

Vinícius Gandra Martins Santos

**Busca Adaptativa em Grandes Vizinhanças
Aplicada à Minimização da Largura de Corte
em Grafos**

Ouro Preto

2018

Vinícius Gandra Martins Santos

Busca Adaptativa em Grandes Vizinhanças Aplicada à Minimização da Largura de Corte em Grafos

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto para obtenção do título de Mestre em Ciência da Computação.

Universidade Federal de Ouro Preto - UFOP

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Orientador: Marco Antonio Moreira de Carvalho

Ouro Preto

2018

S591b

Santos, Vinícius Gandra Martins .

Busca adaptativa em grandes vizinhanças aplicada à minimização da largura de corte em grafos [manuscrito] / Vinícius Gandra Martins Santos. - 2018.
75f.: il.: grafs; tabs.

Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho.

Dissertação (Mestrado) - Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.

Área de Concentração: Ciência da Computação.

1. Heurística. 2. Teoria dos grafos. 3. Otimização combinatória. I. Carvalho, Marco Antonio Moreira de . II. Universidade Federal de Ouro Preto. III. Título.

CDU: 004.4\416



MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas – ICEB
Programa de Pós-Graduação em Ciência da Computação



Ata da Defesa Pública de Dissertação de Mestrado

Aos 23 dias do mês de novembro de 2018, às 16 horas na Sala de Seminários do DECOM no Instituto de Ciências Exatas e Biológicas (ICEB), reuniram-se os membros da banca examinadora composta pelos professores: **Prof. Dr. Marco Antonio Moreira de Carvalho (presidente e orientador)**, **Prof. Dr. Marcone Jamilson Freitas Souza** e **Prof. Dr. André Gustavo dos Santos**, aprovada pelo Colegiado do Programa de Pós-Graduação em Ciência da Computação, a fim de arguirem o mestrando **Vinícius Gandra Martins Santos**, com o título “**Busca Adaptativa em Grandes Vizinhanças Aplicada à Minimização da Largura de Corte em Grafos**”. Aberta a sessão pelo presidente, coube ao candidato, na forma regimental, expor o tema de sua dissertação, dentro do tempo regulamentar, sendo em seguida questionado pelos membros da banca examinadora, tendo dado as explicações que foram necessárias.

Recomendações da Banca:


☒ Aprovada sem recomendações

☐ Reprovada

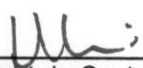
☐ Aprovada com recomendações: _____

Banca Examinadora:


Prof. Dr. Marco Antonio Moreira de Carvalho


Prof. Dr. Marcone Jamilson Freitas Souza


Prof. Dr. André Gustavo dos Santos


Prof. Dr. Joubert de Castro Lima
Coordenador do Programa de Pós-Graduação em Ciência da Computação
DECOM/ICEB/UFOP

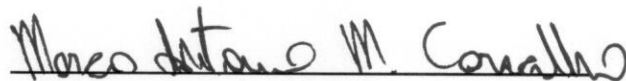
Ouro Preto, 23 de novembro de 2018.

Vinícius Gandra Martins Santos

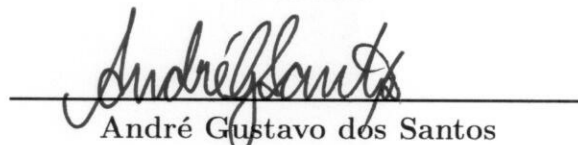
Busca Adaptativa em Grandes Vizinhanças Aplicada à Minimização da Largura de Corte em Grafos

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto para obtenção do título de Mestre em Ciência da Computação.

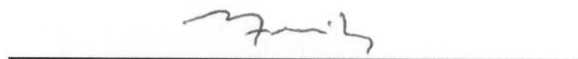
Trabalho aprovado. Ouro Preto, 23 de novembro de 2018:



Marco Antonio Moreira de Carvalho
Orientador



André Gustavo dos Santos
Membro externo



Marccone Jamilson Freitas Souza
Membro interno

Ouro Preto
2018

Este trabalho é dedicado ao compartilhamento do conhecimento, ao ensino público de qualidade, aos jovens cientistas do nosso país e aos trabalhadores de verdade.

Agradecimentos

Agradeço a todos os docentes e servidores da Universidade Federal de Ouro Preto, lugar onde aprendi e amadureci ao longo desses anos.

Agradeço ao meu orientador, Marco Antonio, por todos os seus ensinamentos e as longas horas dedicadas para concluir este trabalho.

Agradeço a minha companheira, a minha mãe e a todos meus amigos pelo carinho, companheirismo e troca de conhecimento que compartilhamos ao longo dessa fase.

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

Resumo

O problema de Minimização da Largura de Corte em Grafos (ou CMP, do inglês *Cutwidth Minimization Problem*) consiste em determinar um leiaute linear para um grafo de forma a minimizar a quantidade máxima de arestas que cruzam cada par de vértices consecutivos. Esse problema pode ser encontrado no projeto de circuitos integrados de larga escala, no desenho de diagramas de grafos e no projeto de compiladores, entre outros. O CMP é um problema NP-Difícil e se apresenta como um desafio para métodos exatos e heurísticas. Neste trabalho, é reportada pela primeira vez na literatura a aplicação do método metaheurístico Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*) para solução do CMP. Os experimentos computacionais envolvem 11.786 instâncias de quatro conjuntos da literatura e os resultados encontrados são comparados com o atual estado da arte. O método proposto se mostra competitivo, sendo capaz de igualar a maioria dos resultados comprovadamente ótimos e melhores resultados conhecidos, além de melhorar alguns resultados que não foram provados ótimos e encontrar pela primeira vez limitantes superiores para instâncias não resolvidas.

Palavras-chave: Heurísticas. Largura de Corte. Grafos. Otimização Combinatória.

Abstract

The cutwidth minimization problem (CMP) consists in determining a linear layout of the vertices of a graph that minimizes the maximum cardinality of edges cut between any consecutive pair of vertices. This problem has applications, for instance, in design of very large-scale integration circuits, graph drawing, and compiler design, among others. The CMP is an \mathcal{NP} -Hard problem and presents a challenge to exact methods and heuristics. In this study, the metaheuristic adaptive large neighborhood search is applied for the first time to the CMP. The computational experiments include 11,786 benchmark instances from the literature, and the obtained results are compared with those of the state-of-the-art methods. The proposed method was demonstrated to be competitive, as it matched most of proved optimal and best known results, improved some of the (not proved optimal) best known solutions, and provided the first upper bounds for unsolved instances.

Keywords: Heuristics. Cutwidth. Graphs. Combinatorial Optimization.

Declaração

Este documento é resultado de meu próprio trabalho, exceto onde referência explícita é feita ao trabalho de outros, e não foi submetida para outra qualificação nesta nem em outra universidade.

Vinícius Gandra Martins Santos

Lista de ilustrações

Figura 1	– (a) Grafo G com cinco vértices e seis arestas. (b) Ordem de rotulação π dos vértices do grafo G com as correspondentes larguras de cortes. (c) Ordem de rotulação π' com alteração na posição dos vértices b e d em relação a π	36
Figura 2	– (a) Grafo G com 9 vértices e 14 arestas; (b) decomposição de caminho do grafo G ; (c) decomposição em árvore do grafo G ; (d) disposição linear do grafo G . Extraído de Goldberg e Goldberg (2012).	38
Figura 3	– Fluxograma do funcionamento do ALNS.	41
Figura 4	– $CW'_\pi(u)$ é o custo de inserir cada vértice que está fora da solução ($u \in U$), π é a solução parcial construída pelo método de solução inicial e $\pi(w)$ é a posição de cada vértice presente na solução. As arestas pontilhadas mostram as possíveis conexões entre os vértices inseridos na solução e os vértices que estão fora da solução e possuem valor mínimo de CW'_π	46
Figura 5	– Grafo G com seis vértices e sete arestas disposto na ordem de rotulação π com as correspondentes larguras de cortes.	49
Figura 6	– Análise de convergência do ALNS para os conjuntos de instâncias Small, Grid e <i>Harwell-Boeing</i> (HB).	63
Figura 7	– Time-to-target plots de instâncias aleatoriamente selecionadas.	65

Lista de tabelas

Tabela 1	–	Valores considerados pelo <i>irace</i> para cada parâmetro.	57
Tabela 2	–	Resultados ALNS.	58
Tabela 3	–	Comparação com o estado da arte.	59
Tabela 4	–	Resultados ALNS para o conjunto <i>Rome Graphs</i>	60
Tabela 5	–	Comparação com o estado da arte CILP.	61
Tabela 6	–	Distância percentual entre as pontuações das heurísticas. As maiores pontuações tem <i>gap</i> zero.	61
Tabela 7	–	<i>P-value</i> do <i>Pairwise t-test</i> no conjunto de heurísticas de remoção. Valores em negrito indicam que existe diferença significativa entre o método da coluna e da linha.	62
Tabela 8	–	<i>P-value</i> do <i>Pairwise t-test</i> no conjunto de heurísticas de inserção. Valores em negrito indicam que existe diferença significativa entre o método da coluna e da linha.	62

Lista de abreviaturas e siglas

CMP	Minimização de largura de corte (do inglês, <i>Cutwidth Minimization Problem</i>)
ALNS	Busca adaptativa em grandes vizinhanças (do inglês, <i>Adaptive Large Neighborhood Search</i>)
HB	Conjunto de instâncias <i>Harwell-Boeing</i>
B&B	<i>Branch-and-bound</i>
CILP	Modelo de programação linear inteira do autor Coudert
GRASP	<i>Greedy Randomized Adaptive Search Procedure</i>
PR	<i>Path Relinking</i>
VNS	<i>Variable Neighborhood Search</i>
VFS	<i>Variable Formulation Search</i>
GAF	Algoritmo Genético <i>Fuzzy</i>
RVC	Remoção de vértices críticos
RRand	Remoção Aleatória
RSh _L C	Remoção de Vértices Relacionados por Largura de Corte
RSh _{grau}	Remoção de Vértices Relacionados por Grau
RD _P	Remoção de Vértices Desbalanceados (grau par)
RD _I	Remoção de Vértices Desbalanceados (grau ímpar)
RD	Remoção de Vértices Desbalanceados
RC _n	Remoção de Vértices Conectados
RdA _L	Remoção de Vértices direcionados por Arestas (esquerda)
RdA _R	Remoção de Vértices direcionados por Arestas (direita)
IRand	Inserção Aleatória
IB	Inserção na melhor posição Balanceada

IB _n	Inserção na melhor posição Balanceada com ruído
IBm	Inserção na melhor posição Balanceada com melhora
IBm _n	Inserção na melhor posição Balanceada com melhora e ruído
BKS	Melhor valor de solução da literatura
OPT	Valor ótimo da solução
SS	<i>Scatter Search</i>
gap	Distância percentual
ANOVA	Análise de variância
ttt-plot	<i>Time-to-Target plot</i>

Lista de símbolos

G	Grafo não dirigido
V	conjunto de vértices
E	conjunto de arestas
n	número de vértices de um grafo
m	número de arestas de um grafo
π	disposição linear ou rotulação dos vértices
$CW_{\pi}(G)$	valor da largura de corte máxima do grafo G na disposição linear π
$\pi(v)$	Posição do vértice v na rotulação π
$grau(v)$	quantidade de arestas incidentes ao vértice v na disposição linear π
$grauL_{\pi}(v)$	quantidade de arestas incidentes pela esquerda do vértice v na disposição linear π
$grauR_{\pi}(v)$	quantidade de arestas incidentes pela direita do vértice v na disposição linear π
π_0	Solução inicial
π^*	Melhor solução encontrada
N^-	Conjunto de heurísticas de remoção
N^+	Conjunto de heurísticas de inserção
σ_1	pontuação atribuída para heurísticas
σ_2	pontuação atribuída para heurísticas
σ_3	pontuação atribuída para heurísticas
ρ	Fator de reação $\in (0,1)$.
θ	Número de segmentos
r_i	pontuação acumulada da heurística i no último segmento
a_i	número de vezes que a heurística i foi chamada no último segmento

$w_{i,j}$	peso da heurística i no segmento j
T	Temperatura
T_{start}	Temperatura inicial
T_{end}	Temperatura final
p_s	Percentual de piora inicial
p_e	Percentual de piora final
S^*	Melhor valor de solução do ALNS
S	Valor médio de solução
S_0	Valor de solução inicial
σ	Desvio padrão
$T(s)$	Tempo de execução em segundos

Sumário

1	INTRODUÇÃO	25
1.1	Justificativa	26
1.2	Objetivos	27
1.3	Organização do texto	27
2	REVISÃO BIBLIOGRÁFICA	29
3	FUNDAMENTAÇÃO TEÓRICA	35
3.1	O problema de largura de corte	35
3.2	Busca adaptativa em grandes vizinhanças	38
4	DESENVOLVIMENTO	45
4.1	Solução inicial	45
4.2	Ruído	47
4.3	Critério de aceitação	48
4.4	Crítérios de parada	49
4.5	Heurísticas de remoção	49
4.5.1	Remoção de vértices críticos	50
4.5.2	Remoção aleatória	50
4.5.3	Remoção de vértices relacionados	50
4.5.4	Remoção de vértices desbalanceados	51
4.5.5	Remoção de vértices adjacentes	51
4.5.6	Remoção de vértices direcionada por arestas	51
4.5.7	Análise de complexidade do pior caso das heurísticas de remoção	52
4.6	Heurísticas de inserção	52
4.6.1	Inserção aleatória	52
4.6.2	Inserção na melhor posição balanceada	52
4.6.3	Inserção na melhor posição balanceada com melhora	53
4.6.4	Análise de complexidade do pior caso das heurísticas de inserção	53
4.7	Pós-processamento	54
5	EXPERIMENTOS COMPUTACIONAIS	55
5.1	Conjuntos de Instâncias	55
5.2	Experimentos Preliminares	56
5.3	Comparação com os Resultados da Literatura	57
5.4	Análises Adicionais	61

6	CONCLUSÃO	67
	REFERÊNCIAS	69

1 Introdução

Um *grafo* é uma abstração de modelagem matemática que representa a relação entre elementos de um determinado conjunto. Denotado por $G = (V, E)$, um grafo é composto por dois conjuntos, V e E , em que V é um conjunto não vazio de n elementos denominados *vértices* e E o conjunto de m pares de elementos de V que dão origem a elementos chamados *arestas*. Um grafo pode ainda ser *direcionado*, ou seja, os elementos de E são pares ordenados de vértices, dando origem a *arcos* que possuem direção específica de percurso. Caso contrário, o grafo é *não direcionado*. Uma aresta denota uma relação de *adjacência* entre os vértices aos quais incide, o que pode representar quaisquer tipos de relações arbitrárias entre os elementos representados pelos vértices. Para refletir estas relações precisamente, as arestas podem possuir um valor associado, caso no qual denota-se o grafo como *ponderado*.

As estruturas de adjacências de um grafo definem a sua topologia, a qual fornece diferentes propriedades e subestruturas. Uma estrutura topológica básica se refere à capacidade de os vértices alcançarem uns aos outros por meio do percurso de suas adjacências. Havendo esta estrutura, o grafo é considerado *conexo*, caso contrário, o grafo é *desconexo*. Uma topologia específica é a denominada *árvore*, na qual em um grafo conexo há n vértices e $n-1$ arestas. Uma propriedade comum dos vértices é denotada pelo *grau*, que consiste no número de arestas incidentes a cada vértice.

Uma propriedade particularmente importante de grafos é a conhecida como *isomorfismo*, em que dois grafos são isomorfos entre si se existe uma correspondência entre os seus vértices e arestas de tal maneira que as relações de incidência sejam preservadas. De outra maneira, o isomorfismo estabelece que um mesmo grafo pode ser representado de maneiras diferentes, desde que sua estrutura de adjacências permaneça preservada.

Problemas de determinação de leiaute em grafos são uma classe particular de problemas de otimização combinatória. Estes problemas consistem em dispôr linearmente os vértices de um dado grafo não direcionado, mantendo as relações de adjacências entre os vértices e de forma a otimizar uma função objetivo específica. Um leiaute linear para um grafo equivale a uma rotulação de cada um de seus vértices com inteiros distintos, em que o inteiro atribuído para o vértice consiste na posição ocupada pelo mesmo no leiaute linear.

Entre os problemas de determinação de leiaute em grafos podemos destacar um subconjunto de problemas que consistem em minimizar um valor máximo. Deste subconjunto cita-se como exemplo os problemas *Bandwidth* (HARPER, 1966), *Antibandwidth* (LEUNG; VORNBERGER; WITTHOFF, 1984), *Vertex Separation* (LIPTON; TARJAN,

1979), *Profile Minimization* (LIN; YUAN, 1994), *Pathwidth* (BODLAENDER; KLOKS, 1996), *Backplane Ordering* (COHOON; SAHNI, 1987) e *Cutwidth* (ADOLPHSON; HU, 1973). Alguns desses problemas estão intimamente relacionados, de maneira que o valor da solução de alguns são limitantes superiores e inferiores para o valor da solução de outros. Uma revisão abrangente sobre problemas de leiaute em grafos é apresentada por Díaz, Petit e Serna (2002).

Este trabalho é focado no problema de Minimização de Largura de Corte (do inglês *Cutwidth Minimization Problem*, CMP). Dado um grafo não dirigido, o CMP consiste em determinar a disposição linear de seus vértices, tal que o número máximo de arestas entre cada par de vértices consecutivos é minimizado. Este problema já foi previamente abordado de forma exata e também de forma heurística. Neste trabalho será dado destaque a abordagens heurísticas como forma de solução do CMP.

A metaheurística Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS), proposta por Ropke e Pisinger (2006), é composta pela combinação de heurísticas de inserção e remoção que são usadas de modo adaptativo a depender do desempenho de cada uma. Essas heurísticas costumam ser desenvolvidas sob medida para o problema e podem ser combinadas para gerar vários procedimentos de busca locais, que, por sua vez, permitem a exploração de uma porção ampla das possíveis soluções. Neste trabalho é proposta uma implementação do ALNS juntamente com novas heurísticas e um método de solução inicial inspirado em Pantrigo et al. (2012).

1.1 Justificativa

Um grande número de problemas pode ser formulado como problemas de leiautes em grafos, como por exemplo, o problema de desenho de grafos (SUDERMAN, 2004), processamento de linguagem natural (KORNAI; TUZA, 1992), o problema de agendamento de migração de redes (ANDRADE; RESENDE, 2007b), o projeto de circuitos integrados de larga escala (OHTSUKI et al., 1979) e recuperação de informação (BOTAFOGO, 1993), entre outros.

Em sua versão de decisão, o CMP foi provado ser NP-Completo por Garey, Johnson e Stockmeyer (1976) e por Gavril (1977). O problema continua sendo NP-Completo para grafos com grau máximo 3 (MAKEDON; PAPADIMITRIOU; SUDBOROUGH, 1985), grafos planares com grau máximo 3 (MAKEDON; SUDBOROUGH, 1989), grafos grade e grafos disco unitários (DIAZ et al., 2001), grafos direcionados acíclicos (EVEN; SHILOACH, 1975) e para árvores ponderadas (MONIEN; SUDBOROUGH, 1988). Na versão de otimização, o CMP foi provado ser NP-Difícil mesmo para grafos de grau máximo três (MAKEDON; PAPADIMITRIOU; SUDBOROUGH, 1985).

Sendo assim, este trabalho, que trata um problema NP-Difícil (MAKEDON; PAPA-

DIMITRIOU; SUDBOROUGH, 1985), pode contribuir para gerar conhecimento e inovação para diversas áreas de pesquisa, por exemplo, engenharias e ciência da computação.

1.2 Objetivos

Este trabalho tem como objetivo geral elaborar uma heurística consistente e eficaz que permita a obtenção de soluções próximas das soluções ótimas para ser utilizada no problema de largura de corte. São objetivos específicos:

- Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre o CMP;
- Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre o ALNS;
- Avaliar o método proposto, testando-o em problemas teste publicamente disponíveis, assim como em casos reais, e comparar seus resultados com os de outros métodos da literatura.

Além dos objetivos principais, outros produtos deste projeto de pesquisa são trabalhos publicados em periódicos e conferências.

1.3 Organização do texto

O restante deste trabalho está organizado como descrito a seguir. O Capítulo 2 apresenta uma revisão detalhada dos métodos de solução para o CMP, seus resultados e contribuições principais. Em seguida, no Capítulo 3, o problema é descrito formalmente. Um novo método é apresentado para a solução do CMP juntamente com as minúcias de sua implementação no Capítulo 4. Os testes computacionais e os resultados obtidos estão descritos no Capítulo 5. Por fim, o Capítulo 6 aponta direções para os trabalhos futuros e apresenta a conclusão do trabalho.

2 Revisão Bibliográfica

Neste capítulo é apresentada uma revisão da literatura relacionada aos métodos propostos para solução do CMP. São apresentados também trabalhos que consideram casos especiais do CMP, como por exemplo, topologias específicas e a versão parâmetro fixo tratável, entre outros.

Algumas topologias específicas de grafos permitem formulações que resolvam o CMP em tempo determinístico polinomial. Por exemplo, [Lengauer \(1982\)](#) propôs um algoritmo linear para o CMP em árvores completas cujos vértices possuem grau fixo. [Yannakakis \(1985\)](#) propôs um algoritmo $O(n \log n)$ para árvores não ponderadas. No mesmo ano, um algoritmo com a mesma complexidade foi proposto por [Chung et al. \(1985\)](#) para árvores de grau três. Ainda em árvores, [Thilikos, Serna e Bodlaender \(2001\)](#) propuseram um algoritmo para árvores parciais com complexidade $n^{O((wd)^2)}$, em que w é o valor da largura de corte e d é o número máximo de filhos de um vértice raiz.

[Harper \(1964\)](#) propôs um algoritmo linear para hipercubos, um análogo n -dimensional do quadrado e do cubo formado por grupos de segmentos paralelos que formam ângulos retos com os outros segmentos do mesmo tamanho. Um algoritmo quadrático foi proposto por [Rolim, Sýkora e Vrt'ó \(1995\)](#) para malhas de duas e três dimensões, malhas toroidais e cilíndricas de duas dimensões e malhas toroidais de 3 dimensões.

Além dos casos das topologias mencionadas, é possível determinar em tempo determinístico polinomial se a largura de corte é no máximo k , dado que k é um parâmetro fixo. O primeiro algoritmo linear para o CMP- k foi proposto por [Garey et al. \(1978\)](#), o qual determina se a largura de corte é no máximo 2. Na década de 80, [Gurari e Sudborough \(1984\)](#) propuseram um algoritmo $O(n^k)$, para todo inteiro positivo k . Em seguida, [Makedon e Sudborough \(1989\)](#) utilizaram programação dinâmica e propuseram um algoritmo de complexidade $O(n^{k-1})$, para cada inteiro fixo $k \geq 2$. Posteriormente, [Fellows e Langston \(1992\)](#) propuseram um algoritmo quadrático para o mesmo fim. O melhor algoritmo até então foi proposto por [Thilikos, Serna e Bodlaender \(2000\)](#) e determina se a largura de corte é no máximo k em tempo linear $O(n)$.

Algoritmos aproximados para o CMP foram propostos por [Leighton e Rao \(1999\)](#) e [Arora, Frieze e Kaplan \(2002\)](#). O primeiro trabalho foi proposto para grafos gerais e consiste em um algoritmo de aproximação polinomial com relação de aproximação de $O(CW \log^2 n)$, em que CW é o valor ótimo ou limite inferior da largura de corte. O segundo trabalho propôs um algoritmo polinomial aproximado para grafos densos, i.e., em que há a relação $m \gg n$. Este algoritmo possui complexidade polinomial para o tempo de execução, $n^{O(\log n / \epsilon^2)}$ para $\epsilon > 0$, e relação de aproximação de $\epsilon \times n^2$.

Alguns dos trabalhos mencionados a seguir tratam a versão de otimização padrão do CMP em grafos gerais e utilizam em seus experimentos computacionais três conjuntos de instâncias principais: *Small* (MARTÍ; CAMPOS; PIÑANA, 2008) composto por 84 instâncias, *Grid* (RASPAUD; SYKORA; VRT'O, 1995) composto por 81 instâncias e *Harwell-Boeing* (HB), um subconjunto derivado do *Harwell-Boeing Sparse Matrix Collection* (BOISVERT et al., 1997), composto por 87 instâncias. O primeiro é considerável pequeno com instâncias com até 24 vértices, o segundo é composto por grafos com até 729 vértices e o terceiro considerado o conjunto com maiores dimensões possui instâncias com até 700 vértices e 41.000 arestas. Os três conjuntos estão disponíveis em Martí et al. (2010). Outro conjunto de instâncias, *Rome Graphs*, é mais recente e contém 11.534 instâncias com até 100 vértices e pode ser encontrado em Coudert (2016a).

Uma adaptação do método *branch and bound* (B&B) foi proposto por Palubeckis e Rubliauskas (2012). O método engloba um teste de dominância para reduzir o espaço de busca através de uma busca tabu. Como resultado, o método é capaz de encontrar valores ótimos para instâncias com até 35 vértices para grafos densos, e instâncias com até 50 vértices para grafos esparsos. Subsequentemente, Martí et al. (2013) propuseram outro método derivado do B&B e utilizaram 112 instâncias retiradas dos três conjuntos de instâncias, *Small*, *Grid* e HB, para os experimentos computacionais. O método proposto conseguiu soluções ótimas para todas as 42 instâncias retiradas do conjunto *Small* e mais 33 soluções ótimas das 36 instâncias retiradas do conjunto *Grid*. Para o terceiro conjunto o método não teve bom desempenho encontrando apenas 9 soluções ótimas para as 34 instâncias selecionadas.

López-Locés et al. (2014) propuseram um modelo de programação inteira. O modelo foi testado com 13 instâncias selecionadas dentre os conjuntos *Small*, *Grid* e HB. Os autores compararam os resultados deste trabalho com os resultados de outro modelo inteiro proposto por eles anteriormente e obtiveram 7 soluções ótimas, 38% a mais que o último modelo inteiro. Posteriormente, Coudert (2016b) propôs um novo modelo de programação inteira, neste trabalho referido como CILP. O modelo foi testado com dois conjunto de instâncias, *Small* e *Rome Graphs* e alcançou soluções com valores ótimos para todas as instâncias do primeiro conjunto. Para o segundo conjunto, o modelo conseguiu determinar as soluções ótimas para 49% das instâncias. Especificamente, o método foi capaz de gerar as soluções ótimas para 99,01% dos grafos com $n + m < 90$, 72,69% dos grafos com $90 < n + m < 120$ e 0,74% dos grafos com $n + m > 120$. Os resultados demonstram que o modelo de Coudert tem melhor desempenho do que o modelo de López-Locés et al. (2014). Algoritmos exatos são eficientes somente para instâncias pequenas, não sendo possível até o momento encontrar o valor ótimo em tempo praticável para instâncias maiores.

Métodos heurísticos e metaheurísticos foram amplamente utilizados, principalmente

na literatura mais recente, para a solução do CMP e problemas equivalentes. Métodos baseados no *Greedy Randomized Adaptive Search Procedure* (GRASP) foram propostos por [Andrade e Resende \(2007a\)](#) e [Andrade e Resende \(2007b\)](#). O primeiro método parte de uma solução inicial gerada por uma busca em profundidade. Em seguida, cada vértice é selecionado na ordem de exploração da busca em profundidade e inserido na melhor posição possível (*best insertion*). A busca local aplicada é baseada em trocas de pares de elementos (*2-swap*) em um esquema de primeira melhora (*first improvement*). O processo de *Path Relinking* (PR) gera soluções vizinhas que estão entre a solução corrente e uma solução alvo. O processo acaba quando a solução corrente é igual a solução alvo. O segundo método difere do primeiro por uma nova metodologia aplicada chamada *Evolutionary Path Relinking*, que com a aplicação do GRASP gera um conjunto de soluções elite e aplica o PR em cada uma dessas soluções. O método não é capaz de encontrar todos os valores ótimos para o conjunto *Small* e para os conjuntos *Grid* e HB menos de 10% dos valores ótimos foram encontrados.

Um algoritmo híbrido evolutivo foi proposto por [Bansal, Srivastava e Srivastava \(2012\)](#), em que a solução inicial do algoritmo genético é gerada através de uma busca em profundidade aleatorizada. Os experimentos utilizaram grafos padrão, por exemplo, grafo bipartido completo, malhas de duas e três dimensões e grafo hipercubo, além de grafos conectados gerados de forma aleatória. O trabalho foi comparado com outro algoritmo genético que possui alguns parâmetros diferentes e apresentou bons resultados principalmente para os grafos padrão.

No mesmo ano, um método baseado em *Scatter Search*, proposto por [Pantrigo et al. \(2012\)](#), obteve os melhores resultados até então para as instâncias *Small*, *Grid* e HB. O algoritmo utiliza o método GRASP com duas políticas de escolha gulosa para gerar um conjunto de soluções iniciais e em seguida realiza um processo evolucionário em quatro etapas: geração de um subconjunto de soluções; combinação das soluções do subconjunto utilizando estratégia de votos; melhoria, através de uma busca local focada em movimentos de inserção; e atualização do subconjunto através de regras de aceitação. O *Scatter Search* alcançou todos os valores ótimos das instâncias do conjunto *Small*, além de 44 soluções com valores ótimas no conjunto *Grid* e 59 no HB, com tempo médio de execução de 213 segundos.

Subsequentemente, métodos baseados na metaheurística *Variable Neighborhood Search* (VNS) foram propostos por [Duarte et al. \(2012\)](#), [Lozano et al. \(2012\)](#), [Pardo et al. \(2013\)](#), [Duarte et al. \(2013\)](#), [Sánchez-Oro, Pantrigo e Duarte \(2014\)](#). Dentre estes, destaca-se o algoritmo *Variable Formulation Search* (VFS) proposto por [Pardo et al. \(2013\)](#) que compõe o estado da arte para o CMP.

Além das estratégias clássicas da metodologia do VNS, que consistem em busca local, perturbação, e mudança de vizinhança, o VFS considera duas metodologias diferentes

para distinguir soluções que inicialmente possuem o mesmo valor de função objetivo. Essa técnica permite lidar com espaços de busca em que a superfície contém platôs. Desta forma, o algoritmo é capaz de diferenciar entre duas soluções com o mesmo valor de função objetivo e escolher a mais promissora para continuar a busca. A primeira formulação considera que uma solução A é melhor que uma solução B se a solução A apresentar menor número de *gargalos*, ou seja, menor quantidade de cortes com o valor de largura de corte máximo. A segunda formulação considera a distância entre as soluções e um limite inferior, tal que a solução com a menor distância é selecionada. O VFS considera o conjunto de instâncias *Grid* e HB, e compara os resultados obtidos com o método GRASP e *Scatter Search* (ANDRADE; RESENDE, 2007a; PANTRIGO et al., 2012). O algoritmo proposto supera as soluções e o tempo dos algoritmos comparados. Adicionalmente, o VFS foi capaz de encontrar 59 valores ótimos para o conjunto *Grid* com tempo médio de 90 segundos e outros 61 valores ótimos para o conjunto HB com uma média de 120 segundos.

Outra variação de VNS, proposta por Duarte et al. (2013), consiste em uma implementação paralela realizada em quatro abordagens diferentes: paralelizar apenas a fase de perturbação, paralelizar apenas fase de busca local, paralelizar apenas a aplicação do conjunto de vizinhanças e, por fim, paralelizar o VNS como um todo, ou seja, todas as etapas do método. A fase de busca local consiste em selecionar vértices candidatos e inseri-los em novas posições de forma a melhorar a solução objetivo. Em contrapartida, a mudança de vizinhança consiste em incrementar um parâmetro K , usado na fase de perturbação para definir a quantidade de pares de vértices que serão trocados de posição. Os experimentos computacionais compararam as diferentes versões do VNS entre si e consideraram 101 grafos retirados do *Harwell-Boeing Sparse Matrix Collection* com dimensões variando entre $30 \leq n \leq 3025$ vértices. A melhor abordagem foi a que paraleliza a fase de perturbação, com um total de 92 melhores resultados. Esta abordagem foi comparada ao VFS de Pardo et al. (2013), obtendo melhores resultados no conjunto de instâncias utilizado. No entanto, não há experimentos disponíveis considerando o conjunto de instâncias *Grid*, o que permitiria uma melhor comparação entre os métodos.

O método metaheurístico mais recente para o CMP, um Algoritmo Genético *Fuzzy* (GAF), foi proposto por Fraire-Huacuja et al. (2017) e usa um controlador com lógica *Fuzzy* para determinar os parâmetros do algoritmo durante a execução. O método proposto utilizou em seus experimentos as instâncias *Small* e *Grid*, e foi comparado com uma versão do mesmo algoritmo genético sem o emprego de lógica *Fuzzy*. O GAF apresentou um desempenho levemente melhor do que o mesmo algoritmo sem a lógica *Fuzzy*.

O GAF, além de possuir um maior tempo de execução, se comparado com o VFS e o *Scatter Search*, não foi capaz de encontrar todas as soluções ótimas para o conjunto *Small*, que é considerado trivial por possuir instâncias pequenas. O GAF também obteve um baixo desempenho para as instâncias do conjunto *Grid*, com uma distância percentual

de mais de 85% se comparado com o resultado encontrado pelo VFS.

Finalmente, neste trabalho os conjuntos *Small*, Grid, HB e *Rome Graphs* são considerados para os experimentos computacionais. Além disso, os resultados encontrados são comparados com os métodos VFS e CILP considerados estado da arte.

3 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica a respeito do problema de minimização da Largura de Corte e do método disponível na literatura que é utilizado neste trabalho. Nas seções seguintes são apresentados uma definição formal e exemplos do CMP. Em seguida é apresentada a definição do método ALNS usado como método de solução neste trabalho.

3.1 O problema de largura de corte

Em termos matemáticos, dado um grafo não dirigido $G = (V, E)$ com $|V| = n$ e $|E| = m$, uma ordem de rotulação ou disposição linear π de G atribui os inteiros $[1, 2, \dots, n]$ aos vértices em V , em que cada vértice recebe um inteiro diferente. Sem que haja perda de generalidade, dada uma disposição linear a largura de corte entre dois vértices consecutivos pode ser considerada como a largura de corte do vértice a esquerda. A posição de um vértice v considerando a ordem de rotulação π é dada por $\pi(v)$ e a largura de corte, $CW_\pi(v)$, é o número de arestas $\{u, w\} \in E$ que satisfaz $\pi(u) \leq \pi(v) < \pi(w)$. A largura de corte do grafo $CW_\pi(G)$ é o máximo da largura de corte dos seus vértices, vide Equação (3.1).

$$CW_\pi(G) = \max_{v \in V} CW_\pi(v). \quad (3.1)$$

A largura de corte ótima de um grafo G é definido pelo mínimo valor de $CW_\pi(G)$ entre todos os possíveis conjuntos de rótulos. Em outras palavras, o CMP consiste em encontrar uma disposição linear π que minimize $CW_\pi(G)$ dentre todas as possíveis permutações de rótulos com tamanho n , dadas por Π_n . A Equação (3.2) demonstra formalmente a função objetivo.

$$CW(G) = \min_{\pi \in \Pi_n} CW_\pi(G). \quad (3.2)$$

O grau de um vértice v é denotado por $\text{grau}(v) = |\{u \in V : \{u, v\} \in E\}|$. Em uma disposição linear cada vértice pode conectar com outros vértices localizados em alguma posição a sua esquerda ou direita. Similar à notação adotada por [Pardo et al. \(2013\)](#), a quantidade de arestas incidentes pela esquerda de um vértice v na disposição linear π é dado por $\text{grau}L_\pi(v) = |\{u \in V : \{u, v\} \in E \wedge \pi(v) > \pi(u)\}|$, e de forma simétrica a quantidade de arestas incidentes pela direita é representado por $\text{grau}R_\pi(v) = |\{u \in V : \{u, v\} \in E \wedge \pi(v) < \pi(u)\}|$, de forma que $\text{grau}(v) = \text{grau}R_\pi(v) + \text{grau}L_\pi(v)$.

Na Figura 1(a) é apresentado um grafo contendo cinco vértices ligados por seis arestas. A Figura 1(b) apresenta o mesmo grafo disposto em uma linha horizontal. Note que $\pi(e) = 1$, o que quer dizer que o vértice e se encontra na primeira posição da disposição linear π , seguido pelo vértice d ($\pi(d) = 2$) e assim por diante. Desta maneira, π pode ser representada por $[e, d, b, a, c]$. A largura de corte de cada vértice é representada pela linha pontilhada com o respectivo valor do corte. Por exemplo, o corte de a é dado por $CW(a) = 2$, o que significa que existem duas arestas com ponto inicial antes do vértice a e com ponto final depois do vértice a ($\{a, c\}$, $\{d, c\}$). Como a largura de corte $CW_\pi(G)$ é dada pelo maior corte entre todos os vértices V , nesse exemplo em particular, $CW_\pi(G) = CW_\pi(d) = 4$. Podemos dizer que o vértice d é o *vértice crítico* ou *gargalo* dessa disposição linear. A Figura 1(c) apresenta uma ordem de rotulação π' em que os vértices b e d foram trocados de lugar e a largura de corte diminui em uma unidade, i.e., $CW_{\pi'}(G) = 3$.

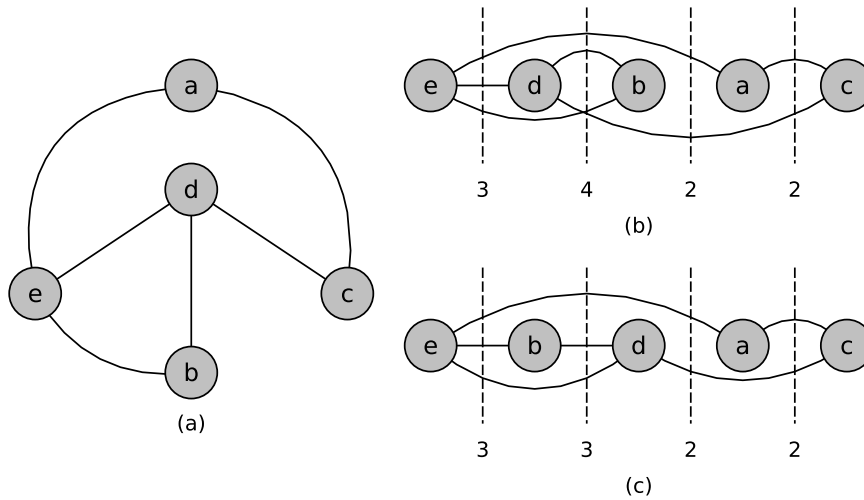


Figura 1 – (a) Grafo G com cinco vértices e seis arestas. (b) Ordem de rotulação π dos vértices do grafo G com as correspondentes larguras de cortes. (c) Ordem de rotulação π' com alteração na posição dos vértices b e d em relação a π .

Limitantes inferiores e superiores de boa qualidade podem delimitar o espaço de busca e são importantes para problemas em que não é possível desenvolver um algoritmo exato que possua tempo polinomial de computação. A utilidade destes se dá por representarem um valor próximo ou igual ao valor da solução ótima para instâncias de problemas de otimização. Os resultados encontrados por metaheurísticas, que não possuem propriedade de corretude, podem ser comparados com esses limitantes para inferir a qualidade das soluções geradas.

Para o problema de largura de corte, limitantes são apresentados para grafos com grau máximo conhecido (SYKORA; VRT'Ů, 1993), grafos de *Bruijn* (RASPAUD;

(SYKORA; VRT'O, 1995) e malhas de d dimensões em árvores r -árias (VRT'O, 2000). Para grafos gerais é possível determinar um limitante inferior ao encontrar o fluxo máximo. É necessário gerar uma matriz M simétrica $n \times n$, em que $M[i, j]$ representa o fluxo máximo do vértice i para o vértice j , de forma que o fluxo máximo encontrado corresponde a largura de corte mínima. Gomory e Hu (1961), representam a matriz de fluxo máximo através de uma árvore geradora, em que cada aresta representa uma possível largura de corte e o peso das arestas representam a largura de corte mínimo entre dois vértices. Para encontrar a menor largura de corte é necessário determinar o fluxo máximo entre cada par de vértices na árvore geradora. Desta forma, o maior fluxo encontrado na árvore geradora representa a largura de corte mínima no grafo original.

Soluções de problemas relacionados também podem ser provados como limitantes para o valor da largura de corte em grafos. Por exemplo, o valor ótimo da largura de corte em um grafo qualquer é no máximo o valor da *largura de caminho* (ou *Pathwidth*) multiplicado pelo grau máximo entre os vértices do grafo, e no mínimo o valor da *largura arbórea* (ou *Tree-width*) (KORACH; SOLEL, 1993).

Uma decomposição de caminho de um grafo $G = (V, E)$ é uma sequência de subconjuntos X_i de V , em que para cada aresta de G existe um i de forma que as duas extremidades da aresta pertencem ao subconjunto X_i , e para cada três índices $i \leq j \leq k$ temos $X_i \cap X_k \subseteq X_j$, ou seja, cada vértice aparece em uma subsequência contígua dos subconjuntos. A largura de caminho por sua vez é a cardinalidade do maior destes conjuntos menos um. A largura arbórea é caracterizada de maneira análoga, no entanto, a decomposição é feita em topologia de árvore. Uma decomposição em árvore de um grafo $G = (V, E)$ é dada por um pares $(\{X_i | i \in I\}, T)$, em que X_i é também um subconjunto de V , e T é uma árvore com os elementos de I como vértices (GOLDBARG; GOLDBARG, 2012). Para a decomposição em árvore, três propriedades devem ser verificadas: (i) $\cup_{i \in I} X_i = V$; (ii) para toda aresta $\{u, v\} \in E$ existe $i \in I$ tal que $u, v \in X_i$; (iii) para todo $i, j, k \in I$, se j pertence a um caminho entre i e k em T então $X_i \cap X_k \subseteq X_j$. A largura de árvore, analogamente ao caso anterior, é dada pela cardinalidade do maior destes conjuntos menos um.

A Figura 2(a) apresenta um grafo de exemplo em que é realizada (b) uma decomposição de caminho, (c) uma decomposição em árvore, e (d) apresenta uma disposição linear. A decomposição de caminho gera uma largura de caminho igual a quatro, enquanto a decomposição em árvore gera uma largura arbórea de valor igual a três. A disposição linear, por sua vez, gera uma largura de corte igual a cinco.

Além de problemas relacionados, existem problemas equivalentes que possuem a mesma função objetivo que o CMP, como é o caso do *Backplane Ordering* (COHOON; SAHNI, 1987). Neste problema cada vértice pode ser interpretado como um *board*, um componente eletrônico, e as arestas são redes (*nets*) que conectam tais componentes.

demandar maior tempo computacional. Neste cenário, os métodos heurísticos são utilizados para gerar soluções de qualidade com custo computacional razoável. É natural pensar que é necessário um método com heurísticas eficazes e formas para escapar de ótimos locais.

Ropke e Pisinger (2006) introduziram a metaheurística Busca Adaptativa em Grandes Vizinhanças tendo como base o método Busca em Grandes Vizinhanças (*Large Neighborhood Search*) proposta por Shaw (1998). O ALNS foi proposto originalmente para resolver o Problema de Roteamento de Veículos com Coleta e Entrega com Janelas de Tempo, cuja solução é representada por uma permutação de elementos, assim como o CMP.

O ALNS utiliza um conjunto N^- de heurísticas de remoção, que removem elementos da solução, e um conjunto N^+ de heurísticas de inserção, que reinserem elementos e constroem uma nova solução dada uma solução parcial. Essas heurísticas são geralmente baseadas em métodos gulosos de bom desempenho para o problema em questão. Uma busca local é composta por uma heurística de remoção e uma heurística de inserção de elementos. O ALNS, por sua vez, possui diversas dessas heurísticas, podendo assim combiná-las para criar variadas buscas locais que induzem diferentes vizinhanças e que competem entre si para modificar uma solução. A escolha dessas buscas locais influencia diretamente no resultado do ALNS. Sendo assim, é de suma importância que a escolha de qual busca local utilizar seja feita de forma eficiente.

O ALNS utiliza um método estatístico baseado nas iterações anteriores do algoritmo para decidir a prioridade da aplicação das heurísticas. A ideia é observar o desempenho de cada heurística durante blocos de iterações de tamanho θ , chamados *segmentos*, e ao final de cada bloco atribuir um *peso*, de acordo com o desempenho anterior, para as heurísticas. Por exemplo, se a metaheurística for executada por 500 iterações, podemos ter 5 segmentos em que $\theta = 100$.

Os pesos são ajustados de forma adaptativa de acordo com a *pontuação* acumulada ao longo dos segmentos. Após a aplicação de um par de heurísticas, uma heurística de remoção e outra de inserção, uma determinada pontuação é atribuída às heurísticas de acordo com seu desempenho. Essa pontuação é classificada em três categorias:

- σ_1 , quando a última remoção ou inserção resultou no melhor resultado até o momento;
- σ_2 , quando a última remoção ou inserção resultou em uma solução cujo custo seja menor que o da solução atual; e
- σ_3 , quando a última remoção ou inserção resulta em uma solução que é aceita por um critério de aceitação, porém, com o custo maior que o da solução atual.

No final do segmento j , a pontuação acumulada de cada heurística é usada para definir os pesos para o próximo segmento $j + 1$. No entanto, a pontuação passa por um

processo de suavização para diminuir a discrepância entre as pontuações observadas. Sendo r_i a pontuação acumulada do operador i durante o último segmento e a_i o número de vezes que a heurística i foi executada durante o último segmento, o processo de suavização de r_i é feito através da divisão de r_i por a_i , em que $a_i \neq 0$.

Ainda no processo de calcular o peso das heurísticas, o fator de reação $\rho \in (0, 1)$ é usado para controlar o quão rápido o ajuste dos pesos reage às mudanças na pontuação de cada heurística. Quando $\rho = 1$ os novos pesos são baseados somente na pontuação do segmento imediatamente anterior. Por outro lado, quando $\rho < 1$, os pontos dos segmentos anteriores são todos levados em consideração. Considerando $w_{i,j}$ o peso da heurística i durante o segmento j , a Equação (3.3) apresenta o cálculo para os pesos do próximo segmento $w_{i,j+1}$. Caso a heurística i não tenha sido executada nenhuma vez durante o segmento, seu peso permanecerá o mesmo do segmento imediatamente anterior.

$$w_{i,j+1} = \begin{cases} w_{i,j} & \text{if } a_i = 0. \\ \rho \frac{r_i}{a_i} + (1 - \rho)w_{i,j} & \text{if } a_i \neq 0. \end{cases} \quad (3.3)$$

A cada iteração o ALNS seleciona uma heurística de remoção e uma outra de inserção. A seleção é feita através do método da *roleta*, representada por um intervalo $R = [0...1] \in \mathbb{R}$, em que cada heurística i recebe uma fatia da roleta proporcional à sua probabilidade p_i de ser escolhida. A probabilidade de selecionar uma heurística i em um segmento j é proporcional ao seu peso $w_{i,j}$ e calculada como $p_i = \frac{w_{i,j}}{\sum_{k=1}^{|N^-|} w_{k,j}}$ para heurísticas de remoção e $p_i = \frac{w_{i,j}}{\sum_{k=1}^{|N^+|} w_{k,j}}$ para heurísticas de inserção. Heurísticas com maior peso possuem uma maior probabilidade de serem selecionadas.

Considere uma roleta com 4 heurísticas com os pesos 60, 125, 115 e 200 para as heurísticas 1, 2, 3 e 4, respectivamente. A heurística 4 possui 40% de probabilidade de ser selecionada, logo, possui a maior fatia na roleta, representada no intervalo entre 0,6 e 1.

Para que a seleção ocorra, um número $\nu \in (0, 1)$ é aleatoriamente escolhido e os valores das fatias são sequencialmente acumulados tal que a heurística correspondente à fatia cujo valor acumulado extrapolar o valor de ν é selecionada. Considere uma roleta com 4 heurísticas com os pesos 60, 125, 115 e 200 para as heurísticas 1, 2, 3 e 4, respectivamente. A heurística 4 possui 40% de probabilidade de ser selecionada, logo, possui a maior fatia na roleta, representada no intervalo entre 0,6 e 1. Por exemplo, para $\nu = 0,73$ deve-se acumular o valor das quatro fatias da roleta para que o valor de ν seja extrapolado. Considerando os valores acumulados das fatias como 0,12 (heurística 1), 0,37 (heurística 2), 0,60 (heurística 3) e 1 (heurística 4), esta última será selecionada.

Uma solução nova π' é gerada dada uma solução corrente π ao aplicar uma heurística de remoção e uma de inserção. A solução π' é aceita como a nova solução corrente se possuir melhor valor da função de avaliação do que π ou a depender de uma determinada

probabilidade. Um critério de aceitação baseado no método *simulated annealing* é usado para determinar a probabilidade de aceite. Este método restringe gradativamente a qualidade das soluções, de forma que, ao longo das iterações, as soluções para serem aceitas devem possuir valores cada vez melhores. Uma solução π' gerada a partir de outra solução π é aceita com probabilidade calculada de acordo com a Equação (3.4), em que T representa a temperatura que decresce a cada iteração e $f(\pi)$ é o valor da solução π .

$$e^{-(f(\pi')-f(\pi))/T} \quad (3.4)$$

O funcionamento do ALNS é representado pelo fluxograma da Figura 3, em que a geração da solução inicial π_0 pode ser realizada pela utilização de heurísticas específicas para o problema tratado, de forma gulosa ou mesmo aleatória. Após a geração da solução inicial, o ALNS inicia sua execução, primeiramente, selecionando uma heurística de remoção e outra de inserção de elementos através do método da roleta e em seguida executa as heurísticas para obter uma nova solução π' a partir da solução inicial. Posteriormente, a nova solução é analisada, comparando-a com a solução corrente, e as heurísticas selecionadas recebem uma pontuação referente a qualidade do valor da nova solução gerada. Caso o algoritmo tenha atingido uma quantidade determinada de iterações, chamada de segmento, o ALNS suaviza a pontuação das heurísticas, reduzindo a diferença da pontuação entre elas. Após a suavização, o algoritmo volta para o início do ALNS até que uma condição de parada seja satisfeita.

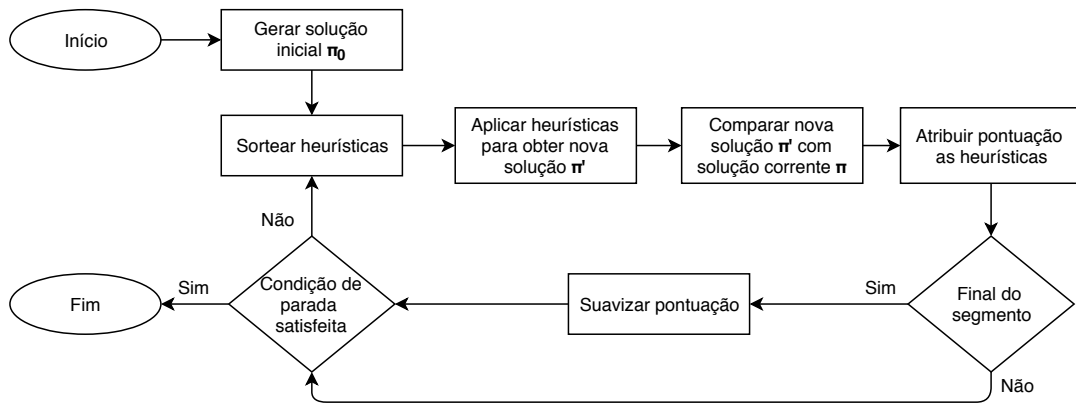


Figura 3 – Fluxograma do funcionamento do ALNS.

O ALNS é estruturado como apresentado no Algoritmo 1. O método parte de uma solução viável e a considera a melhor até então (linhas 1 e 2). A temperatura inicial para o critério de aceitação é definida antes de começar a execução do laço principal (linha 3). Em seguida, um laço de repetição (linhas 4 a 25) é executado enquanto um determinado critério de parada não for atendido. Ao entrar no laço, uma heurística de remoção e outra de inserção são selecionadas através do método da roleta (linha 5), e em seguida aplicadas à solução atual π para gerar uma nova solução π' (linha 6). Posteriormente, o resultado de

π' é comparado com a solução corrente π (linhas 7 a 14) e, caso seja melhor, se torna a nova solução corrente (linha 8). Uma segunda comparação é feita para checar se a solução corrente π é também melhor que a melhor solução π^* (linhas 10 a 13). Se for melhor, π^* é atualizada (linha 11). No entanto, se π' possuir valor igual ou pior que a solução corrente, deve-se avaliá-la de acordo com o método de aceitação do *simulated annealing* (linhas 15 a 18) e, caso seja aceita, se torna a nova solução corrente (linha 16). Após a análise da nova solução, as pontuações das heurísticas aplicadas são alteradas (linha 19) conforme o valor recebido na linha 9, 12 ou 17. Ao final de cada segmento (linha 20), as pontuações de todas as heurísticas são suavizadas (linha 21) e as probabilidades na roleta recalculadas (linha 22). A cada iteração, a última operação consiste em resfriar a temperatura do *simulated annealing* (linha 24). Por fim, quando o laço de repetição termina, a melhor solução π^* é retornada (linha 26).

Algoritmo 1: Busca Adaptativa em Grandes Vizinhanças.

```

1 Input: solução viável  $\pi$ ;
2  $\pi^* \leftarrow \pi$ ;
3 Inicialize a temperatura do simulated annealing;
4 while critério de parada não for atingido do
5   Escolha uma heurística de remoção e uma heurística de inserção usando o
     método da roleta;
6   Gere uma solução nova  $\pi'$  a partir de  $\pi$  usando as heurísticas selecionadas;
7   if  $f(\pi') < f(\pi)$  then
8      $\pi \leftarrow \pi'$ ;
9      $\sigma \leftarrow \sigma_2$ ;
10    if  $f(\pi') < f(\pi^*)$  then
11       $\pi^* \leftarrow \pi$ ;
12       $\sigma \leftarrow \sigma_1$ ;
13    end
14  end
15  else if  $\pi'$  atender o critério de aceitação do simulated annealing then
16     $\pi \leftarrow \pi'$ ;
17     $\sigma \leftarrow \sigma_3$ ;
18  end
19  Atualize a pontuação das heurísticas conforme  $\sigma$ ;
20  if contador de interação for múltiplo de  $\theta$  then
21    Suavize a pontuação observada das heurísticas e atualize os pesos;
22    Atualize a probabilidade das heurísticas na roleta conforme novos pesos;
23  end
24  Resfrie a temperatura do simulated annealing;
25 end
26 return  $\pi^*$ ;

```

O ALNS foi escolhido para este trabalho devido a suas diversas vantagens. Para a maioria dos problemas de otimização, incluindo o CMP, heurísticas de bom desempenho

são conhecidas e podem ser incluídas no conjunto de heurísticas do ALNS. Devido ao tamanho e à variedade das vizinhanças, o ALNS é capaz de explorar grandes porções do espaço de busca de forma estruturada, resultando em um algoritmo robusto que se adapta a várias características das instâncias e dificilmente fica preso em ótimos locais. Sendo assim, o ALNS é indicado para problemas nos quais vizinhanças pequenas não são suficientes para escapar de ótimos locais. Ademais, o ALNS vem sendo usado em uma variedade de problemas e tem apresentado bons resultados, como por exemplo em [Aksen et al. \(2014\)](#), [Li, Chen e Prins \(2016\)](#), [Mauri et al. \(2016\)](#), [Gullhav et al. \(2017\)](#), [Bach, Hasle e Schulz \(2018\)](#), [He et al. \(2018\)](#), [Roozbeh, Ozlen e Hearne \(2018\)](#) e [Liu, Tao e Xie \(2019\)](#).

4 Desenvolvimento

Neste capítulo são descritas com detalhes as particularidades do ALNS implementadas nesse trabalho. As seções abrangem o método de solução inicial, o operador de ruído, critérios de aceitação e de parada, e a implementação das heurísticas de remoção e inserção de vértices. Por fim, um método de pós-processamento é também apresentado.

4.1 Solução inicial

O ALNS depende de uma solução inicial viável para iniciar sua execução. [Pantrigo et al. \(2012\)](#) propuseram quatro diferentes métodos heurísticos de geração de solução inicial para o CMP. Dentre estes, o método denominado C1 é o que tem o melhor desempenho considerando qualidade de solução e é utilizado também em [Pardo et al. \(2013\)](#). Neste trabalho, o método de solução inicial é baseado no método C1 com algumas mudanças cruciais.

Considere v um vértice candidato para ser inserido na solução parcial π de tamanho k , considere também $labeled(v)$ como o conjunto dos vértices adjacentes a v presentes em π e $unlabeled(v)$ o conjunto dos vértices adjacentes a v ainda não inseridos na solução π . [Pantrigo et al. \(2012\)](#) definem a avaliação da largura de corte do vértice v , considerando sua inserção na posição $k + 1$, como $CW'_\pi(v) = CW'_\pi(w) - |labeled(v)| + |unlabeled(v)|$, em que $\pi(w) = k$.

O método original C1 é baseado na metodologia GRASP e a solução π é construída gradativamente. O primeiro vértice é selecionado aleatoriamente dentre os vértices de menor grau e adicionado a π . Em seguida, uma lista de candidatos é formada por todos os vértices adjacentes aos vértices presentes na solução. O valor da largura de corte $CW'_\pi(v)$ é calculado para cada vértice na lista de candidatos e uma lista restrita é criada com os melhores candidatos observando-se um limite para o valor da largura de corte dos vértices, definido aleatoriamente. Por fim, um vértice é selecionado aleatoriamente da lista restrita e inserido na solução. O método é executado até que uma solução completa seja obtida.

O novo método desenvolvido neste trabalho consiste em uma metodologia composta por escolhas gulosas. O primeiro vértice é escolhido dentre os vértices de menor grau. Em seguida, vértices com o menor valor de $CW'_\pi(v)$ são consecutivamente selecionados para compor a solução. Em caso de empates, o vértice adjacente a um vértice w presente na solução com o maior valor $\pi(w)$ deve ser selecionado. Caso o empate continue, a escolha deve ser feita de forma aleatória. O método termina quando a solução π estiver completa. A idéia consiste em selecionar a cada iteração o vértice que aumente o mínimo possível

o valor da largura de corte e que evite estender desnecessariamente arestas ao longo da disposição linear.

Considerando o grafo apresentado na Figura 1(a), uma possível solução inicial construída pelo método proposto é representada por $\pi = [a, c, d, e, b]$. Primeiramente, um vértice é selecionado aleatoriamente entre os vértices de menor grau: a , b e c . Se o vértice a for selecionado para compor a solução π , o próximo passo é escolher entre os vértices c ($CW'_\pi(c) = 2$) e e ($CW'_\pi(e) = 3$), ambos adjacentes ao vértice a . O vértice c deve obrigatoriamente ser escolhido, dado que possui o menor valor de largura de corte. Os próximos candidatos são os vértices d e e . Visto que ambos possuem largura de corte igual a três, a seleção é feita com base no maior valor de $\pi(w)$. Como demonstrado na Figura 4, o vértice d é adjacente ao vértice c ($\pi(c) = 2$) e o vértice e é adjacente ao a ($\pi(a) = 1$), desta forma d deve ser o vértice selecionado. Os dois vértices remanescentes fora da solução são b e e com $CW'_\pi(b) = 3$ e $CW'_\pi(e) = 2$. Finalmente, o vértice e é selecionado seguido do vértice b .

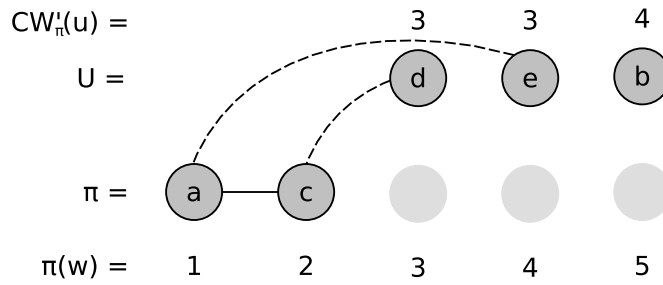


Figura 4 – $CW'_\pi(u)$ é o custo de inserir cada vértice que está fora da solução ($u \in U$), π é a solução parcial construída pelo método de solução inicial e $\pi(w)$ é a posição de cada vértice presente na solução. As arestas pontilhadas mostram as possíveis conexões entre os vértices inseridos na solução e os vértices que estão fora da solução e possuem valor mínimo de CW'_π

O novo método de solução inicial é apresentado no Algoritmo 2. O método inicia recebendo um grafo (linha 1), e os conjuntos de vértices presentes e vértices ausentes da solução são inicializados (linhas 3–4), assim como a disposição linear (linha 2). O primeiro vértice é selecionado aleatoriamente (linha 5), dentre os vértices com menor número de adjacências, e inserido na primeira posição na disposição linear (linha 6). Após a inserção, os conjuntos de vértices são atualizados (linhas 7 e 8). Um laço de repetição (linhas 9 a 18) é então executado até que todos os vértices tenham sido adicionados à solução. A cada iteração do laço, o valor da largura de corte é calculado para cada vértice fora da solução (linha 11) e o valor mínimo é salvo (linha 12). A lista de candidatos (CL) é construída (linha 13) contendo todos os vértices que possuem valor de largura de corte mínimo e são adjacentes a um vértice w com valor de $\pi(w)$ máximo. Em caso de empate, um vértice é selecionado aleatoriamente da lista CL (linha 14) e inserido na solução na próxima posição

disponível (linhas 10 e 15). Finalmente, os conjuntos de vértices são atualizados (linhas 16-17) e a disposição linear é retornada ao final (linha 19).

Algoritmo 2: Heurística construtiva proposta.

```

1 Input: grafo  $G = (V, E)$ ;
2  $\pi = \emptyset$ ;
3  $S = \emptyset$ ;
4  $U = V$ ;
5  $w = \arg \min_{v \in V} \{deg(v)\}$ . Em caso de empate, a decisão é aleatória;
6 Atribuir a posição  $k = 1$  para  $w$  ( $\pi(w) = 1$ );
7  $S = \{w\}$ ;
8  $U = U \setminus \{w\}$ ;
9 while  $U \neq \emptyset$  do
10    $k = k + 1$ ;
11   Calcule  $CW'_\pi(u) \quad \forall u \in U$ ;
12   Calcule  $CW'_{\min} = \min_{u \in U} CW'_\pi(u)$ ;
13   Construa  $CL = \arg \max_{u \in U} \{\pi(w) | \{u, w\} \in E \wedge CW'_\pi(u) = CW'_{\min}\} \forall w \in S$ ;
14   Selecione aleatoriamente um vértice  $v$  em  $CL$ ;
15    $\pi(v) = k$ ;
16    $S = S \cup \{v\}$ ;
17    $U = U \setminus \{v\}$ ;
18 end
19 return  $\pi$ ;
```

É importante observar que devido a aleatoriedade do método C1, [Pantrigo et al. \(2012\)](#) utilizam o método para gerar 100 soluções iniciais antes de executar o *Scatter Search*, e [Pardo et al. \(2013\)](#) geram 1.000 soluções iniciais usando o mesmo método antes de executar o VFS. No entanto, neste trabalho, o método de solução inicial proposto é executado por 10 vezes apenas e a melhor solução gerada é usada pelo ALNS como solução inicial. Os experimentos computacionais apresentados no Capítulo 5 demonstram que este número de execuções é o suficiente para gerar soluções iniciais de qualidade.

4.2 Ruído

Para fazer com que a busca realizada pelo ALNS se torne mais diversificada e evitar um comportamento similar a um método de descida rápida, aplica-se um operador de *ruído* ([ROPKE; PISINGER, 2006](#), sect. 3.6) no critério de aceitação e também em duas das heurísticas de inserção. Dado o valor da largura de corte CW de uma solução, o valor do ruído é aleatoriamente calculado no intervalo $[-maxN, maxN]$ dando origem a um novo valor de largura de corte $CW' = \max\{0, CW + \text{ruído}\}$. O valor $maxN$ é determinado

a depender de onde o ruído é aplicado.

O ruído é aplicado no critério de aceitação de acordo com uma probabilidade adaptativa. Quanto melhor for a solução gerada com o ruído, maior será a probabilidade de usá-lo em futuras iterações. Assim como as heurísticas de inserção e remoção, a probabilidade de utilizar o ruído passa pelo processo de suavização no final de cada segmento. Dado uma disposição linear π de um grafo G , o limitante do valor do ruído $maxN$ é calculado como $maxN = noiseAcc \times CW_{\pi}(G)$, em que o $noiseAcc$ é um parâmetro fixo e definido *a priori* que controla a quantidade de ruído usado.

Duas heurísticas de inserção foram implementadas de forma que o ruído é sempre usado e calculado a cada avaliação de inserção. Considerando uma disposição linear π de um grafo G , um vértice para ser inserido é posicionado em diferentes posições de π , gerando uma disposição linear π' a cada tentativa de inserção. O limitante $maxN$ do valor do ruído é calculado após a geração de cada π' , definido por $maxN = noiseIn \times CW_{\pi'}(G)$, em que $noiseIn$ é um parâmetro fixo definido *a priori*. O vértice deve ser inserido na posição que gerou a menor largura de corte, considerando o ruído. Desta forma, diversidade é introduzida nas soluções geradas pelas heurísticas com ruído.

4.3 Critério de aceitação

Após as heurísticas do ALNS gerarem uma nova solução, é necessário determinar se esta é melhor do que a solução corrente ou melhor do que a melhor solução já obtida pelo método. Como mencionado, o CMP possui platôs em seu espaço de soluções, o que significa que várias soluções com estruturas diferentes possuem o mesmo valor de solução. Levando esta característica em consideração, são utilizadas duas funções de avaliação para comparar duas soluções π_1 e π_2 com ordens de rotulação diferentes. Desta forma, é possível diferenciar soluções que poderiam ser caracterizadas como idênticas dado o valor de solução. Como mencionado na seção anterior, um operador de ruído pode ser aplicado às duas funções de avaliação utilizadas pelo ALNS.

Inicialmente, a largura de corte máxima é utilizada, conforme a Equação (3.1), para comparar se uma solução π_1 é melhor que outra solução de referência π_2 . Dependendo de uma dada probabilidade, o ruído pode ser calculado e adicionado a $CW_{\pi_1}(G)$. Se $CW_{\pi_1}(G) < CW_{\pi_2}(G)$ então π_1 deve ser aceita. Caso $CW_{\pi_1}(G) = CW_{\pi_2}(G)$, a segunda função de avaliação é utilizada.

Denominada CW^2 , a segunda função de avaliação é baseada no método proposto por Pardo et al. (2013), sendo é definida pelo somatório dos valores de largura de corte de uma disposição linear π : $CW_{\pi}^2(G) = \sum_{v \in V} CW_{\pi}(v)$. A solução π_1 deve ser aceita se somente se $CW_{\pi_1}^2(G) < CW_{\pi_2}^2(G)$. Por exemplo, na Figura 5 temos $CW_{\pi}(G) = 5$ e $CW_{\pi}^2(G) = 17$. Se trocarmos as posições dos vértices d e a , gerando uma nova disposição π' , teríamos

ainda assim $CW_{\pi'}(G) = 5$. Porém ao usar a segunda função, $CW_{\pi'}^2(G) = 18$, é possível diferenciar as duas soluções.

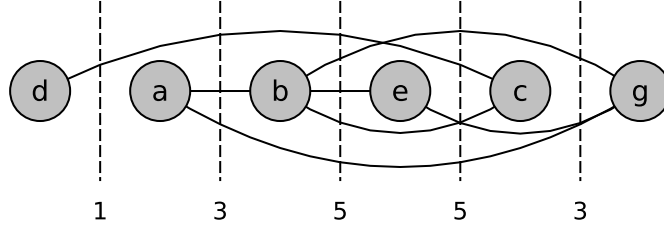


Figura 5 – Grafo G com seis vértices e sete arestas disposto na ordem de rotulação π com as correspondentes larguras de cortes.

4.4 Critérios de parada

O ALNS implementado neste trabalho possui duas condições de parada. O primeiro critério consiste em alcançar um número máximo de iterações, determinado *a priori*. O segundo critério consiste em temperatura T igual ou menor que 0,01, dado que uma temperatura muito baixa impede que soluções piores sejam aceitas, fazendo com que o algoritmo tenha mais dificuldade de sair de ótimos locais.

A temperatura inicial é definida por $T_{start} = -p_s \times CW_{\pi_0}(G) / \ln 0,5$ (ROPKE; PISINGER, 2006) em que π_0 é a solução inicial e $p_s \in [0, 1]$ representa o percentual de piora inicial. T_{start} aceita uma solução $p_s\%$ pior que a solução inicial π_0 com probabilidade de 0.5. Uma variante do cálculo da taxa de resfriamento, proposta por Santini, Ropke e Hvattum (2016), foi implementada neste trabalho. Este novo cálculo leva em consideração que a melhor solução (π^*) em uma certa iteração pode ser muito melhor do que a solução inicial e faz com que a temperatura resfrie mais rápido a cada nova melhor solução encontrada. Desta forma, a cada melhor solução encontrada uma temperatura adaptativa é atualizada como $T_{ad} = -p_e \times CW_{\pi^*}(G) / \ln 0,5$, em que p_e representa o percentual de piora final, também definido no intervalo $[0, 1]$. A taxa de resfriamento exponencial, $T = T \times (T_{ad}/T_{start})^{1/k}$, na qual k é o número máximo de iterações, é utilizada a cada iteração do ALNS para resfriar a temperatura.

4.5 Heurísticas de remoção

O ALNS implementado neste trabalho para o CMP é composto originalmente por 10 heurísticas de remoção. Na fase de remoção, os vértices são adicionados a uma lista de

remoção ao invés de serem imediatamente removidos da solução. A Figura 5 será utilizada como referência dos exemplos para algumas das heurísticas de remoção utilizadas.

4.5.1 Remoção de vértices críticos

Vértices críticos são aqueles que possuem o valor máximo da largura de corte. A remoção de vértices críticos (RVC) seleciona como candidatos para remoção todos os vértices críticos da solução. É plausível acreditar em uma melhora no valor da solução ao realocar todos os vértices de valor crítico em uma melhor posição. Por exemplo, na Figura 5, os vértices b e e são os que possuem o maior valor da largura de corte, portanto estes devem estar no conjunto de vértices a serem reinseridos.

4.5.2 Remoção aleatória

A remoção aleatória (RRand) seleciona q vértices de forma aleatória para remoção. O critério aleatório para remoção é utilizado com o intuito de perturbar a solução. O valor de q é calculado conforme a Equação (4.1), proposta por [Pereira et al. \(2015\)](#).

$$q = \lfloor n - \sqrt{(1 - u)(n - 1)^2 + 0.5} \rfloor \quad (4.1)$$

A Equação (4.1) segue uma distribuição triangular e seleciona aleatoriamente um número entre $[1, n]$, em que n é o número de vértices e u é uma variável aleatória entre $[0, 1]$. Este operador foi ajustado de forma que a seleção do valor de q permaneça entre $[0.15 \times n, 0.85 \times n]$, desta forma, podemos evitar que a heurística remova quantidades muito pequenas ou muito grandes de vértices. Por possuir uma distribuição triangular, o valor aleatório de q tende a ser mais próximo de $n/2$, de forma que essa operação pode ter grande impacto quando o número de vértices é grande.

4.5.3 Remoção de vértices relacionados

Nesta heurística são removidos vértices que são de alguma forma relacionados entre si ([SHAW, 1998](#)). Duas heurísticas de remoção são propostas com diferentes maneiras de calcular a relação entre os vértices. Primeiramente, a remoção de vértices com mesmo valor de largura de corte, RSh_{LC} , seleciona aleatoriamente um vértice e então marca para remoção todos os vértices com o mesmo valor de largura de corte. Por exemplo, na Figura 5, se o vértice c ($CW_{\pi}(c) = 3$) for selecionado para ser removido, o vértice a o será também por possuir o mesmo valor de largura de corte. A segunda remoção, RSh_{grau} , mede a relação através do grau do vértice. Um vértice é aleatoriamente selecionado, por exemplo vértice a , e todos os outros vértices com o mesmo grau também devem ser removidos (vértices e e c).

4.5.4 Remoção de vértices desbalanceados

Esta heurística de remoção é baseada na busca local introduzida por [Pardo et al. \(2013\)](#) e propõe remover todos os vértices que estão desbalanceados. O desbalanceamento de um vértice v depende se o $\text{grau}(v)$ tem valor par ou ímpar. O vértice com número par de adjacências está desbalanceado se $\text{grau}L_\pi(v) \neq \text{grau}R_\pi(v)$. Por outro lado, vértice com grau ímpar é considerado desbalanceado se $|\text{grau}L_\pi(v) - \text{grau}R_\pi(v)| > 1$.

Por exemplo, na Figura 5 os vértices a ($\text{grau}L_\pi(a) = 0$ e $\text{grau}R_\pi(a) = 2$), b ($\text{grau}L_\pi(b) = 1$ e $\text{grau}R_\pi(b) = 3$) e c ($\text{grau}L_\pi(c) = 2$ e $\text{grau}R_\pi(c) = 0$) possuem grau par e estão deslocados. Na mesma figura, o vértice g possui grau ímpar e está desbalanceado ($|\text{grau}L_\pi(g) - \text{grau}R_\pi(g)| = 3$).

Esta vizinhança foi decomposta em três vizinhanças diferentes: a primeira remove todos os vértices desbalanceados com número par de vértices adjacentes (RD_P); a segunda remove os vértices desbalanceados com número ímpar de vértices adjacentes (RD_I); e a terceira remove todos os vértices desbalanceados (RD) independente da paridade de adjacências.

4.5.5 Remoção de vértices adjacentes

A remoção de vértices adjacentes (RVA) seleciona um vértice de forma aleatória e em seguida o remove juntamente com todos os outros vértices adjacentes a ele. Por exemplo, se o vértice c da Figura 5 for selecionado, os vértices b e d devem também ser selecionados para remoção.

4.5.6 Remoção de vértices direcionada por arestas

A remoção direcionada por arestas é uma das contribuições deste trabalho. A ideia é encontrar um vértice crítico e, ao invés de removê-lo, remover os vértices incidentes às arestas que induzem o gargalo. Se múltiplos vértices com largura de corte máximo existem, um deve ser selecionado aleatoriamente. A heurística proposta é dividida em RdA_L , que remove os vértices na extremidade à esquerda das arestas que compõe o gargalo e RdA_R que de forma similar remove os vértices na extremidade à direita das arestas do gargalo. Na Figura 1(c) os vértices e e b representam o gargalo da disposição linear. Por exemplo, considerando o vértice b , note que o gargalo é composto por três arestas: $\{e, a\}$, $\{b, d\}$ e $\{e, d\}$. A heurística RdA_L deve remover os vértices e e b , e a heurística RdA_R deve remover os vértices d e a .

4.5.7 Análise de complexidade do pior caso das heurísticas de remoção

As heurísticas de remoção introduzidas nas Seções 4.5.1 e 4.5.3 (RSh_{LC}) precisam calcular o valor da largura de corte para cada um dos vértices. Isso quer dizer que para cada vértice todas as suas arestas incidentes devem ser analisadas. Essa operação possui complexidade limitada por $O(nm)$ e define a complexidade das vizinhanças. As heurísticas apresentadas nas Seções 4.5.4 e 4.5.6 são análogas à primeira heurística, em que se deve analisar toda a solução para determinar os vértices desbalanceados ou os vértices ligados à esquerda ou à direita do gargalo. Portanto também tem complexidade limitada por $O(nm)$.

A heurística RSh_{grau} (Seção 4.5.3) e a heurística presente na Seção 4.5.5 possuem complexidade limitada por $O(n)$, dado que é preciso percorrer somente os vértices verificando o grau de cada um. A Seção 4.5.2 descreve uma heurística que remove q vértices aleatoriamente, e para que essa remoção seja possível é necessário embaralhar a ordem dos vértices em uma operação de complexidade $O(n)$. Portanto, sua complexidade é limitada pelo custo desta operação de embaralhamento.

4.6 Heurísticas de inserção

Na fase de inserção, um vértice por vez é aleatoriamente selecionado da lista de remoção, removido da solução, e então reinserido na solução parcial. Após todos os vértices da lista de remoção terem sido reinseridos, uma nova solução é obtida. A escolha de remover um vértice por vez foi feita com o intuito de preservar as características da solução para tomar melhores decisões quando das inserções.

4.6.1 Inserção aleatória

A inserção aleatória (IRand) insere cada vértice em uma posição aleatória da solução parcial. A utilização desta inserção introduz diversidade à solução, que posteriormente pode ser aceita pelo ALNS mesmo que tenha um valor de solução pior.

4.6.2 Inserção na melhor posição balanceada

Nesta heurística, cada vértice v é inserido na melhor posição possível de uma solução parcial. Entretanto, diferentemente do método de melhor inserção tradicional, este somente considera aquelas posições que estejam entre os vértices adjacentes a v , em outras palavras, considera as posições que tornam o vértice v balanceado. Pardo et al. (2013) provaram que a inserção em outras posições não causa melhora no valor de solução.

Um vértice v depende do seu grau de adjacência para ser considerado balanceado, vértices com grau par são considerados balanceados quando inseridos em alguma posição

que $\text{grau}R_\pi(v) = \text{grau}L_\pi(v)$. Por exemplo, se o vértice c da Figura 5 for removido, o mesmo será inserido em alguma posição entre seus vértices adjacentes d e b . Logo, o vértice c deve ser inserido na melhor das duas posições disponíveis: entre d e a ou entre a e b . Por outro lado, vértices com grau ímpar devem ser inseridos antes ou depois do vértice adjacente médio, de forma que $|\text{grau}R_\pi(v) - \text{grau}L_\pi(v)| = 1$. Por exemplo, o vértice g é adjacente aos vértices a , b e e , desta forma, g deve ser inserido imediatamente antes ou depois do vértice b .

Esta heurística foi implementada de duas formas diferentes, com e sem a utilização do operador de ruído (IB_N e IB respectivamente). É esperado que a heurística com a aplicação do ruído tome decisões que se divergem das decisões míopes de melhor vizinho.

4.6.3 Inserção na melhor posição balanceada com melhora

A última heurística de inserção é similar à anterior (Seção 4.6.2). No entanto, depois de escolher a melhor posição e realizar a inserção de um determinado vértice o valor da função objetivo é considerada. A inserção deve ser desfeita em caso de piora no valor da função objetivo. A ideia é que, dados q vértices para serem reinseridos na solução, é razoável pensar que um determinado vértice a ser inserido não necessariamente precisa ser movido para uma posição balanceada para melhorar a solução. Por exemplo, na Figura 5 se o vértice a for inserido entre os vértices b e e , gerando uma disposição linear π' , o mesmo assume uma posição balanceada, porém, a quantidade de gargalos aumenta de dois para três. Considerando a segunda função de avaliação, temos $CW_\pi^2(G) = 17$ e $CW_{\pi'}^2(G) = 19$. Esta heurística foi também implementada com e sem o operador de ruído (IBm_N e IBm respectivamente).

4.6.4 Análise de complexidade do pior caso das heurísticas de inserção

A primeira heurística de inserção introduzida neste trabalho, descrita na Seção 4.6.1, insere elementos na solução em posições aleatórias e não avalia a função objetivo do problema. Considerando que a inserção é feita em uma estrutura de lista, esta heurística tem complexidade $O(nq)$ em que q é a quantidade de vértices a serem inseridos na solução.

As heurísticas apresentadas nas Seções 4.6.2 e 4.6.3 devem calcular a função objetivo a cada tentativa de inserção. A análise da função objetivo tem custo $O(nm)$, sendo que é necessário avaliar cada um dos vértices e suas arestas. Para a análise de complexidade desta heurística é necessário considerar dois casos: o de inserir vértices com número par de adjacências e vértices com número ímpar de adjacências. Para os vértices com número par de adjacências, o pior caso é dado quando suas adjacências se encontram na extremidade da disposição linear π . Isso quer dizer que o método deve fazer $n - 2$ tentativas de inserção para descobrir a melhor posição, logo a complexidade é limitado por $O(n^2m)$. Para vértices

com grau ímpar, somente duas inserções precisam ser avaliadas, antes e depois do vértice adjacente médio, portanto a complexidade é limitada por $O(nm)$.

4.7 Pós-processamento

Depois de concluir a execução do ALNS e obter uma solução, uma etapa de pós-processamento é aplicada na tentativa de refinar o valor da solução. Esta etapa consiste em uma busca local *2-swap*, que usualmente é aplicada a cada iteração do ALNS. Entretanto, neste trabalho optou-se por uma aplicação por instância, o que se justifica pelo desejo de não interferir no processo adaptativo do ALNS e também de não gerar um aumento no tempo de execução.

A busca local *2-swap* gera $n(n - 1)/2$ pares de diferentes vértices e os embaralha. Em seguida, movimentos de troca são efetuados nos pares, escolhidos aleatoriamente, de forma que movimentos de não melhora são descartados, enquanto movimentos de melhora são mantidos, o que provoca o reinício do método. A busca local é executada até que todos os pares tenham sido trocados sem obter movimentos de melhora.

5 Experimentos Computacionais

Um conjunto de experimentos preliminares foi realizado com o intuito de definir os parâmetros e heurísticas do ALNS assim como avaliá-lo como um todo. Primeiramente, os parâmetros necessários para o ALNS foram configurados, em seguida a performance de cada heurística foi avaliada para definir a configuração final do ALNS. Em um segundo experimento, mais abrangente, a versão final do ALNS foi avaliada considerando todos os conjuntos de instâncias disponíveis e os resultados gerados foram comparados com os métodos que compõem o estado da arte relacionado ao CMP.

O ambiente computacional consiste em um computador com processador *Intel Core i7-4790* de 3.6 GHz e 16 GB RAM, utilizando o sistema operacional Ubuntu 14.04 LTS. O método proposto foi codificado utilizando a linguagem C++ e compilado com gcc 4.8.4 e opções `-O3` e `-march=native`. Nenhuma opção de paralelismo foi utilizada.

5.1 Conjuntos de Instâncias

Os experimentos computacionais consideraram quatro conjunto de instâncias da literatura, *Small*, *Grid*, *Harwell-Boeing* e *Rome Graphs*. Os três primeiros conjuntos estão disponíveis em [Martí et al. \(2010\)](#) e o último conjunto pode ser encontrado em [Coudert \(2016a\)](#). Os resultados obtidos foram comparados com os resultados dos trabalhos considerados estado da arte: VFS ([PARDO et al., 2013](#)) e CILP ([COUDERT, 2016b](#)). Devido ao fator de aleatoriedade do método, foram realizadas 10 execuções independentes para cada uma das instâncias.

O conjunto *Small*, proposto por [Martí, Campos e Piñana \(2008\)](#), foi introduzido no contexto do problema *bandwidth*. O conjunto possui 84 grafos com dimensões que variam entre $16 \leq n \leq 24$ e $18 \leq m \leq 49$. Vários autores, entre eles [Martí et al. \(2013\)](#), [Coudert \(2016b\)](#) e [Pantrigo et al. \(2012\)](#), foram capazes de encontrar todas as soluções ótimas para este conjunto de instâncias.

O conjunto *Grid*, proposto por [Raspaud, Sýkora e Vrt'o \(1995\)](#), é composto por 81 grafos com dimensões que variam entre $9 \leq n \leq 729$ e $12 \leq m \leq 1409$. O valor ótimo da largura de corte é conhecido por construção ([RASPAUD et al., 2009](#)). No entanto, nenhum método revisado neste trabalho foi capaz de obter todas as soluções ótimas para este conjunto de instâncias.

Harwell-Boeing (HB) é um subconjunto derivado do *Harwell-Boeing Sparse Matrix Collection* disponível em domínio público no *Matrix Market Library* ([BOISVERT et al., 1997](#)). Esta coleção consiste em um conjunto de matrizes de teste oriundas de problemas

relacionados à solução de sistemas de equações lineares, quadrados mínimos e cálculos do autovalor de uma ampla variedade de disciplinas científicas e de engenharia. Do conjunto original, 87 instâncias foram selecionadas e variam de 30 a 700 vértices e 46 a 41686 arestas.

O último conjunto, *Rome Graphs*, é composto por 11.534 grafos com número de vértices definido no intervalo $10 \leq n \leq 100$ e número de arestas $9 \leq m \leq 158$. O mesmo foi utilizado pelos autores [Biedl et al. \(2013\)](#) e [Coudert \(2016b\)](#) para testar métodos exatos. Este conjunto é relativamente recente se comparado com os outros conjuntos e também pouco utilizado.

Uma análise foi realizada para cada uma das instâncias para identificar grafos com estruturas que podem facilitar a resolução do CMP. As estruturas que foram testadas são: grafos completos, grafos desconexos, grafos 1-árvore e grafos árvores. Os conjuntos de instâncias *Small* e *Grid* não possuem nenhuma dessas estruturas. O conjunto HB possui somente um grafo completo. Por fim, o conjunto de instâncias *Rome* possui 3 grafos desconexos, 287 1-árvores e 130 árvores.

O intuito de identificar instâncias com estruturas especiais é a possibilidade de tratá-las de forma diferente para solucioná-las de maneira ótima. A instância que possui o grafo completo é resolvida em tempo constante pois não depende da disposição linear escolhida, desta forma o ALNS não é mais necessário para essa instância. As demais instâncias com estrutura especial são resolvidas à otimalidade pelo ALNS, desta forma, nenhuma alteração adicional precisou ser feita.

5.2 Experimentos Preliminares

O ALNS necessita que sejam definidos alguns parâmetros para a sua execução. Para os experimentos apresentados neste capítulo, esses parâmetros foram determinados utilizando-se a ferramenta *irace* ([LÓPEZ-IBÁÑEZ et al., 2016](#)), um pacote codificado na linguagem R. O *irace* é um método *offline* de configuração automática de algoritmos de otimização. Dado um conjunto de instâncias do problema e um conjunto de possíveis valores para os parâmetros, o *irace* determina uma combinação apropriada de valores para os parâmetros. A Tabela 1 apresenta os intervalos de valores que o *irace* considerou ao definir cada um dos parâmetros.

Os parâmetros definidos nos experimentos preliminares e seus respectivos valores são:

- $\sigma_1 = 50$;
- $\sigma_2 = 15$;

Tabela 1 – Valores considerados pelo *irace* para cada parâmetro.

Parâmetro	Valores
σ_1	{10, 15, 25, 30, 35, 40, 45, 50}
σ_2	{10, 15, 25, 30, 35, 40, 45, 50}
σ_3	{10, 15, 25, 30, 35, 40, 45, 50}
Fator de reação ρ	[0,00 .. 1,00]
Percentual de piora inicial p_s	[0,50 .. 1,00]
Percentual de piora final p_e	[0,00 .. 0,50]
Número de iterações	{300, 600, 1000, 1300, 1600, 2000, 2300, 2600, 3000}
Tamanho de segmento θ	{50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300}
Porcentagem de noise $noiseAcc$	{0,02, 0,05, 0,07, 0,1}
Porcentagem de noise $noiseInser$	{0,02, 0,05, 0,07, 0,1}

- $\sigma_3 = 25$;
- Fator de reação $\rho = 0,85$;
- Percentual de piora inicial $p_s = 0,85$;
- Percentual de piora final $p_e = 0,45$;
- Número de iterações = 3000;
- Tamanho de cada segmento $\theta = 200$;
- Porcentagem de noise $noiseAcc = 0.07$;
- Porcentagem de noise $noiseInser = 0.07$.

Após definir os valores para cada um dos parâmetros necessários, o *irace* foi novamente aplicado para definir qual a melhor combinação de heurísticas no ALNS. O teste foi executado considerando todas as 10 heurísticas de remoção e as 5 de inserção. Cinco configurações diferentes foram selecionadas pelo *irace* como não dominantes entre si. Desta maneira, optou-se por selecionar a que obteve melhor qualidade de soluções em valores totais, composta por cinco heurísticas de remoção (RRand, RD, RD_P , RdA_L e RdA_R), e quatro heurísticas de inserção (IRand, IB_N , IB e IBm).

5.3 Comparação com os Resultados da Literatura

Nesta seção os melhores resultados encontrados pelo ALNS são comparados com os métodos do estado da arte. Os resultados para o conjunto de instâncias *Small* foram comparados com os valores ótimos e os conjuntos *Grid* e HB são comparados com o VFS (PARDO et al., 2013).

Um resumo dos resultados médios considerando 10 execuções independentes do ALNS são apresentados na Tabela 2. Esta tabela apresenta a melhor solução da literatura (OPT/BKS), tendo em vista que o conjunto HB é o único que não possui resultados ótimos, mas limitantes superiores. São também apresentados a melhor solução encontrada pelo ALNS (S^*), a média das soluções encontradas considerando as 10 execuções (S), o valor médio da solução inicial (S_0), a distância percentual entre S_0 e S^* (gap_{S_0,S^*}), a distância percentual entre S^* e OPT/BKS ($gap_{S^*,OPT}$), o desvio padrão das 10 execuções (σ) e, por fim, o tempo médio de execução em segundos (T). O $gap_{b,a}$ é calculado conforme a expressão $100 \times \frac{b-a}{a}$.

Tabela 2 – Resultados ALNS.

	OPT/BKS	S^*	S	S_0	gap_{S_0,S^*}	$gap_{S^*,OPT}$	σ	$T(s)$
Small	4,92	4,92	4,93	5,15	6,52	0,00	0,02	0,08
Grid	11,56	11,56	11,56	11,56	0,00	0,00	0,00	8,60
HB	311,55*	311,80	315,01	336,70	22,12	0,39	2,19	222,23

*Algumas soluções não são comprovadamente ótimas.

O ALNS proposto conseguiu alcançar os valores ótimos de todas as instâncias do conjunto *Small*, o que era esperado, tendo em vista que este conjunto é pequeno e trivial. A distância percentual entre a solução inicial e a solução ótima é de 6,52%, o que demonstra a importância do uso do ALNS mesmo para este conjunto com instâncias pequenas. As soluções geradas para o conjunto *Small* apresentaram um baixo valor de desvio padrão, o que significa que o método alcançou valores próximos durante as 10 execuções. Além disso, o tempo médio de execução é menor que um segundo.

O método proposto foi capaz de encontrar, pela primeira vez na literatura, os valores ótimos de todas as instâncias do conjunto *Grid*. No entanto, o ALNS não contribuiu para este resultado, tendo em vista que as soluções com valores ótimos foram encontradas pelo novo método de solução inicial proposto neste trabalho com tempo médio de execução igual a 0,03 segundos. Considerando a execução do ALNS, este conjunto obteve tempo médio de execução igual a 8,60 segundos, sendo o valor máximo de 44 segundos. Este tempo é considerado satisfatório, dado que o conjunto contém instâncias de até 729 vértices.

Para o conjunto HB, o de maiores dimensões, o ALNS obteve um gap médio de 0,39%, o menor até então reportado na literatura. Neste conjunto, o ALNS mostrou um desempenho satisfatório com um gap total entre a solução inicial e a melhor solução de 22,12%. Além disso, novos limitantes superiores foram encontrados para as instâncias *can*____445, *dwt*____361, *dwt*____503, *saylr3* e *str*____0 com valores 116, 38, 127, 44, e 381. Como já era esperado, o método exigiu maior esforço computacional neste conjunto, com tempo médio de execução de 222 segundos. O tempo de execução é fortemente influenciado pelas instâncias *shl*____0, *shl*____200, *shl*____400 e *west0655*, que somadas representam

63% do tempo total de execução. Embora baixo, o desvio padrão foi o maior dentre os conjuntos, pouco mais de 2%.

O método de pós-processamento foi capaz de diminuir o *gap* do conjunto HB em 0,28 pontos percentuais, para os conjuntos *Small* e *Grid* nenhuma melhoria pôde ser observada. A respeito do tempo de processamento, o método de refinamento proposto aumentou o tempo médio em 5 segundos para o conjunto HB e 3 segundos para o conjunto *Grid*.

Na Tabela 3, os melhores resultados do ALNS são comparados com os resultados encontrados pelo VFS. A coluna S^* apresenta a média dos valores das melhores soluções para cada instância de cada conjunto, a coluna $\#OPT$ apresenta a quantidade de soluções ótimas encontradas por cada método e a coluna *gap* apresenta a distância percentual média entre a solução de cada método e a melhor solução encontrada na literatura.

Tabela 3 – Comparação com o estado da arte.

	VFS			ALNS		
	S^*	$\#OPT$	gap	S^*	$\#OPT$	gap
Grid	12,23	59	3,25	11,56	81	0,00
HB	314,39	61	1,77	311,80	77	0,39

Nos dois conjuntos de instâncias, o ALNS superou os resultados do VFS. Primeiramente, no conjunto de instâncias *Grid*, o ALNS foi capaz de encontrar os valores ótimos de todas as instâncias, enquanto o VFS apresenta um *gap* de 3,25% e não conseguiu igualar 22 valores ótimos. Em seguida, para o conjunto de instâncias HB, considerando somente os resultados do ALNS e do VFS, a distância percentual é de 1,21%, com vantagem para o ALNS. Adicionalmente, o ALNS alcançou 16 soluções a mais com o melhor valor conhecido do que o VFS.

Ademais, testes estatísticos adicionais foram aplicados aos resultados do ALNS e do VFS para certificar a predominância do ALNS. O *Shapiro-Wilk Test* (SHAPIRO; WILK, 1965) foi aplicado separadamente aos resultados dos conjuntos *Grid* e HB e rejeitou a hipótese nula que os resultados do ALNS ($p\text{-value} = 4.989\text{e-}05$ e $p\text{-value} < 2.2\text{e-}16$) e do VFS ($p\text{-value} = 4.873\text{e-}06$ e $p\text{-value} < 2.2\text{e-}16$) podem ser modelados de acordo com uma distribuição normal. Em seguida, o *Wilcoxon Signed Rank Test* (REY; NEUHÄUSER, 2011), um método não-paramétrico para comparação de duas amostras pareadas, foi aplicado para verificar se existe diferença significativa entre os resultados. Para o conjunto *Grid*, o teste ($V = 0$, $p\text{-value} = 1.923\text{e-}05$) indicou que há diferença significativa entre os resultados. O valor de V indica que todos os valores de solução gerados pelo ALNS são iguais ou melhores do que os gerados pelo VFS. No conjunto HB, o $p\text{-value}$ é também menor do que 0,05, o que indica que os resultados gerados pelo ALNS são significativamente diferentes daqueles gerados pelo VFS ($V = 28.5$, $p\text{-value} = 5.714\text{e-}05$). Três soluções obtidas

pelo VFS possuem valor melhor que os encontrados pelo ALNS e 24 soluções geradas pelo ALNS são melhores que as geradas pelo VFS. Estes experimentos afirmam que o ALNS é melhor que o VFS nos conjuntos de instâncias testados.

Os resultados do conjunto *Rome Graph* são comparados com os resultados gerados pelo método CILP. Coudert (2016b), em sua seção de experimentos, ordenou as instâncias do conjunto *Rome Graphs* por valores de $n + m$ e determinou um limite de tempo de cinco minutos para solução de cada instância. O autor definiu um gatilho de parada que interrompe a execução do algoritmo depois de 400 instâncias consecutivas terem atingido o limite máximo de tempo. Por consequência, das 11.534 instâncias, o método proposto foi capaz de provar a otimalidade de 5.683 instâncias e encontrou limites superiores para 2.417 instâncias. Outras 3.434 instâncias não tem nenhum valor de solução reportado. Sendo assim os resultados do ALNS são comparados usando as 8.100 instâncias para as quais o CILP reportou algum valor.

Antes de fazer a comparação com o estado da arte, a Tabela 4 apresenta pela primeira vez na literatura o resultado médio do conjunto *Rome Graphs* completo. O ALNS apresenta uma média das melhores soluções de 8,41, com baixo desvio padrão e tempo de execução. Neste conjunto é reportado o limite superior para 3.434 instâncias. A ALNS demonstra uma boa convergência neste conjunto, apresentando uma distância percentual de 8,03% entre a solução inicial e a melhor solução encontrada. O método de pós-processamento não apresentou nenhum impacto nas soluções encontradas para este conjunto.

Tabela 4 – Resultados ALNS para o conjunto *Rome Graphs*.

	S^*	S	S_0	gap_{S_0, S^*}	σ	$T(s)$
Rome	8,41	8,60	9,13	8,03	0,19	0,82

A Tabela 5 compara os melhores resultados encontrados pelo ALNS com os resultados reportados pela formulação inteira CILP. A primeira linha mostra a média dos valores de solução para as 8.100 instâncias do conjunto *Rome Graphs*. A linha seguinte ($\#OPT$) apresenta a quantidade de instâncias que o CILP foi capaz de provar a otimalidade e quantos desses valores o ALNS foi capaz de igualar. A terceira linha mostra quantos limitantes superiores, de 2.417, o ALNS e o CILP encontraram o mesmo valor. Por fim, a última linha mostra quantos limitantes superiores o CILP encontrou com o valor mais baixo do que o ALNS e vice versa.

O ALNS foi capaz de encontrar mais de 99% das soluções comprovadamente ótimas, e foi capaz de melhorar 1.217 dos melhores resultados conhecidos. Como resultado, o método ALNS tem uma média menor que o CILP com uma distância percentual de 1,90%. O maior tempo de execução do ALNS, considerando todas as instâncias do *Rome Graphs*, é de 5,22 segundos em uma arquitetura 1,71 vezes mais rápida do que a usada

Tabela 5 – Comparação com o estado da arte CILP.

	CILP	ALNS
Média	6,91	6,70
# OPT	5.683	5.660
#Limites superiores iguais	1.197	1.197
#Limites superiores melhores	3	1.217

por Coudert (2016b) segundo Passmark (2018). O teste de *Anderson-Darling* foi aplicado aos resultados do ALNS e do CILP, que rejeitou a hipótese nula de que os resultados poderiam ser modelados de acordo com uma distribuição normal (ambos com $p\text{-value} = 2,2e-16$). O teste de normalidade de *Shapiro-Wilk* não foi aplicado devido à sua limitação quanto ao tamanho da amostra. A fim de analisar melhor a diferença entre os resultados dos dois métodos, o teste *Wilcoxon signed-rank* (REY; NEUHÄUSER, 2011) foi aplicado mais uma vez. O $p\text{-value}$ adquirido ($2,2e - 16$) estabelece que os resultados do ALNS são significativamente diferentes dos resultados do CILP. Considerando a qualidade dos resultados é possível concluir que o ALNS tem um desempenho superior.

5.4 Análises Adicionais

No Capítulo 4, heurísticas de inserção e remoção foram apresentadas e é de grande interesse analisar quais foram mais eficientes, para que em trabalhos futuros possa ser realizado um refinamento nesta parte importante do ALNS. A análise de convergência da solução é outra análise importante a ser feita, por permitir avaliar o número de iterações exigidas pelo ALNS para gerar soluções de boa qualidade e também sua eficiência. Com base nessas necessidades, esta seção é dedicada a essas duas análises.

Ao decorrer das iterações as pontuações (σ_1 , σ_2 e σ_3) atribuídas às heurísticas foram acumuladas a fim de obter um somatório total por instância. A Tabela 6 apresenta a distância percentual entre os pontos de cada uma das heurísticas para cada grupo de instâncias. A heurística com a maior pontuação tem distância percentual igual a zero.

Tabela 6 – Distância percentual entre as pontuações das heurísticas. As maiores pontuações tem *gap* zero.

Conjuntos	Heurísticas de remoção					Heurísticas de Inserção			
	RRand	RD	RD _P	RdA _L	RdA _R	IRand	IB	IB _N	IBm
Small	0.63	34.13	13.38	41.58	46.94	85.16	2.00	0.36	5.81
Grid	0.04	68.54	19.08	46.90	48.52	282.32	10.93	0.70	16.82
HB	0.60	85.60	48.70	56.61	59.60	383.64	17.76	0.62	34.11
Rome	0.20	46.57	20.36	51.42	56.00	101.39	6.77	0.16	10.65
Média	0.37	58.71	25.38	49.13	52.77	213.13	9.36	0.46	16.85

Devemos enfatizar que cada heurística foi escolhida pela ferramenta *irace*, o que

indica que todas elas foram testadas e um subconjunto promissor foi escolhido, eliminando assim as heurísticas menos importantes. Consequentemente, mesmo que uma heurística tenha baixa pontuação a mesma tem uma contribuição importante para obtenção do resultado final.

Dentre as heurísticas de remoção, RRand obteve a melhor pontuação acumulada e está a 58% de distância da heurística com a pior pontuação (RD). Considerando as heurísticas de inserção, IB_N obteve a melhor pontuação não muito distante das outras inserções balanceadas. Por outro lado, inserir vértices de forma completamente aleatória em soluções parciais não necessariamente gera soluções de qualidade, o que pode explicar a baixa pontuação da inserção aleatória. Tanto as heurísticas de remoção quanto de inserção que alcançaram as melhores pontuações empregaram a aleatoriedade dos componentes, o que introduz diversidade na busca. Além disso, o ruído produziu a melhor pontuação entre as operações de inserção, novamente, um mecanismo de diversidade.

O teste paramétrico de Análise de Variância (ANOVA) foi aplicado para verificar se existe diferença significativa entre a pontuação das heurísticas de remoção e inserção. O teste resultou em $p\text{-value} = 6.62e - 05$ para as heurísticas de remoção e $p\text{-value} = 0,0035$ para as de inserção, o que significa que pelo menos uma das médias é diferente das demais. Para uma análise completa, o *Pairwise t-test* (CRAMÉR, 2016) foi aplicado para avaliar quais são as heurísticas com diferença significativa. As Tabelas 7 e 8 apresentam onde se encontram tais diferenças, em que os valores em negrito representam diferença significativa entre duas heurísticas, i.e., $p\text{-value} < 0,05$.

Tabela 7 – $P\text{-value}$ do *Pairwise t-test* no conjunto de heurísticas de remoção. Valores em negrito indicam que existe diferença significativa entre o método da coluna e da linha.

	RRand	RD	RD _P	RdA _L
RD	0,0002			
RD _P	0,1797	0,0293		
RdA _L	0,0012	1,0000	0,2420	
RdA _R	0,0006	1,0000	0,1139	1,0000

Tabela 8 – $P\text{-value}$ do *Pairwise t-test* no conjunto de heurísticas de inserção. Valores em negrito indicam que existe diferença significativa entre o método da coluna e da linha.

	IRand	IB	IB _N
IB	0,0113		
IB _N	0,0083	1,0000	
IBm	0,0147	1,0000	1,0000

A heurística de remoção RRand apresenta diferença significativa entre todas as outras, com exceção da RD_P. RD_P por sua vez, apresenta diferença significativa com a

heurística RD. Note que, as novas heurísticas propostas neste trabalho (RdA_L e RdA_R) são competitivas e não apresentam diferença significativa em relação às heurísticas RD e RD_P . Considerando as heurísticas de inserção, somente a heurística IRand apresenta diferença significativa entre todas as outras, demonstrando equilíbrio na fase de inserção.

A Figura 6 apresenta o gráfico *boxplot* com a análise de convergência do método. O eixo y representa o número das iterações nas quais o método obteve sua melhor solução para cada instância, e o eixo x representa cada um dos conjuntos de instâncias. Para cada conjunto, o valor aferido representa o valor da média das 10 execuções para cada instância. A linha vermelha corresponde à mediana, ou seja, metade dos valores estão abaixo desta linha, divididos em dois quartis, e a outra metade está acima desta linha, divididos em mais dois quartis. As cruces no gráfico correspondem aos *outliers*, ou valores atípicos, valores que apresentam um grande afastamento dos demais.

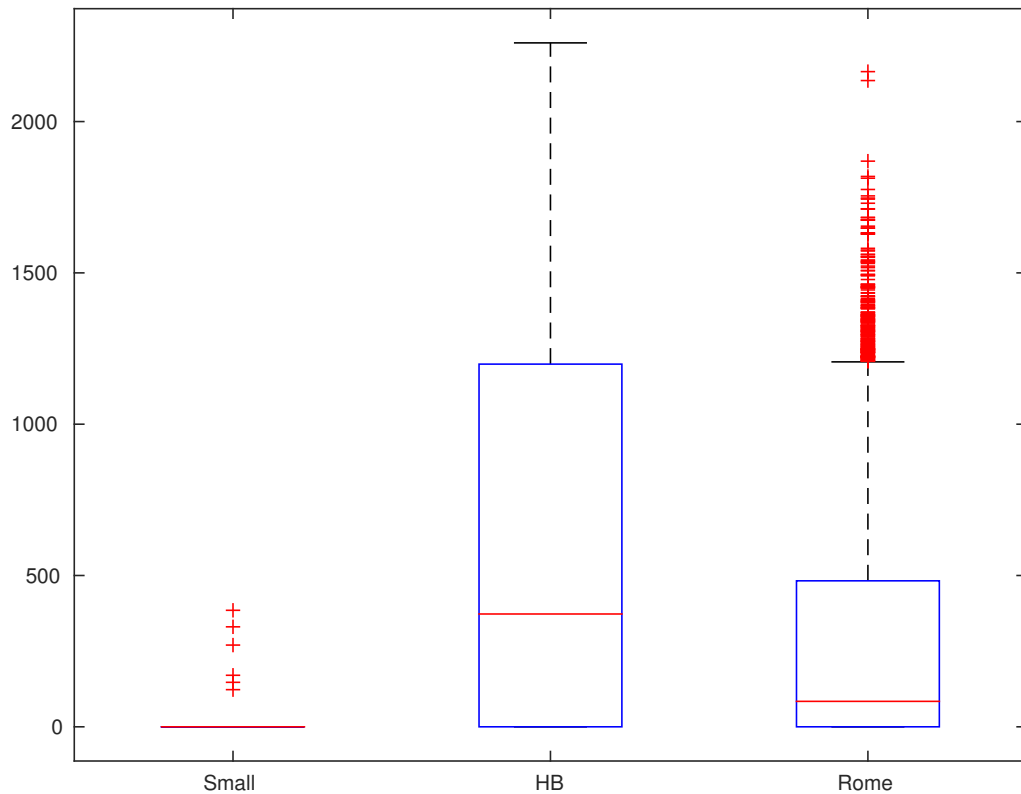


Figura 6 – Análise de convergência do ALNS para os conjuntos de instâncias Small, Grid e *Harwell-Boeing* (HB).

A convergência do ALNS não é apresentada para o conjunto de instâncias *Grid*, dado que para este conjunto todas as melhores soluções foram encontradas pelo método de solução inicial. Desta forma, a convergência para todo o conjunto é igual a zero. O conjunto *Small* possui baixos valores de convergência, o que era esperado por ser um conjunto de dimensões pequenas. Dentre as 84 instâncias, para 78 a solução ótima equivale a solução inicial e as outras 6 instâncias convergem até a iteração de número 385.

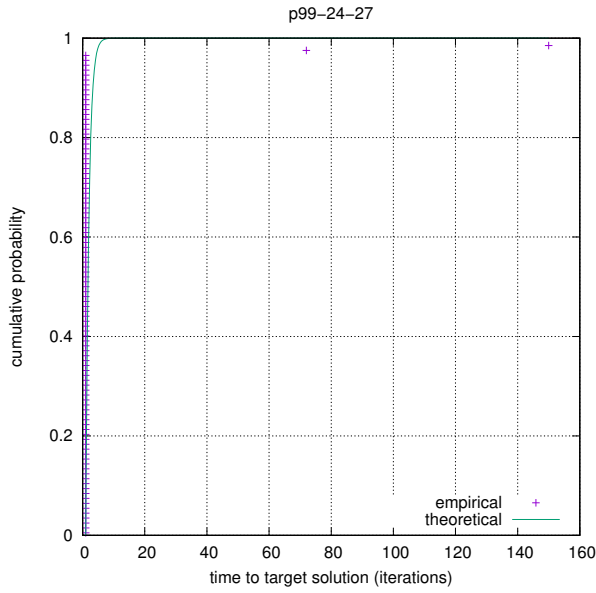
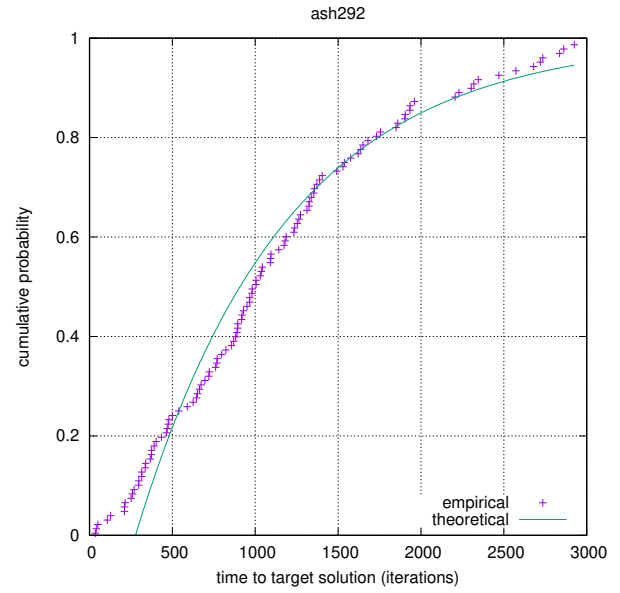
O conjunto *Harwell-Boeing* (HB), por ser o conjunto de maior dificuldade e com

as maiores instâncias, apresenta um espalhamento maior dos seus valores. Este conjunto não apresenta *outliers*, o que significa que possui a mesma quantidade de valores abaixo e acima da mediana. A média reportada é de 633, mediana equivalente a 372 e o máximo valor de convergência igual a 2.260. O último conjunto, *Rome Graphs*, apresenta média igual a 273 e mediana igual a 84. Este conjunto é composto por milhares de instâncias dentro de um intervalo de tamanhos considerável, por consequência, apresenta diversos *outliers* referentes as instâncias de tamanhos maiores. Para mais de 5.000 instâncias o melhor valor foi obtido já na solução inicial. O valor máximo, considerando os *outliers*, é de 2.162.

O limite de iterações no ALNS implementado neste trabalho foi definido com valor 3000. A análise de convergência demonstra que esse valor é suficiente, tendo em vista que os maiores valores estão abaixo de 2.300 iterações.

A fim de ilustrar a convergência do ALNS, uma instância de cada conjunto, com exceção do conjunto *Grid*, foi aleatoriamente selecionada para gerar gráficos *time-to-target* (ttt) (AIEX; RESENDE; RIBEIRO, 2007), vide Figura 7. A hipótese por trás dos gráficos ttt é que o tempo de execução de um método se ajusta a uma distribuição exponencial se o método for executado uma quantidade suficiente de vezes. Para uma determinada instância, o ALNS foi executado independentemente 100 vezes e reportou o número de iterações necessárias para atingir um valor de solução alvo de até no máximo 5% maior que o valor ótimo.

Em seguida, comparamos a distribuição resultante (*empirical*, dado pelas cruzes) com a distribuição exponencial (*theoretical*, representada pela linha). Os gráficos ttt apresentam a probabilidade cumulativa (eixo y) de atingir uma solução alvo em cada número de iterações (eixo x). Por exemplo, a Figura 7 (a) mostra que a probabilidade do ALNS encontrar uma solução pelo menos tão boa quanto o valor alvo em no máximo 1 iteração é de cerca de 85% a 90%. No entanto, a Figura 7 (b) e (c) mostram que é preciso 2500 iterações para encontrar a solução alvo com a mesma probabilidade anterior. Esta ilustração da convergência reforça a ideia do gráfico *boxplot* que estabelece as 3000 iterações que o ALNS é executado como suficientes.

(a) Instância *Small*.

(b) Instância HB.

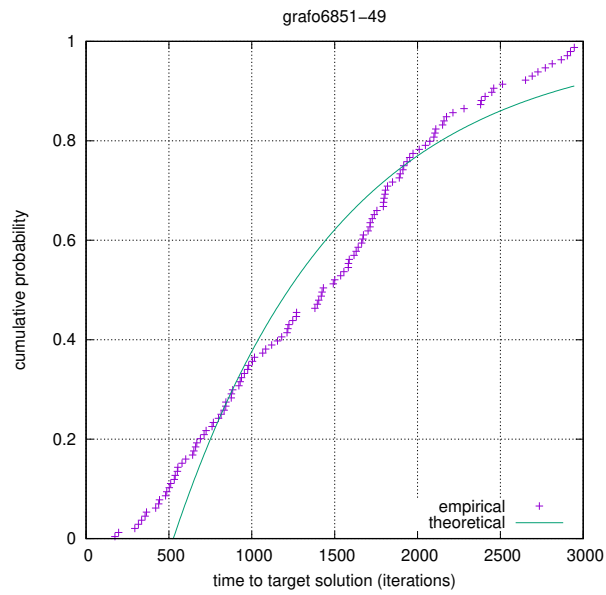
(c) Instância *Rome*.

Figura 7 – Time-to-target plots de instâncias aleatoriamente selecionadas.

6 Conclusão

Neste trabalho foi apresentada uma abordagem para o problema de Minimização de Largura de Corte (ou CMP, do inglês *Cutwidth Minimization Problem*), um problema \mathcal{NP} -Difícil com ampla área de aplicação. A abordagem reportada consiste em uma implementação da metaheurística Busca Adaptativa em Grandes Vizinhanças (ou ALNS, do inglês *Adaptive Large Neighborhood Search*), que incorpora novas heurísticas implementadas para o problema específico e também um novo método guloso para geração de soluções iniciais. É importante ressaltar que esta é a primeira aplicação reportada do ALNS para solução do CMP.

Os experimentos computacionais envolveram 11.786 instâncias de 4 diferentes conjuntos da literatura. Os resultados obtidos foram comparados com os melhores resultados encontrados na literatura e também com os métodos que compõem o estado da arte. O ALNS proposto juntamente com o novo método de solução inicial foram capazes de alcançar 97% de soluções comprovadamente ótimas, demonstrando competitividade em termos da qualidade das soluções geradas e do tempo de execução. Ademais, o método de solução inicial encontrou 22 novas soluções com valores ótimos para o conjunto *Grid*, e o ALNS melhorou o melhor valor conhecido para 5 instâncias do conjunto HB, e encontrou 3.434 novos limitantes superiores para o conjunto *Rome Graphs*, além de melhorar 1.217 melhores valores conhecidos para o mesmo conjunto. Testes estatísticos comprovaram a significativa vantagem do ALNS em termos de qualidade de solução quando comparado aos algoritmos que compõem o estado da arte. Os trabalhos futuros devem ser concentrados em melhorias finas do método proposto para aumentar a diversificação da busca, como a criação de buscas locais. Por fim, este trabalho gerou uma publicação no L Simpósio Brasileiro de Pesquisa Operacional e uma submissão para o periódico *European Journal of Operational Research*.

Referências

- ADOLPHSON, D.; HU, T. C. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, SIAM, v. 25, n. 3, p. 403–423, 1973. Citado na página 26.
- AIEX, R. M.; RESENDE, M. G.; RIBEIRO, C. C. Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, Springer, v. 1, n. 4, p. 355–366, 2007. Citado na página 64.
- AKSEN, D. et al. An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem. *European Journal of Operational Research*, Elsevier, v. 239, n. 2, p. 413–426, 2014. Citado na página 43.
- ANDRADE, D. V.; RESENDE, M. G. GRASP with evolutionary path-relinking. In: *Proc. of Seventh Metaheuristics International Conference (MIC 2007)*. [S.l.: s.n.], 2007. Citado 2 vezes nas páginas 31 e 32.
- ANDRADE, D. V.; RESENDE, M. G. GRASP with path-relinking for network migration scheduling. In: *Proceedings of the International Network Optimization Conference*. [S.l.: s.n.], 2007. Citado 2 vezes nas páginas 26 e 31.
- ARORA, S.; FRIEZE, A.; KAPLAN, H. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical programming*, Springer, v. 92, n. 1, p. 1–36, 2002. Citado na página 29.
- BACH, L.; HASLE, G.; SCHULZ, C. Adaptive large neighborhood search on the graphics processing unit. *European Journal of Operational Research*, Elsevier, 2018. Citado na página 43.
- BANSAL, R.; SRIVASTAVA, K.; SRIVASTAVA, S. A hybrid evolutionary algorithm for the cutwidth minimization problem. In: IEEE. *Evolutionary Computation (CEC), 2012 IEEE Congress on*. [S.l.], 2012. p. 1–8. Citado na página 31.
- BIEDL, T. et al. Using ILP/SAT to determine pathwidth, visibility representations, and other grid-based graph drawings. In: SPRINGER. *International Symposium on Graph Drawing*. [S.l.], 2013. p. 460–471. Citado na página 56.
- BODLAENDER, H. L.; KLOKS, T. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, Elsevier, v. 21, n. 2, p. 358–402, 1996. Citado na página 26.
- BOISVERT, R. F. et al. Matrix market: a web resource for test matrix collections. In: *Quality of Numerical Software*. [S.l.]: Springer, 1997. p. 125–137. Citado 2 vezes nas páginas 30 e 55.
- BOTAFOGO, R. A. Cluster analysis for hypertext systems. In: ACM. *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.], 1993. p. 116–125. Citado na página 26.

- CHUNG, M.-J. et al. Polynomial time algorithms for the min cut problem on degree restricted trees. *SIAM Journal on Computing*, SIAM, v. 14, n. 1, p. 158–177, 1985. Citado na página 29.
- COHOON, J. P.; SAHNI, S. Heuristics for backplane ordering. *Journal of VLSI and computer systems*, Computer Science Press, v. 2, n. 1-2, p. 37–60, 1987. Citado 2 vezes nas páginas 26 e 37.
- COUDERT, D. *Linear ordering problems on graphs*. [S.l.], 2016. <<http://www-sop.inria.fr/members/David.Coudert/code/graph-linear-ordering.shtml>> Acessado em 10/09/2018. Citado 2 vezes nas páginas 30 e 55.
- COUDERT, D. *A note on Integer Linear Programming formulations for linear ordering problems on graphs*. Tese (Doutorado) — Inria; I3S; Universite Nice Sophia Antipolis; CNRS, 2016. Citado 5 vezes nas páginas 30, 55, 56, 60 e 61.
- CRAMÉR, H. *Mathematical methods of statistics (PMS-9)*. [S.l.]: Princeton university press, 2016. v. 9. Citado na página 62.
- DIAZ, J. et al. Approximating layout problems on random geometric graphs. *Journal of Algorithms*, Elsevier, v. 39, n. 1, p. 78–116, 2001. Citado na página 26.
- DÍAZ, J.; PETIT, J.; SERNA, M. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, ACM, v. 34, n. 3, p. 313–356, 2002. Citado na página 26.
- DUARTE, A. et al. Variable neighborhood search for the vertex separation problem. *Computers & Operations Research*, Elsevier, v. 39, n. 12, p. 3247–3255, 2012. Citado na página 31.
- DUARTE, A. et al. Parallel variable neighbourhood search strategies for the cutwidth minimization problem. *IMA Journal of Management Mathematics*, Oxford University Press, v. 27, n. 1, p. 55–73, 2013. Citado 2 vezes nas páginas 31 e 32.
- EVEN, S.; SHILOACH, Y. NP-completeness of several arrangement problems. *Department of Computer Science, Israel Institute of Technology, Haifa, Isreal, Tech. Rep*, 1975. Citado na página 26.
- FELLOWS, M. R.; LANGSTON, M. A. On well-partial-order theory and its application to combinatorial problems of vlsi design. *SIAM Journal on Discrete Mathematics*, SIAM, v. 5, n. 1, p. 117–126, 1992. Citado na página 29.
- FRAIRE-HUACUJA, H. J. et al. Solving the cut width optimization problem with a genetic algorithm approach. In: *Nature-Inspired Design of Hybrid Intelligent Systems*. [S.l.]: Springer, 2017. p. 729–738. Citado na página 32.
- GAREY, M. R. et al. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, SIAM, v. 34, n. 3, p. 477–495, 1978. Citado na página 29.
- GAREY, M. R.; JOHNSON, D. S.; STOCKMEYER, L. Some simplified NP-complete graph problems. *Theoretical computer science*, Elsevier, v. 1, n. 3, p. 237–267, 1976. Citado na página 26.
- GAVRIL, F. Some NP-complete problems on graphs. In: *Proc. Conf. on Inform. Sci. and Systems, 1977*. [S.l.: s.n.], 1977. p. 91–95. Citado na página 26.

- GOLDBARG, M.; GOLDBARG, E. *Grafos: Conceitos, algoritmos e aplicações*. [S.l.]: Elsevier, 2012. Citado 3 vezes nas páginas 15, 37 e 38.
- GOMORY, R. E.; HU, T. C. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, SIAM, v. 9, n. 4, p. 551–570, 1961. Citado na página 37.
- GULLHAV, A. N. et al. Adaptive large neighborhood search heuristics for multi-tier service deployment problems in clouds. *European Journal of Operational Research*, Elsevier, v. 259, n. 3, p. 829–846, 2017. Citado na página 43.
- GURARI, E. M.; SUDBOROUGH, I. H. Improved dynamic programming algorithms for bandwidth minimization and the mincut linear arrangement problem. *Journal of Algorithms*, Elsevier, v. 5, n. 4, p. 531–546, 1984. Citado na página 29.
- HARPER, L. H. Optimal assignments of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, SIAM, v. 12, n. 1, p. 131–135, 1964. Citado na página 29.
- HARPER, L. H. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, Elsevier, v. 1, n. 3, p. 385–393, 1966. Citado na página 25.
- HE, L. et al. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research*, Elsevier, v. 100, p. 12–25, 2018. Citado na página 43.
- KORACH, E.; SOLEL, N. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, Elsevier, v. 43, n. 1, p. 97–101, 1993. Citado na página 37.
- KORNAI, A.; TUZA, Z. Narrowness, pathwidth, and their application in natural language processing. *Discrete Applied Mathematics*, Elsevier, v. 36, n. 1, p. 87–92, 1992. Citado na página 26.
- LEIGHTON, T.; RAO, S. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, ACM, v. 46, n. 6, p. 787–832, 1999. Citado na página 29.
- LENGAUER, T. Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees. *SIAM Journal on Algebraic Discrete Methods*, SIAM, v. 3, n. 1, p. 99–113, 1982. Citado na página 29.
- LEUNG, J. Y.; VORNBERGER, O.; WITTHOFF, J. D. On some variants of the bandwidth minimization problem. *SIAM Journal on Computing*, SIAM, v. 13, n. 3, p. 650–667, 1984. Citado na página 25.
- LI, Y.; CHEN, H.; PRINS, C. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, Elsevier, v. 252, n. 1, p. 27–38, 2016. Citado na página 43.
- LIN, Y.; YUAN, J. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica (English Series)*, Springer, v. 10, n. 1, p. 107–112, 1994. Citado na página 26.

LIPTON, R. J.; TARJAN, R. E. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, SIAM, v. 36, n. 2, p. 177–189, 1979. Citado na página 26.

LIU, R.; TAO, Y.; XIE, X. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, Elsevier, v. 101, p. 250–262, 2019. Citado na página 43.

LÓPEZ-IBÁÑEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016. Citado na página 56.

LÓPEZ-LOCÉS, M. C. et al. A new integer linear programming model for the cutwidth minimization problem of a connected undirected graph. In: *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*. [S.l.]: Springer, 2014. p. 509–517. Citado na página 30.

LOZANO, M. et al. Variable neighborhood search with ejection chains for the antibandwidth problem. *Journal of Heuristics*, Springer, v. 18, n. 6, p. 919–938, 2012. Citado na página 31.

MAKEDON, F.; SUDBOROUGH, I. H. On minimizing width in linear layouts. *Discrete Applied Mathematics*, Elsevier, v. 23, n. 3, p. 243–265, 1989. Citado 2 vezes nas páginas 26 e 29.

MAKEDON, F. S.; PAPADIMITRIOU, C. H.; SUDBOROUGH, I. H. Topological bandwidth. *SIAM Journal on Algebraic Discrete Methods*, SIAM, v. 6, n. 3, p. 418–444, 1985. Citado 2 vezes nas páginas 26 e 27.

MARTÍ, R.; CAMPOS, V.; PIÑANA, E. A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, Elsevier, v. 186, n. 2, p. 513–528, 2008. Citado 2 vezes nas páginas 30 e 55.

MARTÍ, R. et al. *Opticom project, University of Valencia*. [S.l.], 2010. <<http://www.opticom.es/cutwidth>> Acessado em 10/09/2018. Citado 2 vezes nas páginas 30 e 55.

MARTÍ, R. et al. Branch and bound for the cutwidth minimization problem. *Computers & Operations Research*, Elsevier, v. 40, n. 1, p. 137–149, 2013. Citado 2 vezes nas páginas 30 e 55.

MAURI, G. R. et al. An adaptive large neighborhood search for the discrete and continuous berth allocation problem. *Computers & Operations Research*, Elsevier, v. 70, p. 140–154, 2016. Citado na página 43.

MONIEN, B.; SUDBOROUGH, I. H. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, Elsevier, v. 58, n. 1-3, p. 209–229, 1988. Citado na página 26.

OHTSUKI, T. et al. One-dimensional logic gate assignment and interval graphs. *IEEE Transactions on Circuits and Systems*, IEEE, v. 26, n. 9, p. 675–684, 1979. Citado na página 26.

- PALUBECKIS, G.; RUBLIAUSKAS, D. A branch-and-bound algorithm for the minimum cut linear arrangement problem. *Journal of combinatorial optimization*, Springer, p. 1–24, 2012. Citado na página 30.
- PANTRIGO, J. J. et al. Scatter search for the cutwidth minimization problem. *Annals of Operations Research*, Springer, p. 1–20, 2012. Citado 6 vezes nas páginas 26, 31, 32, 45, 47 e 55.
- PARDO, E. G. et al. Variable formulation search for the cutwidth minimization problem. *Applied Soft Computing*, Elsevier, v. 13, n. 5, p. 2242–2252, 2013. Citado 10 vezes nas páginas 31, 32, 35, 45, 47, 48, 51, 52, 55 e 57.
- PASSMARK. *CPU benchmarks*. [S.l.], 2018. <http://www.cpubenchmark.net/cpu_list.php> Acessado em 10/09/2018. Citado na página 61.
- PEREIRA, M. A. et al. A hybrid method for the probabilistic maximal covering location–allocation problem. *Comp. & Oper. Res.*, Elsevier, v. 57, p. 51–59, 2015. Citado na página 50.
- RASPAUD, A. et al. Antibandwidth and cyclic antibandwidth of meshes and hypercubes. *Discrete Mathematics*, Elsevier, v. 309, n. 11, p. 3541–3552, 2009. Citado na página 55.
- RASPAUD, A.; ŠYKORA, O.; VRT’O, I. Cutwidth of the de bruijn graph. *RAIRO-Theoretical Informatics and Applications*, EDP Sciences, v. 29, n. 6, p. 509–514, 1995. Citado 3 vezes nas páginas 30, 37 e 55.
- REY, D.; NEUHÄUSER, M. Wilcoxon-signed-rank test. In: *International encyclopedia of statistical science*. [S.l.]: Springer, 2011. p. 1658–1659. Citado 2 vezes nas páginas 59 e 61.
- ROLIM, J.; ŠYKORA, O.; VRT’O, I. Optimal cutwidths and bisection widths of 2-and 3-dimensional meshes. In: SPRINGER. *Graph-Theoretic Concepts in Computer Science*. [S.l.], 1995. p. 252–264. Citado na página 29.
- ROOZBEH, I.; OZLEN, M.; HEARNE, J. W. An adaptive large neighbourhood search for asset protection during escaped wildfires. *Computers & Operations Research*, Elsevier, v. 97, p. 125–134, 2018. Citado na página 43.
- 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. Citado 4 vezes nas páginas 26, 39, 47 e 49.
- SÁNCHEZ-ORO, J.; PANTRIGO, J. J.; DUARTE, A. Combining intensification and diversification strategies in vns. an application to the vertex separation problem. *Computers & Operations Research*, Elsevier, v. 52, p. 209–219, 2014. Citado na página 31.
- SANTINI, A.; ROPKE, S.; HVATTUM, L. M. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, Submitted, 2016. Citado na página 49.
- SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, JSTOR, v. 52, n. 3/4, p. 591–611, 1965. Citado na página 59.

- 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. Citado 2 vezes nas páginas 39 e 50.
- SUDERMAN, M. Pathwidth and layered drawings of trees. *International Journal of Computational Geometry & Applications*, World Scientific, v. 14, n. 03, p. 203–225, 2004. Citado na página 26.
- SÝKORA, O.; VRT’O, I. Edge separators for graphs of bounded genus with applications. *Theoretical Computer Science*, Elsevier, v. 112, n. 2, p. 419–429, 1993. Citado na página 36.
- THILIKOS, D.; SERNA, M.; BODLAENDER, H. A polynomial time algorithm for the cutwidth of bounded degree graphs with small treewidth. *Algorithms—ESA 2001*, Springer, p. 380–390, 2001. Citado na página 29.
- THILIKOS, D. M.; SERNA, M. J.; BODLAENDER, H. L. Constructive linear time algorithms for small cutwidth and carving-width. In: SPRINGER. *International Symposium on Algorithms and Computation*. [S.l.], 2000. p. 192–203. Citado na página 29.
- VRT’O, I. Cutwidth of the r-dimensional mesh of d-ary trees. *RAIRO-Theoretical Informatics and Applications*, EDP Sciences, v. 34, n. 6, p. 515–519, 2000. Citado na página 37.
- YANNAKAKIS, M. A polynomial algorithm for the min-cut linear arrangement of trees. *Journal of the ACM (JACM)*, ACM, v. 32, n. 4, p. 950–988, 1985. Citado na página 29.