Conf_{K4}$. A configuração geral ($Conf_K$) recebeu gráficos para cada métrica e para os três valores de k, enquanto as demais configurações apresentam os gráficos das três métricas consideradas apenas para o respectivo valor de k para o qual foram projetadas. Por fim, a instância *Bristol* foi selecionada para os testes, por se tratar de uma instância de

tamanho mediano dentro do conjunto analisado, oferecendo uma visão mais equilibrada do comportamento dos operadores de reconstrução e do mecanismo adaptativo.

Acerca dos operadores, observa-se que os desempenhos obtidos de forma isolada, seja como heurísticas construtivas determinísticas ou como operadores de reconstrução no contexto do LNS, foram refletidos pela avaliação do mecanismo adaptativo do ALNS. Em outras palavras, os algoritmos que apresentaram melhor desempenho nesses cenários também foram favorecidos durante a seleção dos operadores ao longo da execução do ALNS. Os operadores correspondentes às heurísticas H_V, $H\varphi_V$ e $H\varphi_D$ destacaram-se em relação aos operadores baseados no grau de dominação (H_D) e na construção aleatória (H_R), como pode ser verificado tanto na Figura 9a quanto na Figura 10a, que apresentam o número de chamadas de cada operador ao longo da execução.

A Figura 10a, em particular, revela um comportamento interessante quando associada às métricas exibidas nas Figuras 10b e 10c. Elas mostram, respectivamente, a pontuação acumulada e a evolução dos pesos dos operadores durante a execução. Inicialmente, o operador $H\varphi_D$ obtém sucesso, encontrando boas soluções e acumulando peso. Contudo, à medida que os ótimos locais tornam-se mais difíceis de serem superados, o peso elevado desse operador faz com que ele seja selecionado com maior frequência, mesmo sem conseguir romper o ótimo local atual. Com isso, ele deixa de acumular a pontuação necessária para se manter relevante. Por outro lado, os operadores H_V e $H\varphi_V$ demonstram maior eficácia em escapar do ótimo local nesse cenário, sendo recompensados com aumentos de peso e, por consequência, um maior número de chamadas até o final da execução.

As Figuras 9d, 9e e 9f, que representam a configuração geral $Conf_K$, e as Figuras 10d, 10e e 10f, referentes à configuração $Conf_{K2}$ para $k = 2$, revelam comportamentos semelhantes no que diz respeito ao desempenho dos operadores. Em ambos os casos, o operador $H\varphi_V$ destaca-se como o mais eficiente, o que corrobora os resultados observados tanto nas heurísticas construtivas quanto nos experimentos realizados com o LNS. Apesar dessa semelhança, nota-se que a configuração $Conf_{K2}$ apresenta uma distribuição de chamadas ligeiramente mais equilibrada entre os operadores, indicando uma maior diversidade no seu uso ao longo da execução. Por outro lado, a configuração $Conf_K$ adota um comportamento mais conservador, apresentando uma tendência mais acentuada de concentração nos operadores que inicialmente demonstraram melhor desempenho, refletindo certo favoritismo no processo de seleção adaptativa. Esse comportamento não necessariamente se prova como má opção, visto que $Conf_K$ apresentou resultados superiores para essa instância.

Por fim, a linha inferior exibe os comportamentos mais distintos entre as configurações $Conf_K$ e $Conf_{K4}$. Como discutido em seções anteriores, elas apresentam uma série de parâmetros significativamente diferentes. Os resultados da Tabela 17 já haviam indicado que $Conf_{K4}$

era a configuração superior, fato que também pode ser observado a partir dos gráficos. Primeiramente, $Conf_{K4}$ revela uma maior diversidade na utilização dos operadores, com destaque para H_V, que acompanha $H\varphi_V$, mesmo que com peso, pontuação e número de chamadas consideravelmente menores.

As Figuras 10h e 10i, que representam a pontuação acumulada e a evolução dos pesos dos operadores na terceira linha (onde $k = 4$) para $Conf_{K4}$, indicam que a pontuação da solução na instância *Bristol* seguiu em crescimento, enquanto os pesos dos principais operadores eram mantidos estáveis. Isso sugere que esses operadores continuavam sendo capazes de superar ótimos locais e conduzir a solução a melhorias sucessivas. Esse comportamento contrasta com os mesmos gráficos para $Conf_K$, que demonstram uma menor diversidade na utilização dos operadores. Embora $H\varphi_V$ ainda se mantenha como o operador de destaque, sua pontuação apresenta sinais claros de estagnação, ao passo que seus pesos, assim como os dos demais operadores, começam a convergir, indicando uma dificuldade maior em superar novos ótimos locais. Isso é refletido em seus resultados, visto que $ALNS_{ConfK}$ teve sua melhor solução com 200 vértices, enquanto $ALNS_{ConfK4}$ obteve 182. Mais expressiva é a média das soluções encontradas, onde o segundo algoritmo supera o primeiro por 40 vértices.

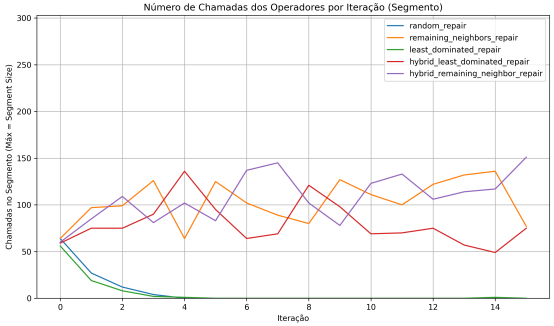
Aproveitando os tópicos discutidos nos parágrafos anteriores, que tratam dos cenários nos quais o algoritmo parece conduzir a soluções mais promissoras, é igualmente relevante destacar como o próprio mecanismo adaptativo pode, em certas circunstâncias, induzir a busca a vizinhanças excessivamente restritas, dificultando a superação de ótimos locais. Há um conjunto de fatores observáveis nos gráficos que caracterizam esse comportamento: baixa diversidade na escolha de operadores, dificuldade do operador predominante em superar o ótimo local atual e uma temperatura excessivamente baixa, proveniente do critério de Metropolis, que restringe a aceitação de soluções piores.

Esses traços são claramente observáveis nas linhas referentes a $k = 2$ e $k = 4$ na Figura 9 (gráficos *d*, *e*, *f*, *g*, *h* e *i*) e também para $k = 2$ na Figura 10 (gráficos *d*, *e* e *f*). Todas essas figuras apresentam, em diferentes níveis, um cenário no qual há uma predominância prolongada de um único operador, cuja pontuação acumulada fica estagnada, seguida por um declínio gradual em seu peso. Ainda assim, os demais operadores mantêm pesos persistentemente baixos, sem recuperarem seu espaço na busca. A estagnação da pontuação, aliada à redução do peso do operador dominante, sugere uma dificuldade clara em escapar de ótimos locais, um problema que é agravado pelo baixo valor da temperatura e que reduz drasticamente a chance de aceitar movimentos sub-ótimos.

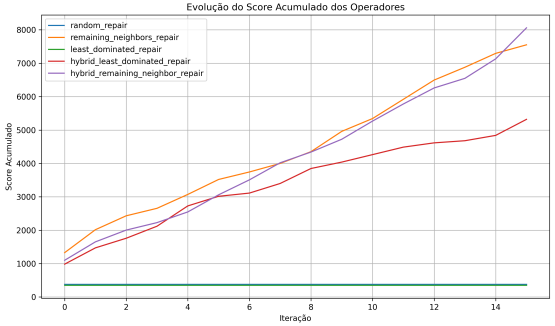
Vale destacar que, neste trabalho, foi utilizado apenas um operador de destruição. É plausível supor que a introdução de operadores adicionais poderia mitigar esse problema, embora não haja garantias concretas dessa suposição. Outra estratégia viável seria a adoção de um

mecanismo de reinicialização dos pesos dos operadores, de forma a restaurar a diversidade e reiniciar o processo adaptativo, favorecendo a retomada da exploração do espaço de soluções. Por fim, observa-se que o operador de reconstrução $H\varphi_V$ demonstrou desempenho destacado em grande parte dos experimentos. Dessa forma, não é surpreendente que o mecanismo adaptativo tenda a priorizá-lo, gerando o cenário descrito acima.

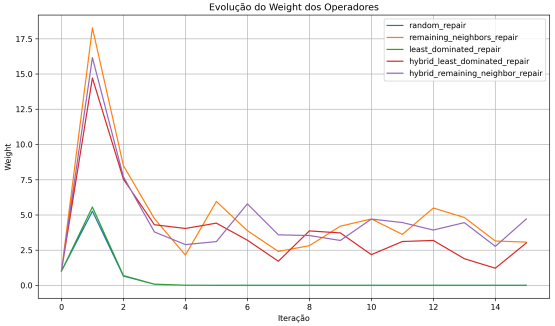
Figura 9 – Progressão dos operadores de reconstrução para a instância *Bristol* considerando três métricas (número de chamadas, score acumulado e peso) com a configuração $Conf_K$ para diferentes valores de $k = 1, 2, 4$



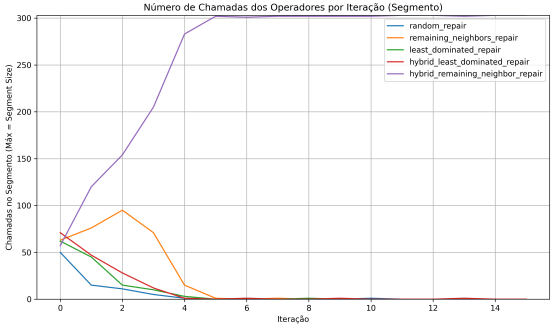
(a) Chamadas ($k = 1$)



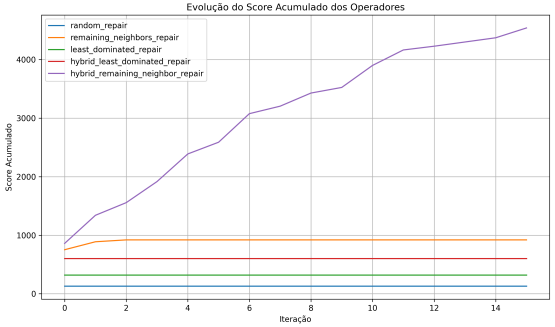
(b) Score agregado ($k = 1$)



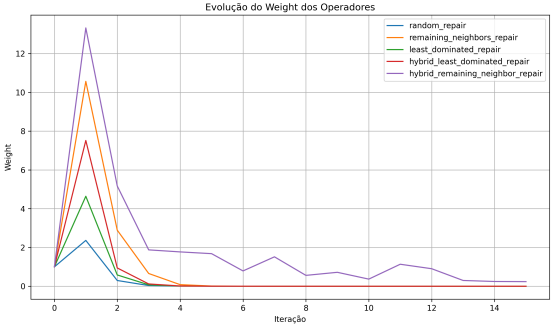
(c) Pesos ($k = 1$)



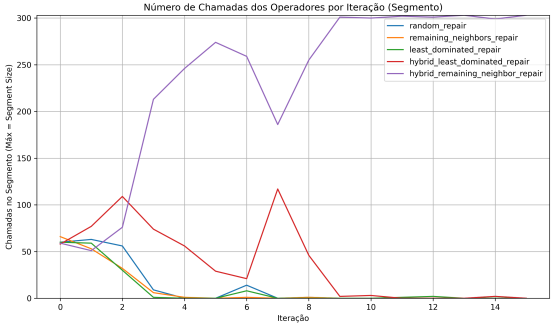
(d) Chamadas ($k = 2$)



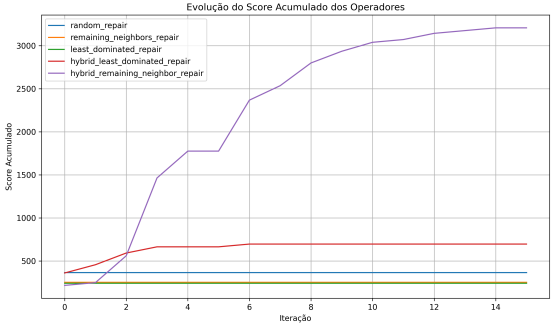
(e) Score agregado ($k = 2$)



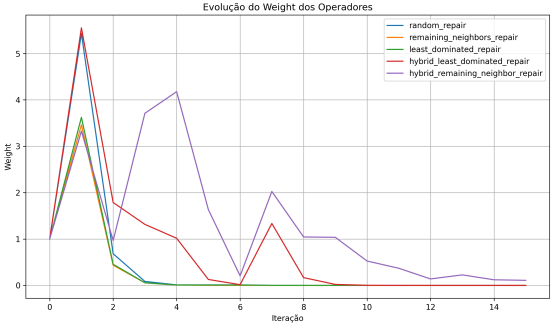
(f) Pesos ($k = 2$)



(g) Chamadas ($k = 4$)

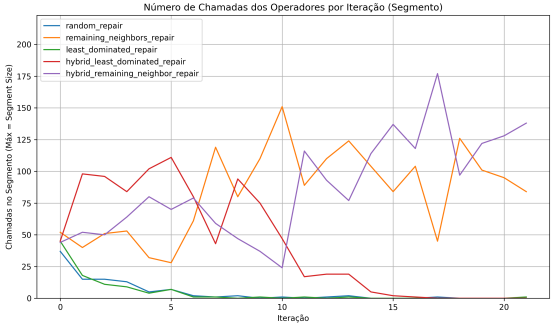


(h) Score agregado ($k = 4$)

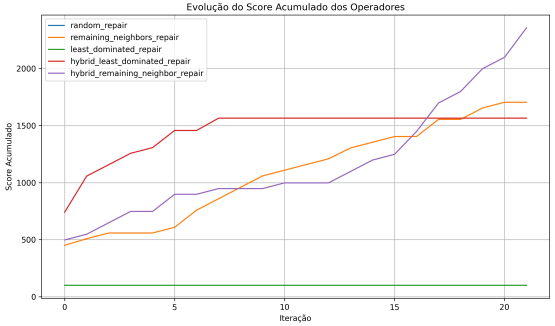


(i) Pesos ($k = 4$)

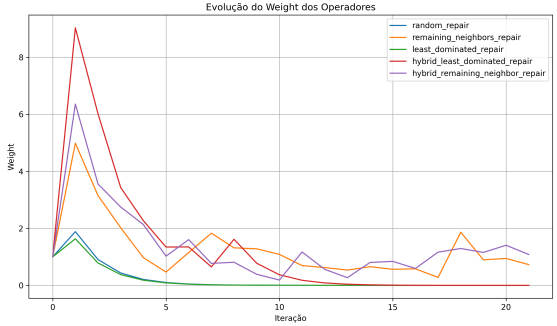
Figura 10 – Progressão dos operadores de reconstrução para a instância *Bristol* considerando três métricas (número de chamadas, score acumulado e peso) com as configurações $Conf_{K1}$, $Conf_{K2}$ e $Conf_{K4}$ para os respectivos valores de $k = 1, 2, 4$



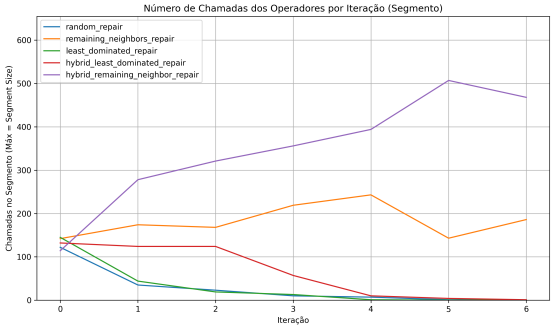
(a) Chamadas ($k = 1$)



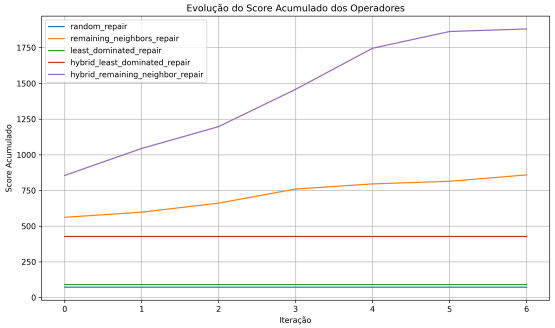
(b) Score agregado ($k = 1$)



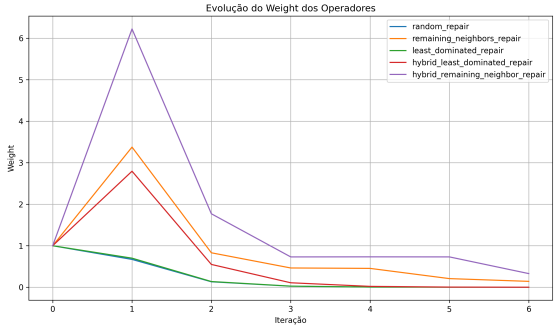
(c) Pesos ($k = 1$)



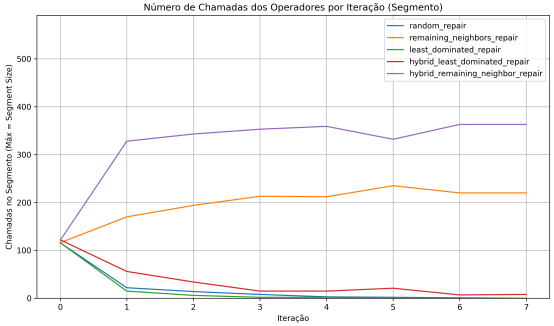
(d) Chamadas ($k = 2$)



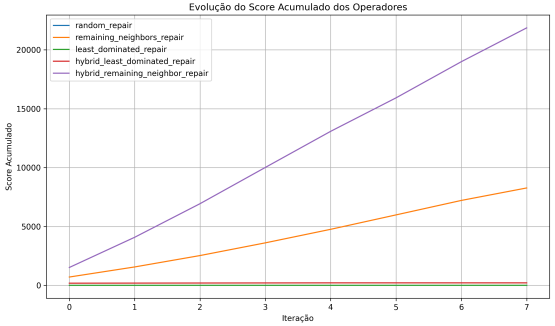
(e) Score agregado ($k = 2$)



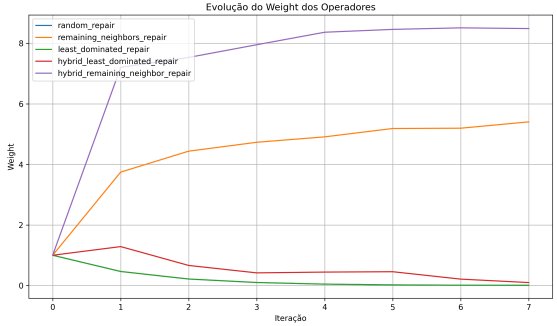
(f) Pesos ($k = 2$)



(g) Chamadas ($k = 4$)



(h) Score agregado ($k = 4$)



(i) Pesos ($k = 4$)

6 CONSIDERAÇÕES FINAIS

Este trabalho aborda o problema do conjunto k-dominante mínimo (MkDSP) em grafos não ponderados e não direcionados, que consiste em uma generalização do clássico problema do conjunto dominante mínimo (MDSP). O objetivo do MkDSP é determinar o menor subconjunto de vértices tal que todos os vértices do grafo estejam no conjunto ou sejam k-dominados por ele, i.e. possuam ao menos k vizinhos pertencentes ao conjunto solução.

O trabalho tem início com uma ampla revisão da literatura, na qual são apresentados os principais métodos desenvolvidos até o momento para resolver o MkDSP. As abordagens identificadas foram devidamente categorizadas de acordo com as técnicas adotadas e tiveram seu funcionamento analisado e detalhado, fornecendo uma base sólida para o desenvolvimento das propostas deste trabalho. Ao longo do desenvolvimento deste trabalho, foram adaptadas e implementadas cinco heurísticas construtivas, baseadas nas estratégias descritas por Rangel (2023). Tais heurísticas serviram como base para a formulação de duas metaheurísticas distintas. A primeira delas é o *Large Neighborhood Search* (LNS), que combina operadores de destruição e reconstrução para explorar o espaço de soluções. Foram desenvolvidos cinco algoritmos baseados no LNS, cada um associado a uma das heurísticas construtivas, utilizadas como operadores de reconstrução, junto a um operador de destruição aleatório. A segunda abordagem, que representa a principal contribuição deste trabalho, é o *Adaptive Large Neighborhood Search* (ALNS). Este algoritmo expande a lógica do LNS ao incorporar um mecanismo adaptativo de seleção dos operadores, permitindo que diferentes heurísticas atuem como operadores de reconstrução ao longo da busca, com suas probabilidades ajustadas dinamicamente de acordo com o desempenho observado durante a execução.

A partir disso, foram conduzidos experimentos comparativos, iniciando pela avaliação das heurísticas construtivas, avançando para as combinações da metaheurística LNS e, por fim, para as diferentes configurações propostas para o ALNS. Este último foi então comparado com os melhores resultados da literatura, demonstrando ser competitivo para valores baixos de k, embora apresente limitações à medida que esse parâmetro aumenta. Ainda assim, o ALNS mostrou-se capaz de alcançar resultados satisfatórios para as instâncias analisadas. Uma análise detalhada do mecanismo adaptativo e dos operadores é conduzida no último segmento deste trabalho, explorando seu comportamento nas diferentes configurações propostas ao ALNS. A partir dessa análise, são extraídas observações que fomentam o projeto de estratégias para superar as limitações identificadas ao longo dos experimentos.

6.1 TRABALHOS FUTUROS

Esta última seção tem como objetivo apontar os direcionamentos mais promissores para a continuidade do estudo desenvolvido neste trabalho. As ideias apresentadas a seguir incluem tanto conclusões obtidas ao final do desenvolvimento quanto propostas de componentes algorítmicos e operadores discutidos ao longo do trabalho, mas que, por diferentes motivos, não chegaram a ser implementados.

Inicialmente, o simples aumento no número de execuções ou de iterações de certos experimentos poderia ser útil tanto para solidificar algumas conclusões quanto para testar novas hipóteses. Por exemplo, a dificuldade do LNS em superar os melhores resultados das heurísticas construtivas determinísticas para $k = 2$ e $k = 4$ poderia ser melhor compreendida ao aumentar a quantidade de execuções, especialmente com um número maior de iterações em cada experimento. Isso permitiria verificar se um tempo de execução mais prolongado aumentaria significativamente a capacidade de exploração do espaço de solução em cada operador de reconstrução.

De modo semelhante às diferentes configurações propostas para o ALNS, o LNS também poderia se beneficiar de uma configuração automática de parâmetros. Mais interessante ainda do que a possível melhoria nos resultados seria a análise dos valores atribuídos aos parâmetros α (grau de restrição da lista RCL) e β (grau de destruição da solução) para os diferentes operadores, especialmente considerando que o `irace` forneceu valores de α especialmente baixos em todas as configurações geradas para o ALNS.

Além desses aspectos, melhorias relacionadas ao consumo de memória e, em segundo plano, à complexidade temporal, podem ser implementadas com o objetivo de tornar os algoritmos mais escaláveis e eficientes na resolução de instâncias maiores do problema.

O restante desta discussão é voltada aos ajustes e sugestões que podem ser aplicados ao ALNS com o intuito de melhorar seus resultados e aproximá-los ainda mais do estado da arte, representado pelo VNS, tendo em vista a proximidade do desempenho de ambos os algoritmos.

Dentre as ideias mais diretas para aprimorar o ALNS está a adição de operadores de reconstrução mais competitivos, que rivalizem com o operador da heurística H_{φ_V}. Essa pode se provar uma tarefa complexa, mas há alguns operadores que foram idealizados durante o desenvolvimento do trabalho. O primeiro vem de encontro ao mau desempenho do grau de dominação, que poderia ser reformulado, agregando o atual grau de dominação da vizinhança fechada de cada vértice, o que refletiria melhor as áreas que de fato são menos dominadas, e não apenas os vértices menos dominados. Essa reformulação também tornaria essa nova métrica, que poderia ser chamada de grau de dominação agregado, mais adequada para valores menores de k, como foi o caso dos valores explorados neste trabalho. Finalizando a proposta,

um critério de desempate poderia ser usado para determinar a escolha do vértice da vizinhança menos dominada do grafo.

Outras propostas similares, que buscam fazer uma exploração de diferentes áreas do grafo, idealizam o uso de métricas já estudadas na teoria dos grafos, como a distância entre cada vértice, e também a centralidade dos vértices do grafo, para ocupar áreas igualmente afastadas durante a construção da solução. Isso pode se provar interessante em grafos reais. Contudo, vale ressaltar que as funções de avaliação que usassem qualquer uma dessas métricas precisariam de outra métrica, como o grau de vizinhança, a fim de fundamentar a escolha dos vértices. Em outras palavras, enquanto estas são ideias interessantes ainda não propostas pela literatura, não atacam o problema da dominação diretamente, e provavelmente deveriam ser acompanhadas de métricas que avaliam o problema mais diretamente, tornando-se guias secundários.

A proposta de operadores de destruição específicos ao M*k*DSP também se mostra bastante interessante, não apenas por permitir uma destruição mais orientada, mas simplesmente porque uma maior variabilidade seria benéfica por si só. Entre os conceitos de operadores de destruição mais promissores que foram idealizados está a destruição a partir de *clusters* do grafo. Ao organizar o grafo em segmentos a partir de características similares, operadores que se destacam ao construir soluções para estas características poderiam ter seu desempenho potencializado, e isso seria possível simplesmente destruindo essas parcelas da solução e permitindo os operadores de reconstrução agirem nesses *clusters* isolados.

Ainda que a implementação de novos operadores seja uma proposta interessante, ela não representa o único caminho para o aprimoramento do ALNS. Mecanismos adicionais, como etapas de pré e pós-processamento de vértices, já foram empregados em diferentes algoritmos da literatura e constituem formas simples de conduzir a solução por regiões mais promissoras do espaço de busca, bem como de eliminar vértices redundantes da solução final. Além disso, mecanismos de memória capazes de manter um histórico das últimas soluções geradas poderiam viabilizar estratégias de recompensa para soluções inéditas, aumentando a variabilidade sem a necessidade de criação de novos operadores de vizinhança.

Por fim, conforme discutido na Seção 5.3.6, o mecanismo adaptativo do ALNS tende a priorizar excessivamente um único operador, o que pode levar, em casos específicos, à estagnação da exploração do espaço de busca. Nesse sentido, seria interessante implementar um componente capaz de detectar esse tipo de comportamento e reinicializar os pesos e probabilidades dos operadores, reequilibrando a seleção, a fim de escapar de ótimos locais.

Finalmente, a adição de testes estatísticos para avaliar a significância dos resultados mais próximos seria relevante para definir se os melhores resultados encontrados apresentam uma melhoria significante levando em consideração o tempo usado para alcançá-los.

REFERÊNCIAS

BIRATTARI, M.; KACPRZYK, J. **Tuning metaheuristics: a machine learning perspective**. [S.l.]: Springer, 2009. v. 197.

CÁCERES, L. P.; LÓPEZ-IBÁÑEZ, M.; HOOS, H.; STÜTZLE, T. An experimental study of adaptive capping in irace. In: SPRINGER. **International Conference on Learning and Intelligent Optimization**. [S.l.], 2017. p. 235–250.

CASADO, A.; BERMUDO, S.; LÓPEZ-SÁNCHEZ, A. D.; SÁNCHEZ-ORO, J. An iterated greedy algorithm for finding the minimum dominating set in graphs. **Mathematics and Computers in Simulation**, Elsevier, v. 207, p. 41–58, 2023.

CORCORAN, P.; GAGARIN, A. Heuristics for k-domination models of facility location problems in street networks. **Computers & Operations Research**, Elsevier, v. 133, p. 105368, 2021.

COUTURE, M.; BARBEAU, M.; BOSE, P.; KRANAKIS, E. Incremental construction of k-dominating sets in wireless sensor networks. **Adhoc & Sensor Wireless Networks**, v. 5, 2008.

ERICKSON, J. **Algorithms**. [S.l.: s.n.], 2023.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. **Journal of global optimization**, Springer, v. 6, p. 109–133, 1995.

FINK, J. F. n-domination in graphs. In: WILEY. **Graph theory with applications to algorithms and computer science**. [S.l.], 1985. p. 282–300.

GAGARIN, A.; CORCORAN, P. Multiple domination models for placement of electric vehicle charging stations in road networks. **Computers & Operations Research**, Elsevier, v. 96, p. 69–79, 2018.

GAGARIN, A.; POGHOSYAN, A.; ZVEROVICH, V. Randomized algorithms and upper bounds for multiple domination in graphs and networks. **Discrete Applied Mathematics**, Elsevier, v. 161, n. 4-5, p. 604–611, 2013.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). **Proceedings of the 7th Python in Science Conference**. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15.

KHOMAMI, M. M. D.; REZVANIAN, A.; BAGHERPOUR, N.; MEYBODI, M. R. Minimum positive influence dominating set and its application in influence maximization: a learning automata approach. **Applied Intelligence**, Springer, v. 48, p. 570–593, 2018.

KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

LAN, J. K.; CHANG, G. J. Algorithmic aspects of the k-domination problem in graphs. **Discrete Applied Mathematics**, Elsevier, v. 161, n. 10-11, p. 1513–1520, 2013.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, Elsevier, v. 3, p. 43–58, 2016.

MARA, S. T. W.; NORCAHYO, R.; JODIAWAN, P.; LUSIANTORO, L.; RIFAI, A. P. A survey of adaptive large neighborhood search algorithms and applications. **Computers & Operations Research**, Elsevier, v. 146, p. 105903, 2022.

MCKINNEY, W. et al. Data structures for statistical computing in python. In: AUSTIN, TX. **Proceedings of the 9th Python in Science Conference**. [S.l.], 2010. v. 445, p. 51–56.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & operations research**, Elsevier, v. 24, n. 11, p. 1097–1100, 1997.

NACHER, J. C.; AKUTSU, T. Minimum dominating set-based methods for analyzing biological networks. **Methods**, Elsevier, v. 102, p. 57–63, 2016.

OW, P. S.; MORTON, T. E. Filtered beam search in scheduling. **The International Journal Of Production Research**, Taylor & Francis, v. 26, n. 1, p. 35–62, 1988.

PAREKH, A. K. Analysis of a greedy heuristic for finding small dominating sets in graphs. **Information Processing Letters**, v. 39, n. 5, p. 237–240, 1991. ISSN 0020-0190. Disponível em: <https://www.sciencedirect.com/science/article/pii/0020019091900219>.

PISINGER, D.; ROPKE, S. A general heuristic for vehicle routing problems. **Computers & operations research**, Elsevier, v. 34, n. 8, p. 2403–2435, 2007.

PREDOJEVIĆ, M.; KARTELJ, A.; DJUKANOVIĆ, M. Variable neighborhood search for solving the k-domination problem. In: **Proceedings of the Companion Conference on Genetic and Evolutionary Computation**. [S.l.: s.n.], 2023. p. 239–242.

PUGLIESE, L. D. P.; FERONE, D.; FESTA, P.; GUERRIERO, F.; MACRINA, G. Solution approaches for the vehicle routing problem with occasional drivers and time windows. **Optimization Methods and Software**, Taylor & Francis, v. 37, n. 4, p. 1384–1414, 2022.

RANGEL, B. d. S. Um estudo de heurísticas para o problema da k-dominação. 2023.

REID, K. N.; LI, J.; BROWNLEE, A.; KERN, M.; VEERAPEN, N.; SWAN, J.; OWUSU, G. A hybrid metaheuristic approach to a real world employee scheduling problem. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. [S.l.: s.n.], 2019. p. 1311–1318.

ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. **Transportation science**, Informs, v. 40, n. 4, p. 455–472, 2006.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European journal of operational research**, Elsevier, v. 177, n. 3, p. 2033–2049, 2007.

SHAW, P. Using constraint programming and local search methods to solve vehicle routing problems. In: SPRINGER. **International conference on principles and practice of constraint programming**. [S.l.], 1998. p. 417–431.

THAI, M. T.; ZHANG, N.; TIWARI, R.; XU, X. On approximation algorithms of k-connected m-dominating sets in disk graphs. **Theoretical Computer Science**, Elsevier, v. 385, n. 1-3, p. 49–59, 2007.

WAINER, J. et al. Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. **Atualização em informática**, Sociedade Brasileira de Computação/Editora PUC Rio Rio de Janeiro, v. 1, n. 221-262, p. 32–33, 2007.

WEST, D. B. et al. **Introduction to graph theory**. [S.l.]: Prentice hall Upper Saddle River, 2001. v. 2.

ZÄPFEL, G.; BRAUNE, R.; BÖGL, M. **Metaheuristic search concepts: A tutorial with applications to production and logistics**. [S.l.]: Springer Science & Business Media, 2010.