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

Vinícius Gandra Martins Santos
Busca Adaptativa em Grandes Vizinhanças Aplicada à Minimização da Largura de Corte em Grafos/ Vinícius Gandra Martins Santos. – Ouro Preto, 2018-
61 p. : il. (algumas color.) ; 30 cm.

Orientador: Marco Antonio Moreira de Carvalho

Dissertação (Mestrado) – Universidade Federal de Ouro Preto - UFOP
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação, 2018.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Carvalho, Marco Antonio Moreira de. II. Universidade Federal de Ouro Preto. III. Departamento de Computação. IV. Título

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, 16 de maio de 2018:

Marco Antonio Moreira de Carvalho
Orientador

Túlio Ângelo Machado Toffolo
Membro Interno

Ouro Preto
2018

Resumo

O problema de 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, desenho de diagramas de grafos e 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 252 instâncias 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 maior parte dos resultados da literatura.

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

Abstract

The Cutwidth Minimization Problem (CMP) consists in determining a linear layout for a graph in order to minimize the maximum linear cut of edges between each consecutive pair of vertices. This problem has applications in design of very large scale integration circuits, graph drawing and design of compilers, among others. The CMP is an NP-Hard problem and presents a challenge to exact methods and heuristics. In this work, it is reported for the first time the application of the metaheuristic Adaptive Large Neighborhood Search as a mean to the solution of the CMP. The computational experiments include 252 instances from the literature and the results found are compared with the current state-of-the-art. The proposed method is shown to be competitive, as it is able to match most of the literature best results.

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

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
CMP	Minimização de largura de corte (do ingês, *Cutwidth Minimization Problem*)
ALNS	Busca adaptativa em grandes vizinhanças (do ingês, *Adaptive Large Neighborhood Search*)
HB	Conjunto de instâncias *Harwell-Boeing*
B&B	*Branch-and-bound*
PLI	Modelo de programação linear inteira
GRASP	*Greedy Randomized Adaptive Search Procedure*
PR	*Path Relinking*
VNS	*Variable Neighborhood Search*
VFS	*Variable Formulation Search*
GAF	Algoritmo Genético *Fuzzy*
RVC	Remoção de vértices críticos
RRand	Remoção Aleatória
RShaw	Remoção de Vértices Relacionados
RDP	Remoção de Vértices Desbalanceados (grau par)
RDI	Remoção de Vértices Desbalanceados (grau ímpar)
RD	Remoção de Vértices Desbalanceados
IRand	Inserção Aleatória
$IBest_{CM}$	Inserção na melhor posição (desfaz movimento em caso de piora no valor de solução)
$IBest_{SM}$	Inserção na melhor posição
BKS	Melhor valor de solução da literatura
OPT	Valor ótimo da solução

SS	*Scatter Search*
gap	Distância percentual
ANOVA	Análise de variância
ttt-plot	*Time-to-Target plot*

Lista de símbolos

G	Grafo não dirigido
V	conjunto de vértices
E	conjunto de arestas
f	disposição linear ou rotulação dos vértices
$CW(G)$	valor da largura de corte do grafo G
π_0	Solução inicial
N^-	Conjunto de vizinhanças de remoção
N^+	Conjunto de vizinhanças de inserção
$\sigma 1$	pontuação atribuita para vizinhanças
$\sigma 2$	pontuação atribuita para vizinhanças
$\sigma 3$	pontuação atribuita para vizinhanças
ρ	Fator de reação $\in$ (0,1).
θ	Número de segmentos
$r_{i,j}$	pontos observados da vizinhança 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
$T(sec)$	Tempo de execução em segundos

Sumário

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 propriedade comum dos vértices é denotada pelo *grau*, ou seja, o número de arestas incidentes aos mesmos. Uma topologia específica é a denominada *grafo linha*, na qual todos os vértices possuem grau dois, exceto por dois vértices que possuem grau um, desta forma, assemelhando-se a uma linha. Outra propriedade importante 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.

Problemas 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 leiautes 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. Um 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 matemática, como demonstrado por Coudert (2016) que propôs um método de programação linear inteira, e também de forma heurística e metaheurística. Neste trabalho será dado destaque a abordagens heurísticas como forma de solução do CMP.

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).

Para a solução do CMP, este trabalho propõe implementar o método Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS), proposta por Ropke e Pisinger (2006). Esta é uma metaheurística recente que utiliza buscas locais e perturbações para explorar uma porção ampla das possíveis soluções para problemas combinatórios. Este algoritmo é robusto, se adapta às várias características das instâncias e dificilmente fica preso em ótimos locais.

1.1 Justificativa

Um grande número de problemas podem ser formulados 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.

Sendo assim este trabalho, que trata um problema NP-Difícil (MAKEDON; PAPADIMITRIOU; SUDBOROUGH, 1985), pode contribuir para gerar conhecimento e inovação para diversas áreas de pesquisa, por exemplo, engenharia elétrica e teoria 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 implementado considerando dados reais e também com problemas teste publicamente disponíveis, realizando uma análise crítica considerando outros métodos da literatura.

Além dos objetivos principais, outros produtos deste projeto de pesquisa serão trabalhos publicados em periódicos e eventos nacionais.

1.3 Organização do Texto

O restante deste trabalho está organizado como descrito a seguir. O Capítulo 2 apresenta uma revisão detalhadas 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, os Capítulos 6 e 7 apontam direções para os trabalhos futuros e apresentam a conclusão do trabalho.

2 Revisão Bibliográfica

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

Alguns dos trabalhos mencionados nessa revisão utilizam em seus experimentos computacionais três conjuntos de instâncias principais. O primeiro conjunto, *Small* (MARTÍ; CAMPOS; PIÑANA, 2008), é composto por 84 instâncias com número de vértices variando entre 16 a 24 e número de arestas entre 18 e 49. O segundo conjunto de instâncias, *Grid* (RASPAUD; SỲKORA; VRT'O, 1995), é composto por 81 matrizes que representam grades bidimensionais com dimensões entre 9×9 a 729×729. O ultimo conjunto, *Harwell-Boeing* (HB), é um subconjunto derivado do *Harwell-Boeing Sparse Matrix Collection* (LIBRARY, 2007). Este conjunto possui as instâncias de maiores dimensões: 87 instâncias que variam entre 30 a 700 vértices e 46 a 41686 arestas. Os três conjuntos estão disponíveis em Optsicom (2017). Outro conjunto de instâncias, *Rome Graphs*, é mais recente e foi introduzido pela comunidade de pesquisa de desenho de grafos. Este conjunto pode ser encontrado em Coudert (2017), foi usado por Biedl et al. (2013) e por Coudert (2016) e contém 11.534 grafos com cardinalidade $10 \leq n \leq 100$.

Algumas topologias específicas de grafos permitem formulações que resolvam o CMP em tempo determinístico polinomial. Por exemplo, 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, Lengauer (1982) propôs um algoritmo linear para o CMP em árvores completas com grau definido. 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, Sỳkora e Vrt'o (1995) para malhas de duas e três dimensões, malhas toroidais e cilíndricas de duas dimensões e malhas toroidais de 3 dimensões. Raspaud et al. (2009) soluciona o problema *Antibandwidth* para grafos grade e grafos toroidais, além de estimar o valor da solução para hipercubos de ordem máxima três.

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 Arora, Frieze e Kaplan (2002) e Leighton e Rao (1999). O primeiro trabalho propôs um algoritmo polinomial aproximado para grafos densos. Este algoritmo possui complexidade polinomial para o tempo de execução, $n^{O(logn/\epsilon^2)}$ para $\epsilon > 0$, e relação de aproximação de $\epsilon \times n^2$. O segundo trabalho, foi proposto para grafos gerais e consiste em um algoritmo de aproximação polinomial com relação de aproximação de $O(CWlog^2n)$, em que CW é o valor ótimo ou limite inferior da largura de corte.

Algoritmos exatos para grafos gerais foram propostos por diversos autores. Um algoritmo exato *branch-and-bound* (B&B) foi proposto por Martí, Campos e Piñana (2008) para solucionar o problema de *Bandwidth* em matrizes. Um conjunto composto por 20 grafos, entre eles grafos cíclicos e árvores binárias perfeitas, foram usados para testar o B&B proposto por Romero-Monsivais, Rodriguez-Tello e Ramírez (2012), o qual consegue obter valores ótimos para grafos com $n \leq 40$.

Uma adaptação do método B&B foi proposto por Palubeckis e Rubliauskas (2012). O método engloba um teste de dominância para reduzir o espaço de busca através da implantação 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 algumas instâncias retiradas dos três conjuntos de instâncias, *Small*, *Grid* e HB, para os experimentos computacionais. O método proposto conseguiu soluções ótimas para algumas das menores instâncias dos dois primeiros conjuntos. 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 anteriormente pelos mesmo autores e obtiveram 7 soluções ótimas, 38% a mais que o último modelo inteiro. Posteriormente, Coudert (2016) propôs um novo modelo de programação inteira (PLI). O modelo foi testado com dois conjunto de instâncias, *Small* e *Rome graphs*. O modelo alcançou soluções com valores ótimos para todas as instâncias

do primeiro conjunto. Para o segundo conjunto, o modelo conseguiu o valor ótimo para quase todos os grafos com até $n+m < 90$ e poucos para $n+m > 120$, totalizando 49% das instâncias com valores ótimos encontrados no segundo conjunto. 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, o método *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; atualização do subconjunto. 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.

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 vértices 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* e outros 61 valores ótimos para o conjunto HB, ambos em tempo médio próximo a 90 e 120 segundos, respectivamente.

Outra variação de VNS, proposta por, Duarte et al. (2013) consiste em uma implementação paralela. A paralelização foi 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. Note que 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 lugar. 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 outra

versão do mesmo algoritmo genético proposto pelos mesmos autores, porém, 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 um gap de mais de 85% se comparado com o resultado encontrado pelo VFS.

3 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica do problema de Largura de Corte e de métodos disponíveis na literatura que são utilizados 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 f de G atribui os inteiros $\{1, 2, ..., n\}$ para os vértices V, em que cada vértice recebe um inteiro diferente. A largura de corte de um vértice v considerando a rotulação f, $CW_f(v)$, é o número de arestas $(u, w) \in E$ que satisfaz $f(u) \leq f(v) < f(w)$. A largura de corte do grafo $CW_f(G)$ é o máximo da largura de corte dos seus vértices, vide Equação (3.1).

$$CW_f(G) = max_{v \in V} CW_f(v). \tag{3.1}$$

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

$$CW(G) = min_{f \in \Pi_n} CW_f(G). \tag{3.2}$$

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 com a ordem de rotulação f. Note que $f(E) = 1$, o que quer dizer que o vértice E se encontra na primeira posição da ordem f, seguido pelo vértice D ($f(D) = 2$) e assim por diante. Desta maneira, f pode ser representada por (E, D, B, A, C). A largura de corte de cada vértice é representado 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 ou no vértice A e com ponto final depois do vértice A ($\{A, C\}$, ($\{D, C\}$). Como a largura de corte $CW_f(G)$ é calculado pelo maior corte entre todos os vértices V, nesse exemplo em particular $CW_f(G) = CW_f(D) = 4$. A Figura 1(c) demonstra uma ordem de rotulação alternativa em que os vértices B e D foram trocados de lugar e a largura de corte diminui em uma unidade ($CW_f(G) = 3$).

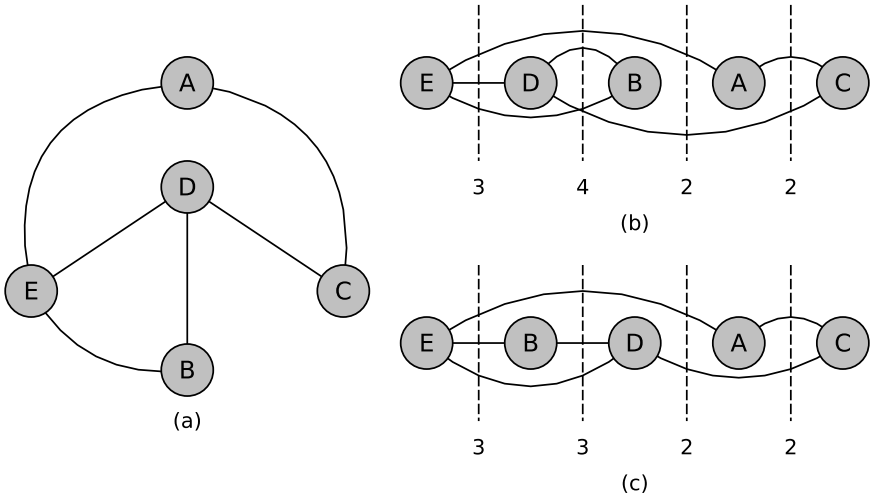


Figura 1 – (a) Grafo G com cinco vértices e seis arestas. (b) Ordem f dos vértices do grafo (a) com as correspondentes larguras de cortes. (c) Ordem f' com alteração na posição dos vértices B e D em relação a f.

Limitantes inferiores e superiores quando de qualidade podem delimitar o espaço de busca e são importantes para problemas em que não é possível desenvolver um algoritmo exato em tempo polinomial. Os resultados encontrados por metaheurísticas, que não possuem propriedade de corretude, podem ser comparados com esses limitantes para inferir a qualidade da solução gerada. Limitantes inferiores de qualidade representam um valor próximo ou igual ao valor da solução ótima para instâncias de problemas de minimização.

Para o problema de largura de corte, limitantes são apresentados para grafos de *Bruijn* (RASPAUD; SỲKORA; VRT'O, 1995), malhas de d dimensões em árvores r-árias (VRT'O, 2000) e grafos com grau máximo conhecido (SỲKORA; VRT'O, 1993). 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), representa 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. O maior fluxo encontrado representa a largura de corte mínima.

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

problema de decomposição de caminho. Uma decomposição de caminho é uma sequência de subconjuntos de vértices de G, em que os pontos finais de cada aresta estão contidos em um dos subconjuntos de forma que cada vértice apareça em uma subsequência contígua dos subconjuntos. O *Pathwidth* por sua vez é o tamanho do maior conjunto de tal decomposição menos um. O *Tree-width* é um problema análogo ao *Pathwidth*, no entanto a decomposição é feita em uma árvore. Uma decomposição em árvore representa os vértices de um dado grafo G como subárvores de uma árvore, de tal forma que os vértices no grafo dado são adjacentes apenas quando as subárvores correspondentes se cruzam.

Além de problemas relacionados, existem problemas equivalentes que possuem a mesma função objetivo, 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. Os *boards* quando dispostos linearmente formam o *backplane*. O objetivo é encontrar uma permutação ótima de *boards* tal que a densidade do *backplane* seja minimizada, análogo ao problema da largura de corte.

3.2 Busca Adaptativa em Grandes Vizinhanças

Em otimização combinatória, técnicas de *busca local* são frequentemente utilizadas para encontrar bons resultados de forma rápida e pouco custosa. Este tipo de busca consiste em explorar o *espaço de busca*, definido por todas as soluções possíveis (conjunto Π), através de *vizinhanças*. Estas por sua vez, equivalem a um conjunto de soluções com estruturas similares obtidas através de simples *movimentos* na solução original, como por exemplo trocar dois elementos de posição em um vetor para soluções que são representada com o mesmo. A busca local parte de uma solução inicial e é aplicada sucessivamente até que encontre um *ótimo local* ou sua condição de parada seja atendida. A busca atinge um ótimo local quando mudanças discretas não resultam em nenhum aprimoramento na solução, enquanto o *ótimo global* é o melhor valor possível para a função que define a qualidade das soluções, chamada de *função objetivo*. Diversas buscas locais podem ser utilizadas em simultâneo com o objetivo de escapar de ótimos locais e realizar uma busca robusta no espaço de busca de um problema. Essas buscas são frequentemente coordenadas por *metaheurísticas*, métodos de solução que utilizam estratégias de mais alto nível.

A revisão da literatura feita no Capítulo 2 mostra que o CMP é um problema de difícil solução, principalmente quando o tamanho das instâncias começam a crescer e demandar maior tempo computacional. Neste cenário, os métodos heurísticos são utilizados para gerar soluções de qualidade com custo computacional razoável. É natural pensar que precisamos de um método com heurísticas eficazes e formas para escapar de ótimos locais.

Ropke e Pisinger (2006) introduziram a metaheurística Busca Adaptativa em

Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS) 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 funcionamento do ALNS é representado através do fluxograma da Figura 2, em que a geração da solução inicial π_0 pode ser realizada através da utilização de heurísticas específicas para o problema tratado, de forma gulosa ou aleatória. Após a geração da solução inicial, o ALNS inicia sua execução, primeiramente, selecionando uma heurística de remoção e outra de inserção de vértices através do método da roleta e em seguida executa as heurísticas para obter uma nova solução π' a partir da solução inicial. Posteriormente, as heurísticas selecionadas recebem uma pontuação referente a qualidade do valor da nova solução gerada: quanto melhor o valor da solução melhor será a pontuação das heurísticas. Caso o algoritmo tenha atingido uma quantidade determinada de iterações, chamada de segmento, o ALNS suaviza a pontuação dada as heurísticas, reduzindo a diferença da pontuação entre elas. Após a suavização, o algoritmo volta ao primeiro passo até que uma condição de parada seja satisfeita. Todos os métodos necessários para a execução do ALNS, serão descritos nesta seção.

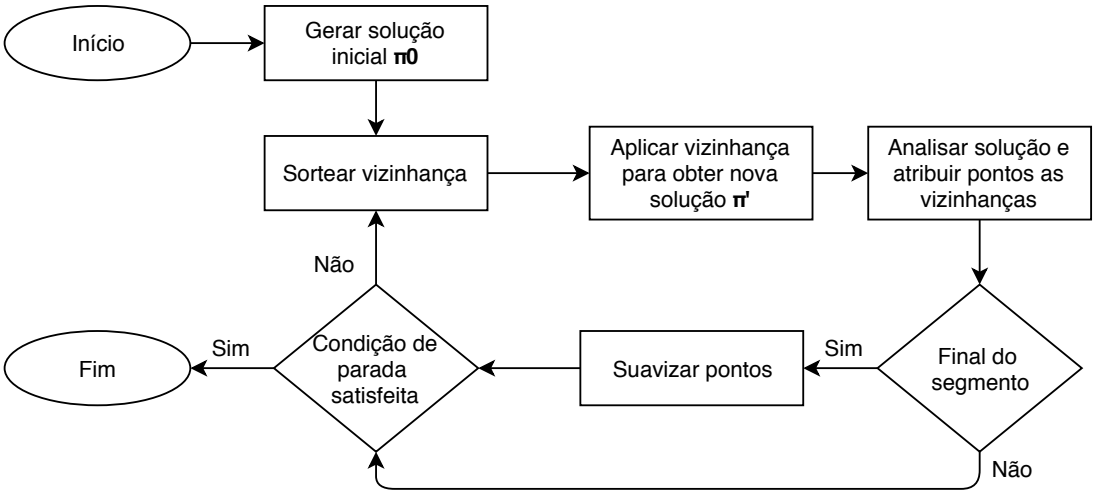


Figura 2 – Fluxograma do funcionamento do ALNS.

O ALNS utiliza um conjunto N^- de heurísticas de remoção, que removem q 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 ao longo das iterações e atribuir pontos para cada uma de acordo com o seu desempenho, tendo em vista que quanto mais uma determinada heurística contribuir para a melhoria no valor da solução, maior será a pontuação atribuída à mesma e consequentemente, maior a chance de ser selecionada em iterações futuras. A probabilidade da heurística N_i ser selecionada é dada pela divisão da pontuação da heurística N_i, denotados por r_i, pelo somatório da pontuação de todas as heurísticas.

A cada iteração o ALNS seleciona uma heurística de remoção e uma outra de inserção. A seleção é feita através do método da *roleta*. A roleta é representada por um intervalo $R = [0...1] \in \mathbb{R}$, em que cada heurística i recebe uma fatia da roleta proporcional à sua probabilidade $P(N_i)$ de ser escolhida. Sendo assim, quanto maior for a probabilidade da heurística ser escolhida, maior será o a sua fatia de intervalo na roleta e consequentemente maior a chance da heurística ser selecionada. Considere uma roleta com 4 heurística com os valores 0,12, 0,25, 0,23 e 0,40 paras as heurística 1, 2, 3 e 4 respectivamente. A heurística 4 tem 40% de probabilidade de ser escolhida, 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. 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.

Estes pontos utilizados para a criação da roleta são ajustados de forma adaptativa. A execução do ALNS é dividida em blocos de iterações de tamanho θ chamados *segmentos*, durante os quais os pontos das heurísticas são cumulativamente calculados. Por exemplo, se a metaheurística for executada por 500 iterações, podemos ter 5 segmentos em que $\theta = 100$. Sendo $\overline{r}_{i,j}$ os *pontos observados* da heurística i no segmento j, podemos classificá-los 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.

Ao final de cada segmento, os pontos acumulados para cada heurística passam por um processo de *suavização*, calculado conforme a Equação (3.3), em que a_i é o número de vezes que a heurística i foi chamada durante o segmento j. O fator de reação $\rho \in (0, 1)$ controla o quão rápido o ajuste de pontos do algoritmo reage às mudanças na pontuação de cada heurística. Quando $\rho = 1$ a roleta é baseada somente nos pontos do segmento imediatamente anterior. Por outro lado, quando $\rho < 1$, os pontos dos segmentos anteriores são todos levados em consideração. Caso a heurística i não tenha sido chamada nenhuma vez durante o segmento, sua pontuação permanecerá a mesma do segmento imediatamente anterior. Esta estratégia está intimamente relacionada com os conceitos de memória de curto e longo prazo, comuns em otimização combinatória.

$$r_{i,j+1} = \begin{cases} r_{i,j} & \text{Se } a_i = 0. \\ \rho\frac{\overline{r}_{i,j}}{a_i} + (1-\rho)r_{i,j} & \text{Se } a_i \neq 0. \end{cases} \tag{3.3}$$

Cada nova solução gerada pelo ALNS com o custo maior que o da solução atual passa por um critério de aceitação dinâmico dado por um método similar ao do *simulated annealing*, que restringe gradativamente a qualidade das soluções. Desta forma, 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). Utiliza-se uma taxa de resfriamento exponencial, em que a temperatura T, maior que zero, decresce a cada iteração de acordo com a expressão $T = T \times (T_{end}/T_{start})^{1/k}$), na qual k é o número máximo de iterações.

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

A temperatura inicial é definida por $T_{start} = -p_s \times f(\pi_0)/\ln 0,5$ em que $p_s \in [0..1]$ representa o percentual de piora inicial. Uma variante do cálculo da temperatura final, proposta por Santini, Ropke e Hvattum (2016), foi implementada neste trabalho. Esta 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 a temperatura final é atualizada como $T_{end} = -p_e \times f(\pi^*)/\ln 0,5$, em que p_e representa o percentual de piora final, também definido no intervalo [0..1].

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). Em seguida, um laço de repetição (linhas 3 a 23) 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 4), e em seguida aplicadas à solução atual π para gerar uma nova solução π' (linha 5). Posteriormente, o resultado de π' é comparado com a solução corrente π (linhas 6 a 13) e, caso seja melhor, se torna a nova solução

corrente (linha 7). Uma segunda comparação é feita para checar se a solução corrente π é também melhor que a melhor solução π_{best} (linhas 9 a 12). Se for melhor, π_{best} é atualizada (linha 10). No entanto, se π' possuir valor igual ou pior que a solução corrente, deve-se avaliá-la de acordo com o critério de aceitação (linhas 14 a 17) e, caso seja aceita, se torna a nova solução corrente (linha 15). Após a análise da nova solução, a pontuação das heurísticas são alterados (linha 18) conforme o valor recebido na linha 8, 11 ou 16. Ao final de cada segmento (linha 19), a pontuação das heurísticas são suavizadas (linha 20) e as probabilidades na roleta recalculadas (linha 21). A cada iteração, a última operação consiste em resfriar a temperatura T (linha 23). Por fim, quando o laço de repetição termina a melhor solução π_{best} é retornada (linha 25).

Algoritmo 1: Busca Adaptativa em Grandes Vizinhanças.

Input: solução viável π;
$\pi_{best} \leftarrow \pi$;
while *critério de parada não for atingido* **do**
    Escolha uma heurística de remoção N^- e uma heurística de inserção N^+ usando a roleta de seleção baseado nos pontos obtidos anteriormente $\{r_j\}$;
    Gere uma solução nova π' a partir de π usando a heurística de remoção e inserção escolhidas;
    if $CW(\pi') < CW(\pi)$ **then**
        $\pi \leftarrow \pi'$;
        $sigma \leftarrow \sigma_2$;
        if $CW(\pi) < CW(\pi_{best})$ **then**
            $\pi_{best} \leftarrow \pi$;
            $sigma \leftarrow \sigma_1$;
        end
    end
    else if π' *atender o critério de aceitação* **then**
        $\pi \leftarrow \pi'$;
        $sigma \leftarrow \sigma_3$;
    end
    Atualize os pontos r_j de N^- e N^+ conforme *sigma*;
    if *contador de interação for múltiplo de* θ **then**
        Suavize os pontos r_j acumulados;
        Atualize a probabilidade das heurísticas na roleta conforme nova pontuação;
    end
    $T = T \times (T_{end}/T_{start})^{1/k}$);
end
return π_{best};

O ALNS foi escolhido para este trabalho devido à 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 das vizinhanças e também à variedade das mesmas, 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 os quais vizinhanças pequenas não são suficientes para escapar de ótimos locais.

4 Desenvolvimento

Neste capítulo são descritas com detalhes as particularidades do ALNS implementadas nesse trabalho, incluindo critérios de parada do método e a implementação das heurísticas para remoção e inserção de vértices. As próximas seções descrevem seis métodos, quatro de remoção e dois de inserção, que são combinados pelo ALNS para gerar diferentes buscas locais.

4.1 Solução Inicial

O ALNS depende de uma solução inicial para iniciar sua execução. Neste trabalho, a solução inicial utilizada é gerada por um método guloso proposto por Pantrigo et al. (2012), baseado na metodologia GRASP. Especificamente, trata-se do método C1, que possui os melhores valores referentes a qualidade da solução.

A solução é construída gradativamente. O primeiro vértice é selecionado aleatoriamente dentre os vértices de menor grau de adjacência. Em seguida, uma lista de candidatos é formada por todos os vértices adjacentes aos vértices presentes na solução. O valor da largura de corte é separadamente calculado com base na inserção de cada vértice e uma lista restrita é criada com os melhores candidatos. Por fim, um vértice é selecionado aleatoriamente da lista restrita e inserido na solução. A complexidade de tempo de construir a solução inicial é limitada por $O(n(m+n))$.

4.2 Critério de Aceitação

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 metodologias para comparar soluções com ordens de rotulação f e f'. Desta forma, é possível diferenciar soluções que poderiam ser caracterizadas como idênticas dado o valor de solução.

Inicialmente, f e f' são comparadas pelo valor da largura de corte máxima, conforme a Equação (3.1). Se $CW_{f'} < CW_f$, então f' é aceita. Caso $CW_{f'} = CW_f$, a segunda função de avaliação é utilizada.

A segunda função de avaliação, denominada CW^2 e baseada no método proposto por Pardo et al. (2013), analisa a cardinalidade das diferentes larguras de corte em um grafo. Para este fim, os vértices são agrupados de acordo com sua largura de corte, sendo S^i o conjunto de vértices v tal que $CW(v) = i$. Temos que $CW^2 = \sum_{i=0}^{i_{max}} i \times |S^i|$, em

que i_{max} é o valor da largura de corte máximo do grafo. Por exemplo, na Figura 3 temos $S^1 = 1$, $S^3 = 2$ e $S^5 = 2$ o que resulta em $1 \times 1 + 3 \times 2 + 5 \times 2 = 17$. Se trocarmos de lugar o vértice D com o vértice A teríamos $S^2 = 1$, $S^3 = 2$ e $S^5 = 2$, que resulta no mesmo valor de largura de corte máximo, no entanto na nova formulação o valor seria $2 \times 1 + 3 \times 2 + 5 \times 2 = 18$.

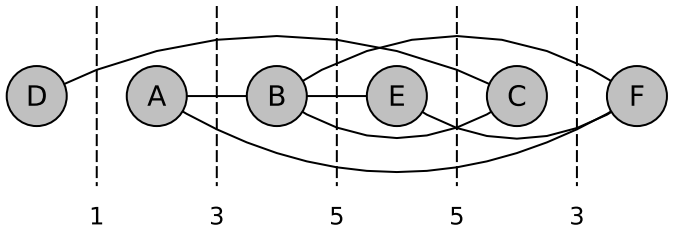


Figura 3 – Grafo G com seis vértices e sete arestas disposto na ordem f com as correspondentes larguras de cortes.

4.3 Critérios de Parada

O ALNS implementado neste trabalho possui duas condições de parada. A primeira condição consiste em temperatura T igual ou menor que 0.01, dado que uma temperatura muito baixa impede que soluções de piora sejam aceitas, fazendo com que o algoritmo tenha mais dificuldade de sair de ótimos locais. O segundo critério consiste em alcançar um número máximo de iterações, determinado *a priori*.

4.4 Heurísticas de Remoção

Cada heurística de remoção recebe uma solução, representada por uma permutação dos vértices, e cria um conjunto de vértices que posteriormente devem ser reinseridos um a um na solução por uma heurística de inserção. A Figura 3 será utilizada como referência dos exemplos para algumas das quatro heurísticas de remoção utilizadas.

4.4.1 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 3, 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.

4.4.2 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).

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

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 $[0, 1]$. Por ter 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.

4.4.3 Remoção de Vértices Relacionados

Nesta heurística, remoção de vértices relacionados (RShaw), são removidos vértices que são de alguma forma relacionados. A relação neste caso é medida através do valor da largura de corte. Um vértice é aleatoriamente selecionado, então os vértices que possuem o mesmo valor de largura de corte são também removidos. Por exemplo, na Figura 3, se o vértice C (CW(C) = 3) for selecionado para ser removido, o vértice A o será também por possuir o mesmo valor de largura de corte.

4.4.4 Remoção de Vértices Desbalanceados

Conforme Pardo et al. (2013), o $grau(v)$ de um vértice v indica a quantidade de arestas incidentes a ele, podendo ser par ou ímpar. Em um leiaute linear f, cada vértice v possui uma quantidade de arestas incidentes pela esquerda, dada por $grauL(v)$, e uma quantidade de arestas incidentes pela direita, dada por $grauR(v)$, tal que $grau(v) = grauL(v) + grauR(v)$.

A última heurística de remoção propõe remover todos os vértices que estão desbalanceados, ou seja, possuem diferença entre $grauL()$ e $grauR()$. O desbalanceamento de um vértice v depende se o $grau(v)$ tem valor par ou ímpar. Caso o vértice tenha um número par de adjacências, o mesmo está desbalanceado se $grauL(v) \neq grauR(v)$. Por outro lado, se o grau de adjacências do vértice é ímpar, o mesmo está balanceado se $|grauL(v) - grauR(v)| > 1$.

Por exemplo, na Figura 3 os vértices A ($grauL(A) = 0$ e $grauR(A) = 2$), B ($grauL(B) = 1$ e $grauR(B) = 3$) e C ($grauL(C) = 2$ e $grauR(C) = 0$) possuem grau par e estão deslocados. Na mesma figura, o vértice F na Figura 3 possui grau ímpar e está deslocado ($|grauL(v) - grauR(v)| = 3$).

Esta vizinhança foi decomposta em três vizinhanças diferentes: a primeira remove todos os vértices desbalanceados com número par de vértices adjacentes (RDP); a segunda remove os vértices desbalanceados com número ímpar de vértices adjacentes (RDI); e a terceira remove todos os vértices desbalanceados (RD).

4.4.5 Análise de complexidade do pior caso das heurísticas de remoção

As heurísticas de remoção introduzidas nas Seções 4.4.1 e 4.4.3 precisam calcular o valor da largura de corte para cada um dos vértices. Isso quer dizer que para cada vértice todas as arestas incidentes ao mesmo devem ser analisadas. Essa operação tem complexidade limitada por $O(nm)$ e define a complexidade das vizinhanças. A Seção 4.4.2 descreve uma heurística que remove q vértices aleatoriamente, e para que essa remoção seja possível é necessário embaralhar a ordem dos vértices (operação com complexidade $O(n)$). Portanto sua complexidade é dada pelo custo de embaralhar a ordem dos vértices. Por fim, a heurística apresentada na seção 4.4.4 é análoga a primeira heurística e deve analisar toda a solução para determinar os vértices desbalanceados. Consequentemente sua complexidade é $O(nm)$.

4.5 Heurísticas de Inserção

Cada heurística de inserção recebe uma solução parcial e um conjunto de vértices para reinserção na solução. Cada vértice do conjunto é selecionado aleatoriamente e reinserido um por vez na solução. Embora outras heurísticas tenham sido consideradas, apenas duas se destacaram em experimentos preliminares.

4.5.1 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.

4.5.2 Inserção na Melhor Posição

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, conforme (PARDO et al., 2013). Por exemplo, se o vértice C da Figura 3 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. No caso de vértices com grau ímpar, como por exemplo o vértice F, estes devem ser inseridos

imediatamente antes ou depois do vértice adjacente central. Por exemplo, o vértice F é adjacente aos vértices A, B e E, desta forma F deve ser inserido imediatamente antes ou depois do vértice B.

Esta heurística foi implementada de duas formas diferentes: uma delas insere o vértice na melhor posição disponível (IBest_{CM}) e mantém a escolha somente se houver melhoria na função objetivo, e a outra insere o vértice na melhor posição disponível (IBest_{SM}) independente do valor final da função objetivo.

4.5.3 Análise de complexidade do pior caso das heurísticas de inserção

A primeira heurística de inserção introduzida neste trabalho, descrita na Seção 4.5.1, insere elementos na solução em posições aleatórias e não avalia a função objetivo do problema. Esta heurística portanto, tem complexidade $\Theta(q)$ em que q é a quantidade de vértices a serem inseridos na solução.

A heurística apresentada na Seção 4.5.2 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 de adjacências par e vértices com número de adjacências ímpar. Para os vértices com número de adjacências par, o pior caso é dado quando suas adjacências se encontram na extremidade da ordem f. 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 devem ser feitas, portanto a complexidade é limitada por $O(nm)$.

5 Experimentos Computacionais

Os resultados e análises dos experimentos computacionais realizados são apresentados neste capítulo. O ambiente computacional consiste em um computador com processador *Intel Core i7* 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 três conjuntos de instâncias da literatura, *Small*, *Grid* e *Harwell-Boeing*, todos eles disponíveis em Optsicom (2017). Os resultados obtidos foram comparados com os resultados dos trabalhos considerados estado da arte: Scatter search (PANTRIGO et al., 2012) e VFS (PARDO et al., 2013). 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 proporções que variam entre $16 \leq n \leq 24$ e $18 \leq m \leq 49$. Vários autores, entre eles Martí et al. (2013), Coudert (2016) e Pantrigo et al. (2012), foram capazes de encontrar todas as soluções ótimas para este conjunto de instâncias.

O conjunto *Grid*, proposto por Raspaud, Sỳkora e Vrt'o (1995), é composto por 81 matrizes que representam grades bidimensionais com dimensões entre 9×9 a 729×729. 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.

O último conjunto, *Harwell-Boeing* (HB), é um subconjunto derivado do *Harwell-Boeing Sparse Matrix Collection* disponível em domínio público no *Matrix Market Library* (LIBRARY, 2007). Esta coleção consiste em um conjunto de matrizes de teste $M = \{m_{ij}\}$ 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.

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 demonstra os intervalos de valores que o *irace* considerou ao definir cada um dos parâmetros.

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}

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

- $\sigma_1 = 50$;
- $\sigma_2 = 15$;
- $\sigma_3 = 25$;
- 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 $\theta = 200$.

5.3 Comparação com os Resultados da Literatura

Um resumo dos resultados médios do ALNS podem ser encontrados 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	5,32	8,00	0,00	0,02	0,12
Grid	11,56	12,94	14,35	16,33	26,20	7,54	0,92	14,06
HB	311,55*	314,03	318,57	358,86	14,00	2,40	3,38	202,71

*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 8%, o que demonstra a importância do uso do ALNS mesmo para este conjunto com instâncias pequenas. O segundo conjunto de instâncias, *Grid*, foi o conjunto com o maior *gap* médio, 7,54%. Apesar disto, o método apresentou uma boa melhora entre a solução inicial e a melhor solução: 26,20%. Este conjunto obteve tempo médio de execução igual a 14 segundos, sendo o valor máximo de 177 segundos. Este é considerado satisfatório para um conjunto que contém instâncias de até 729 vértices. As soluções geradas para os conjuntos *Grid* e *Small* apresentaram um baixo valor de desvio padrão, o que significa que o método alcançou valores próximos durante as 10 execuções. Apesar do *gap* para o conjunto *Grid* ser alto, o resultado é competitivo com os outros métodos apresentados na literatura que possuem *gap* no intervalo de 3,25 a 201%.

O método proposto obteve um *gap* médio de 2,40% para o conjunto HB, o conjunto de maiores dimensões, e um *gap* total entre a solução inicial e a melhor solução de 14%. Como já era esperado, o método exigiu maior esforço computacional neste conjunto, com tempo médio de execução de 202 segundos. Embora baixo, o desvio padrão também foi maior do que os outros dois conjuntos anteriores, pouco mais de 3%. Das 87 instâncias do conjunto HB, o ALNS alcançou para 61 (70%) valores iguais aos melhores valores reportados na literatura.

Na Tabela 3 os melhores resultados do ALNS são comparados com os métodos da literatura que compõem o estado da arte, o *Variable Formulation Search* (VFS) (PARDO et al., 2013) e o *Scatter Search* (SS) (PANTRIGO et al., 2012). A coluna S apresenta a média das soluções 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 do método e a melhor solução da literatura.

Tabela 3 – Comparação com os métodos estado da arte.

	VFS			SS			ALNS		
	S	$\#OPT$	gap	S	$\#OPT$	gap	S	$\#OPT$	gap
Small	–	–	–	4,92	84	0,00	4,92	84	0,00
Grid	12,23	59	3,25	13,00	44	7,76	12,94	42	7,54
HB	314,39	61	1,77	315,22	59	3,40	314,03	61	2,40

Assim como o SS, o ALNS foi capaz de encontrar todos os valores ótimos das instâncias do conjunto *Small*, que não foi considerado pelo VFS. No entanto, o ALNS superou o valor de soluções médias do SS nos outros dois conjuntos, *Grid* e HB. O *gap* total entre a melhor solução do SS e a melhor do ALNS é igual a 0,46% e 0,38% para os respectivos conjuntos. Ao comparar os resultados com o VFS, o ALNS obteve soluções piores, com *gap* total 5,80% no conjunto *Grid*, mas superou o VFS com *gap* de 0,11% no conjunto HB. Devido ao fato dos testes serem executados em arquiteturas diferentes, a comparação de tempo não pode ser feita de forma justa.

Além disso, testes estatísticos adicionais foram aplicados aos resultado do ALNS e do VFS no conjunto HB para melhor observar as diferenças dos mesmos. O *Shapiro-Wilk Test* (SHAPIRO; WILK, 1965) foi aplicado e determinou que os resultados não são normalmente distribuídos (p-*value* = 2,2e-16). 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. O teste demonstrou que não há diferença significativa entre os resultados (p-*value* = 0,7333). Contudo, o ALNS demostra ser competitivo em questão da qualidade das soluções geradas.

5.4 Análises Adicionais

No Capítulo 4, vizinhanças 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. Por exemplo, otimizar as vizinhanças com uma alta pontuação ou retirar as com baixa pontuação. 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. A Tabela 4 apresenta a distância percentual entre os pontos de cada uma das vizinhanças para cada grupo de instâncias. A vizinhança com a maior pontuação tem distância percentual igual a zero.

Tabela 4 – Distância percentual entre os pontos das vizinhanças. As maiores pontuações tem *gap* zero.

	Vizinhanças de Remoção						Vizinhanças de Inserção		
	RVC	RRand	RShaw	RDP	RDI	RD	IRand	IBest$_{CM}$	IBest$_{SM}$
Small	24,28	71,52	0,00	19,64	6,21	40,70	61,48	0,00	12,64
Grid	15,84	123,74	0,00	53,71	24,05	90,54	80,36	0,00	23,91
HB	17,06	113,09	0,00	92,60	62,16	136,62	31,41	0,00	18,77
Média	19,06	102,78	0,00	55,32	30,81	89,29	57,75	0,00	18,44

A vizinhança de remoção de vértices relacionados (RShaw) obteve a melhor pontuação em todos os conjuntos de instâncias, enquanto a remoção aleatória (RRand) obteve os piores valores, cerca da metade da vizinhança RShaw. É compreensível pensar que a remoção aleatória obtivesse os piores resultados, tendo em vista que esta vizinhança utiliza aleatoriedade como métrica de inserção. O mesmo aconteceu com as vizinhanças de inserção, em que a inserção aleatória obteve a pior pontuação. Esta vizinhança de inserção está a 57% de distância da melhor vizinhança (IBest$_{CM}$).

O teste paramétrico de Análise de Variância (ANOVA) foi aplicado para verificar se existe diferença significativa entre os pontos das vizinhanças de remoção e inserção. O teste resultou em $p\text{-}value = 0,000219$ para as vizinhanças de remoção e $p\text{-}value = 0,0285$ 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 vizinhanças com diferença significativa. As Tabela 5 e 6 demonstram onde se encontram tais diferenças, em que os valores em negritos representam diferença significativa entre duas vizinhanças, i.e., $p\text{-}value < 0,05$.

Tabela 5 – *P-value* do *Pairwise t-test* no conjunto de vizinhanças de remoção. Valores em negrito indicam que existe diferença significativa entre o método da coluna e da linha.

	RVC	RRand	RShaw	RDP	RDI
RRand	**0,0109**	-	-	-	-
RShaw	0,9090	**0,0004**	-	-	-
RDP	0,5316	0,8151	**0,0121**	-	-
RDI	1,0000	**0,0467**	0,2052	1,0000	-
RD	**0,0333**	1,0000	**0,0010**	1,0000	0,1491

Tabela 6 – *P-value* do *Pairwise t-test* no conjunto de vizinhanças de inserção. Valores em negrito indicam que existe diferença significativa entre o método da coluna e da linha.

	IRand	IBest$_{CM}$
IBest$_{CM}$	**0,031**	-
IBest$_{SM}$	0,232	0,516

As vizinhanças de remoção com maior discrepância entre outras vizinhanças são RShaw e RRand, ambas com diferença significativa para outras três vizinhanças. As outras vizinhanças se diferem significativamente de uma ou duas dentre as quatro outras vizinhanças. Para as vizinhanças de inserção, somente a vizinhança IRand e $IBest_{CM}$ apresentaram diferença significativa.

A Figura 4 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 a 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. Os pontos vermelhos correspondem aos *outliers*, ou valores atípicos, valores que apresentam um grande afastamento dos demais.

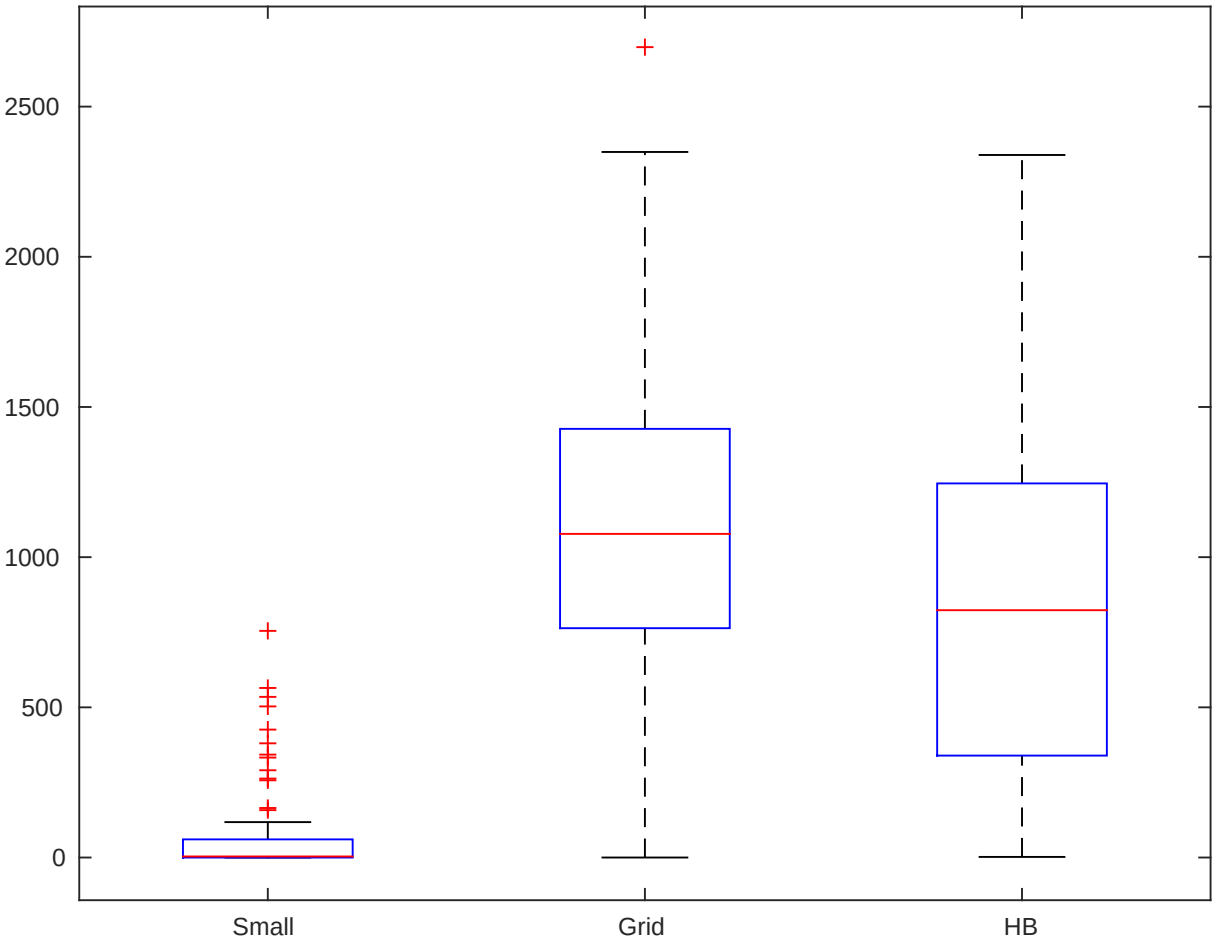


Figura 4 – Análise de convergência do ALNS para os conjuntos de instâncias Small, Grid e *Harwell-Boeing* (HB).

O conjunto *Small* possui baixos valores de convergência, o que era esperado por ser um conjunto de dimensões pequenas. Em média o algoritmo converge na iteração de número 78 para este conjunto, sendo que para 29 instâncias a solução ótima equivale a

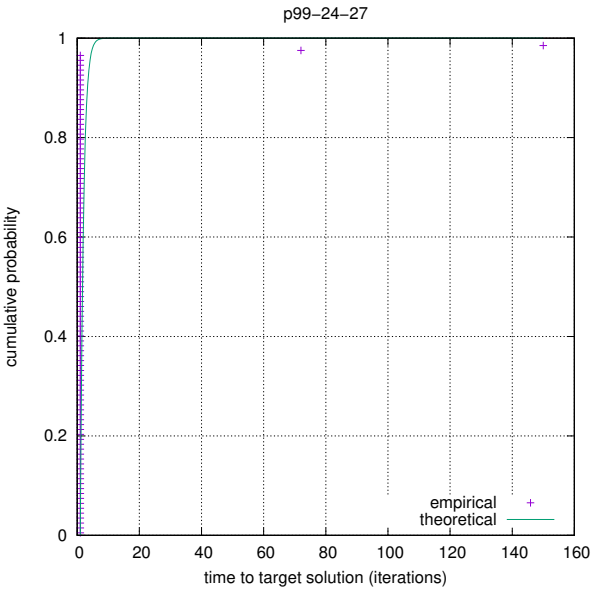
solução inicial, o que explica o baixo valor da mediana. Um número pequeno de *outliers* pode ser observado, sendo que o maior deles é 754.

Os conjuntos *Grid* e *Harwell-Boeing* (HB), por serem os dois conjuntos de maior dificuldade de solução e também com as maiores instâncias, apresentam um espalhamento maior dos seus valores. O primeiro obteve média igual a 1019, um valor bem próximo de sua mediana, e possui valores que vão de 1 até aproximadamente 2300 (desconsiderando o único *outlier*, 2698).

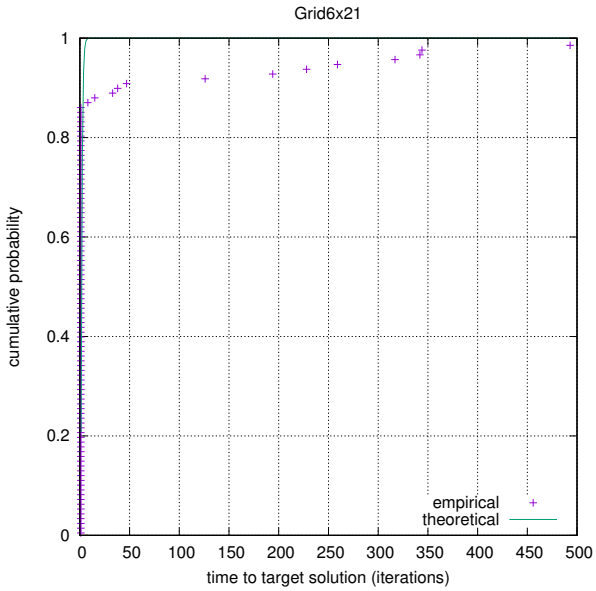
O último conjunto representado possui sua mediana bem próximo do valor da média (879) e não apresenta *outliers*, o que significa que possui a mesma quantidade de valores abaixo e acima da mediana. O conjunto apresenta valor mínimo de 2 iterações e máximo de 2339. O limite de iterações no ALNS implementado 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 2500 iterações, desconsiderando o *outlier* do conjunto *Grid*.

A fim de ilustrar a convergência do ALNS, uma instância de cada conjunto foi aleatoriamente selecionada para gerar gráficos *time-to-target* (ttt) (AIEX; RESENDE; RIBEIRO, 2007), vide Figura 5. 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.

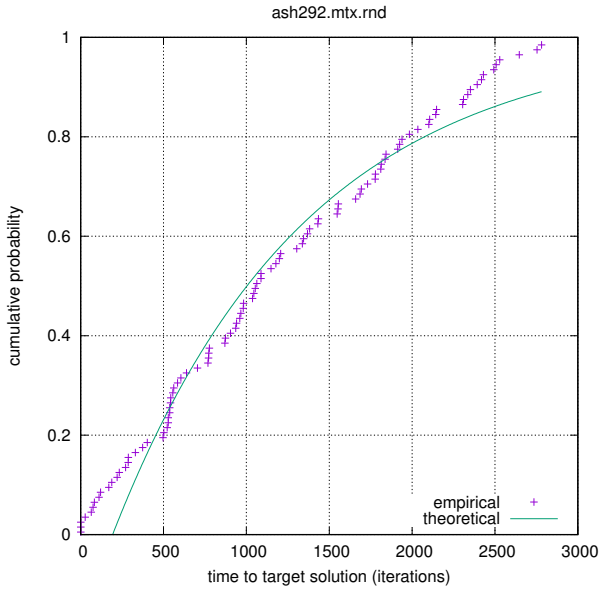
Em seguida, comparamos a distribuição resultante (*empirical*, dado pelas cruzes roxas) com a distribuição exponencial (*theoretical*, representada pela linha verde). Os gráficos *ttt* apresentam a probabilidade cumulativa (eixo y) de atingir uma solução alvo em cada número de iterações (eixo x). Por exemplo, a Figura 5 (a) e (b) mostram 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, na Figura 5 (c) é preciso de 2500 iterações para encontrar a solução alvo com a mesma probabilidade anterior.



(a) Instância *Small*.



(b) Instância *Grid*.



(c) Instância HB.

Figura 5 – Time-to-target plots de instâncias aleatoriamente selecionadas.

6 Plano de Atividades Restantes

Na Tabela 7 são apresentadas informações sobre atividades planejadas para os trabalhos futuros da dissertação de mestrado, visando a conclusão deste trabalho de pesquisa.

Tabela 7 – Cronograma de atividades restantes.

Tarefas	Bimestre 1	Bimestre 2	Bimestre 3	Bimestre 4	Bimestre 5	Bimestre 6
Pesquisar novas heurísticas de inserção e remoção	✓	✓	✓			
Pesquisar sobre métodos de refinamento	✓	✓	✓			
Implementar novas heurísticas e métodos de refinamento		✓	✓	✓		
Realizar experimentos computacionais			✓	✓		
Realizar experimentos computacionais em novos conjuntos de instâncias			✓	✓		
Analisar experimentos computacionais realizados			✓	✓	✓	
Descrever os experimentos computacionais					✓	✓
Escrever artigo científico					✓	✓
Conclusão do texto da dissertação					✓	✓
Realização do exame de defesa						✓

As atividades basicamente se concentram em pesquisar, implementar e analisar métodos de melhoria fina para o método proposto, assim como novas heurísticas de remoção e inserção. Após a definição e implementação dessas melhorias, novos experimentos incluindo novos conjuntos de instâncias serão executados e analisados. Além da realização deste trabalho de dissertação, tem-se também como objetivo a confecção de um artigo cientifico para submissão em simpósios e revistas de impacto.

7 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 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*). É importante ressaltar que esta é a primeira aplicação reportada do ALNS para solução do CMP. Os experimentos computacionais envolveram 252 instâncias de 3 diferentes conjuntos da literatura. Os resultados obtidos foram comparados com as melhores soluções encontradas na literatura e também com os métodos que compõem o estado da arte. O método proposto alcançou uma quantidade significativa de soluções ótimas, demonstrando ser competitivo em termos da qualidade das soluções geradas e do tempo de execução. Os trabalhos futuros serão concentrados em melhorias finas do método proposto, como a aplicação de um método de busca local e criação de novas estruturas de vizinhança, no intuito de obter melhores resultados.

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 21.

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 51.

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 27 e 28.

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 22 e 27.

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 26.

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 27.

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 25.

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 21.

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 22.

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 25.

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 21 e 33.

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 4 vezes nas páginas 22, 25, 26 e 45.

COUDERT, D. *Linear ordering problems on graphs*. 2017. Disponível em: <http://www-sop.inria.fr/members/David.Coudert/code/graph-linear-ordering.shtml>. Citado na página 25.

CRAMÉR, H. *Mathematical methods of statistics (PMS-9)*. [S.l.]: Princeton university press, 2016. v. 9. Citado na página 49.

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 22.

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 21.

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 28.

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 na página 28.

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 22.

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 26.

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 28.

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 26.

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 22.

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 22.

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 32.

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 26.

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 25.

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 21.

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 32.

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 22.

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 26.

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 25.

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 21.

LIBRARY, M. M. *Matrix Market Library*. 2007. Disponível em: <http://math.nist.gov/MatrixMarket/index.html>. Citado 2 vezes nas páginas 25 e 45.

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 21.

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 21.

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 46.

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 2 vezes nas páginas 26 e 27.

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 28.

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 22 e 26.

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 na página 22.

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 3 vezes nas páginas 25, 26 e 45.

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 26 e 45.

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 22.

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 22.

OPTSICOM. *Optsicom project, University of Valencia*. 2017. Disponível em: <http://www.optsicom.es/cutwidth>. Citado 2 vezes nas páginas 25 e 45.

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 26.

PANTRIGO, J. J. et al. Scatter search for the cutwidth minimization problem. *Annals of Operations Research*, Springer, p. 1–20, 2012. Citado 5 vezes nas páginas 27, 28, 39, 45 e 47.

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 6 vezes nas páginas 28, 39, 41, 42, 45 e 47.

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 41.

RASPAUD, A. et al. Antibandwidth and cyclic antibandwidth of meshes and hypercubes. *Discrete Mathematics*, Elsevier, v. 309, n. 11, p. 3541–3552, 2009. Citado 2 vezes nas páginas 25 e 45.

RASPAUD, A.; SỲKORA, 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 25, 32 e 45.

REY, D.; NEUHÄUSER, M. Wilcoxon-signed-rank test. In: *International encyclopedia of statistical science*. [S.l.]: Springer, 2011. p. 1658–1659. Citado na página 48.

ROLIM, J.; SỲKORA, 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 25.

ROMERO-MONSIVAIS, H.; RODRIGUEZ-TELLO, E.; RAMÍREZ, G. A new branch and bound algorithm for the cyclic bandwidth problem. In: SPRINGER. *Mexican International Conference on Artificial Intelligence*. [S.l.], 2012. p. 139–150. Citado na página 26.

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 2 vezes nas páginas 22 e 33.

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 28.

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 36.

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 48.

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 na página 34.

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 22.

SỲKORA, 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 32.

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 25.

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 26.

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 32.

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 25.