Uma Heurística Adaptativa Aplicada à Minimização da Largura de Corte Em Grafos

Vinícius Gandra Martins Santos
Departamento de Ciência da Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil.
gandra.vinicius@gmail.com

Marco Antonio Moreira de Carvalho
Departamento de Ciência da Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil.
mamc@ufop.edu.br

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. Largura de Corte, Heurística, Otimização Combinatória.

Área Principal: TAG, MH, OC.

ABSTRACT

The Cutwidth Minimization Problem (CMP) consists in determining a linear layout for a graph in order to minimize a maximum linear cut of edges between each consecutive pair of vertices. This problem has application 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 included 252 instances from the literature and the results found are compared with the current state of the art. The proposed method showed to be competitive, as it was able to match most of the literature best results.

KEYWORDS. Cutwidth, Heuristics, Combinatorial Optimization.

Main Area: TAG, MH, OC.

1. Introdução

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, 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 um inteiro atribuído para o vértice consiste na posição ocupada pelo mesmo no leiaute. Em outras palavras, um leiaute linear consiste em uma permutação da ordem dos vértices.

Um grande número de problemas podem ser formulados como problemas de determinação de leiaute em grafos, como por exemplo, o desenho de grafos [Suderman, 2004], processamento de linguagem natural [Kornai e Tuza, 1992], o agendamento de migração de redes [Andrade e Resende, 2007b], o projeto de circuitos integrados de larga escala [Ohtsuki et al., 1979] e recuperação de informação [Botafogo, 1993], entre outros. Um revisão abrangente sobre problemas de leiaute em grafos é apresentada por Díaz et al. [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 cruzando cada par de vértices consecutivos é minimizado [Adolphson e Hu, 1973].

Em sua versão de decisão, o CMP é provado ser NP-Completo por Garey et al. [1976] e por Gavril [1977]. O problema continua sendo NP-Completo para grafos com grau máximo três [Makedon et al., 1985], grafos planares com grau máximo três [Makedon e Sudborough, 1989], grafos grade e grafos disco unitários [Dıaz et al., 2001], grafos direcionados acíclicos [Even e Shiloach, 1975] e para árvores ponderadas [Monien e Sudborough, 1988]. Na versão de otimização o CMP é provado ser NP-Difícil mesmo para grafos de grau máximo três [Makedon et al., 1985].

Em termos matemáticos, dado um grafo não dirigido $G = (V, E)$ com $|V| = n$ e $|E| = m$, uma rotulação ou leiaute 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 $f(u) \leq f(v) < f(w)$. Logo, a largura de corte $CW_f(G)$ é o máximo da largura de corte dos seus vértices, vide Equação (1).

$$CW_f(G) = \max_{v \in V} CW_f(v). \tag{1}$$

A largura de corte ótima de um grafo G é definida pelo mínimo valor de $CW_f(G)$ entre todas as possíveis rotulações. Em outras palavras o CMP consiste em encontrar o leiaute f que minimize $CW_f(G)$ dentre todos os possíveis, denotado por Π. A Equação (2) demonstra formalmente a função objetivo.

$$CW(G) = \min_{f \in \Pi} CW_f(G). \tag{2}$$

Na Figura 1(a) é apresentado um grafo com $n = 6$ e $m = 7$. A Figura 1(b) apresenta o mesmo grafo disposto linearmente com uma rotulação f. Note que $f(D) = 1$, o que quer dizer que o vértice D é o primeiro, seguido do vértice A ($f(A) = 2$) e assim por diante. Desta maneira f pode ser representado por $[D, A, B, E, C, F]$. 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 $CW(A) = 3$, o que significa que existem três arestas com uma extremidade antes de A (ou incidente a este mesmo vértice) e com a outra extremidade depois de A (arestas $\{D, C\}$, $\{A, B\}$ e $\{A, F\}$). Neste exemplo em particular, temos $CW_f(G) = CW_f(B) = CW_f(E) = 5$.

Para a solução do CMP, este trabalho propõe implementar o método Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS), uma metaheurística recente que utiliza buscas locais e perturbações para explorar uma porção ampla das possíveis soluções para

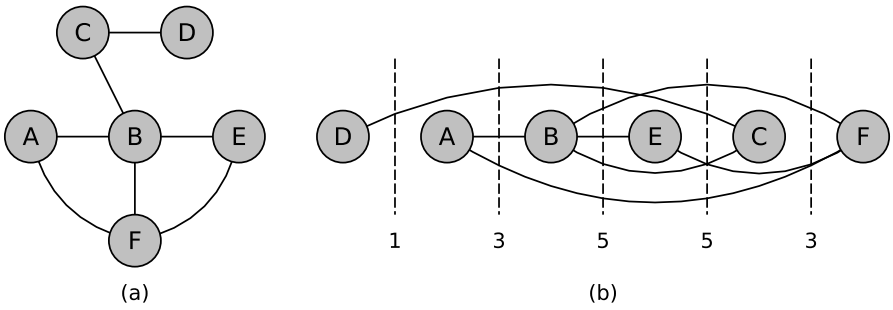

Figura 1: (a) Grafo G com seis vértices e sete arestas. (b) Ordem f dos vértices do grafo (a) com as correspondentes larguras de cortes.

problemas combinatórios. Este algoritmo é robusto, se adapta a várias características das instâncias e dificilmente fica preso em ótimos locais. Esta é a primeira aplicação do método ALNS ao CMP reportada na literatura.

Este trabalho está organizado como a seguir. Uma breve revisão da literatura é apresentada a Seção 2. O desenvolvimento está descrito na Seção 3, em que o método proposto está detalhadamente ilustrado. A Seção 4 demonstra os resultados obtidos pelo trabalho proposto considerando instâncias da literatura. Por fim, a conclusão e propostas de trabalhos futuros são apresentados na Seção 5.

2. Revisão da Literatura

Algumas topologias específicas de grafos permitem formulações que resolvam o CMP em tempo determinístico polinomial. Por exemplo há algoritmos cuja complexidade é limitada por $O(nlogn)$ para árvores não ponderadas [Yannakakis, 1985] e árvores de grau três [Chung et al., 1985]. Há algoritmos lineares para árvores completas com grau fixo [Lengauer, 1982], e hipercubos [Harper, 1964]. Por fim, Thilikos et al. [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.

Algoritmos aproximados para o CMP foram propostos por Leighton e Rao [1999] e Arora et al. [2002]. O primeiro trabalho aborda grafos gerais e consiste em um algoritmo de aproximação polinomial com relação de aproximação limitada por $O(CWlog^2n)$, em que CW é o valor ótimo ou limite inferior da largura de corte. O segundo trabalho propõe um algoritmo polinomial aproximado para grafos densos, cuja complexidade de tempo é limitada por $n^{O(logn/\epsilon^2)}$ para $\epsilon > 0$, e cuja relação de aproximação é de $\epsilon \times n^2$.

Uma adaptação do método *branch-and-bound* (B&B) foi proposto por Palubeckis e Rubliauskas [2012] e testado usando grafos gerados aleatoriamente. Como resultado, o método foi capaz de encontrar valores ótimos para instâncias com até 35 vértices para grafos densos e até 50 vértices para grafos esparsos. Posteriormente, Martí et al. [2013] propuseram outro método derivado do B&B e utilizaram outros três conjuntos de instâncias da literatura para os experimentos computacionais: *Small*, *Grid* e HB, presentes em Optsicom [2017]. O método foi capaz de encontrar soluções ótimas apenas para as menores instâncias.

López-Locés et al. [2014] propuseram um modelo de programação inteira para solução do CMP. Os resultados do modelo foram comparados com os resultados de outro modelo inteiro proposto anteriormente pelos mesmos autores e foi possível aferir uma melhora de 38% nos resultados. Recentemente, Coudert [2016b] propôs um novo modelo de programação linear inteira. O modelo foi testado com dois conjuntos de instâncias da literatura, *Small* e *Rome Graphs* [Coudert, 2016a], e alcançou todas as soluções ótimas para o primeiro conjunto, de menor tamanho. Para o segundo conjunto de instâncias, o modelo conseguiu alcançar 49% das soluções ótimas. De acordo

com os resultados, este modelo matemático demonstrou melhor desempenho em relação ao modelo de López-Locés et al. [2014].

Métodos heurísticos e metaheurísticos foram amplamente utilizados, principalmente na literatura mais recente. Métodos baseados no *Greedy Randomized Adaptive Search Procedure* (GRASP) foram propostos por Andrade e Resende [2007a,b]. Os métodos foram testados com os conjuntos de instâncias *Small*, *Grid* e HB porém, não foram capazes de encontrar todas as soluções ótimas nem mesmo para o conjunto de instâncias consideradas pequenas.

Um algoritmo híbrido evolutivo foi proposto por Bansal et al. [2012], em que a solução inicial do algoritmo genético é gerada através de uma busca em profundidade aleatorizada. Os experimentos consideraram grafos padrão e os resultados foram comparados com outra versão de algoritmo genético, apresentando bons resultados. 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 que envolve combinação de soluções e busca local. O *Scatter Search* alcançou todas as soluções ótimas do conjunto *Small*, além de 44 soluções ótimas dentre 81 instâncias do conjunto *Grid* e 59 soluções ótimas dentre as 87 instâncias do conjunto HB com tempo médio de execução de 213 segundos.

No mesmo período, diferentes métodos baseados na metaheurística *Variable Neighborhood Search* (VNS) foram propostos. 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 formulações diferentes do CMP para diferenciar soluções com o mesmo valor de função objetivo. Essa técnica permite lidar com espaços de busca em que a superfície contém platôs, como é o caso do CMP. Desta forma, o algoritmo é capaz de distinguir duas soluções com o mesmo valor de função objetivo e escolher a mais promissora para continuar a busca. O VFS considera os conjuntos de instâncias *Grid* e HB, e compara os resultados obtidos com o método GRASP [Andrade e Resende, 2007b] e *Scatter Search* [Pantrigo et al., 2012], sendo capaz de superá-los em qualidade da solução e tempo de execução. Adicionalmente, o VFS foi capaz de encontrar 59 soluções ótimas no conjunto *Grid* e outras 61 no conjunto HB, ambos em tempo médio próximo a 90 e 120 segundos, respectivamente.

Outra variação do VNS, proposta por Duarte et al. [2013], consiste em uma implementação com diferentes políticas de paralelização, que consistem em paralelizar diferentes fases do algoritmo. Os experimentos computacionais consideraram 101 grafos retirados do *HB Sparse Matrix Collection* e concluíram que a paralelização da fase de perturbação é a melhor abordagem. Esta versão foi comparada ao VFS [Pardo et al., 2013], obtendo melhores resultados e melhor tempo de execução no conjunto de instâncias utilizado. No entanto, não há experimentos disponíveis para este método considerando o conjunto de instâncias *Grid*, o que permitiria uma melhor comparação entre os métodos.

O método metaheurístico mais recente para o CMP consiste em um Algoritmo Genético *Fuzzy* (GAF) [Fraire-Huacuja et al., 2017], que emprega um controlador com lógica *fuzzy* para determinar os parâmetros do algoritmo durante a sua execução. O método proposto utilizou em seus experimentos as instâncias dos conjuntos *Small* e *Grid*. O GAF quando comparado ao VFS e ao *Scatter Search* obteve uma distância percentual superior a 85% no conjunto *Grid*, além de possuir um maior tempo de execução. Adicionalmente, o método não foi capaz de encontrar todas as soluções ótimas para o conjunto *Small*, que é considerado trivial por possuir instâncias pequenas.

3. Busca Adaptativa em Grandes Vizinhanças

Ropke e Pisinger [2006] introduziram a metaheurística Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS). Este método utiliza um conjunto de diferentes heurísticas para efetuar buscas locais simples e rápidas, que competem entre si para modificar uma solução explorando suas vizinhanças através de movimentos de remoção e inserção de elementos. O algoritmo utiliza um conjunto N^- de vizinhanças de remoção, induzidas por

heurísticas que removem q elementos da solução, e um conjunto N^+ de vizinhanças de inserção, induzidas por heurísticas que reinserem elementos e constrói uma nova solução dada uma solução parcial. Essas heurísticas são geralmente baseadas em métodos gulosos de bom desempenho para o problema em questão. Neste contexto, os termos vizinhanças e heurísticas são usados de forma equivalente.

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. 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 vizinhança ao longo das iterações e atribuir pontos para cada uma de acordo com o seu desempenho, tendo em vista que quanto mais a vizinhança $N_i \in \{N^+ \cup N^-\}$ contribui para uma solução, maior será a quantidade de pontos atribuídos à mesma e consequentemente, maior a chance de ser selecionada em iterações futuras. A probabilidade da vizinhança N_i ser selecionada é dada pela divisão dos pontos da vizinhança N_i, denotados por r_i, pelo somatório dos pontos de todas as vizinhanças.

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 vizinhança i recebe uma fatia da roleta proporcional à sua probabilidade $P(N_i)$ de ser escolhida. Sendo assim, quanto maior for a probabilidade da vizinhança ser escolhida, maior será o a sua fatia de intervalo na roleta e consequentemente maior a chance da vizinhança ser selecionada. 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.

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 vizinhanças são cumulativamente calculados. Sendo $\overline{r}_{i,j}$ os *pontos observados* da vizinhança 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 vizinhança passam por um processo de *suavização*, calculado conforme a Equação (3), em que a_i é o número de vezes que a vizinhança 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 nos pontos de cada vizinhança. 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 vizinhança i não tenha sido chamada nenhuma vez durante o segmento, sua pontuação permanecerá a mesma do segmento imediatamente anterior.

$$r_{i,j+1} = \begin{cases} r_{i,j} & \text{Se } a_i = 0. \\ \rho\frac{\overline{r}_{i,j}}{a_i} + (1-\rho)r_{i,j} & \text{Se } a_i \neq 0. \end{cases} \tag{3}$$

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

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 et al. [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. 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 18) 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 atual π (linhas 6 a 11) e, caso seja melhor, se torna a nova solução atual (linha 7). Uma segunda comparação é feita para checar se a solução atual π é também melhor que a melhor solução π_{best} (linhas 9 a 11), caso no qual π_{best} é atualizado (linha 10). No entanto, se π' possuir valor igual ou pior que a solução atual, deve-se avaliá-la de acordo com o critério de aceitação (linhas 12 a 14) e, caso seja aceita, se torna a nova solução atual (linha 13). Após a análise da nova solução, os pontos das vizinhanças são alterados (linha 15) conforme o valor atribuído nas linhas 8, 11 ou 14. Ao final de cada segmento (linha 16), os pontos das heurísticas são suavizados (linha 17) e a probabilidade nas roletas recalculada (linha 18). Ao final, a melhor solução π_{best} é retornada (linha 19).

Algoritmo 1: ALNS.

```
1  Input: solução viável π;
2  π_best ← π;
3  while critério de parada não for atingido do
4      Escolha v_1 ∈ N^- e v_2 ∈ N^+ usando o método da roleta;
5      Gere uma solução nova π' pela aplicação de v_1 e v_2;
6      if f(π') < f(π) then
7          π ← π';
8          sigma ← σ_2;
9          if f(π) < f(π_best) then
10             π_best ← π;
11             sigma ← σ_1;
12     else if π' atender o critério de avaliação then
13         π ← π';
14         sigma ← σ_3;
15     Atualize os pontos r_j de v_1 e v_2 conforme sigma;
16     if contador de interação for múltiplo de θ then
17         Suavize os pontos r_j acumulados;
18         Atualize a probabilidade na roleta conforme nova pontuação;
19 return π_best;
```

3.1. Solução inicial

O ALNS depende de uma solução inicial para iniciar sua execução. Neste trabalho, a solução inicial utilizada consiste em 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 inserida na solução.

3.2. Critério de Aceitação

Como mencionado, o CMP possui platôs em seu espaço de soluções, o que implica em soluções de estrutura diferente e valores iguais. Levando esta característica em consideração, são utilizadas duas formas de comparar duas soluções f e f'. Desta forma, é possível diferenciar soluções que poderiam ser caracterizadas como idênticas.

Inicialmente, f e f' são comparadas pelo valor da largura de corte máxima, conforme a Equação (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 $CW^2 = \sum_{i=0}^{i_{max}} i * |S^i|$, em que i_{max} é o valor da largura de corte máximo do grafo. Por exemplo, na Figura 1(b) temos $S^1 = 1$, $S^3 = 2$ e $S^5 = 2$ o que resulta em $1 * 1 + 3 * 2 + 5 * 2 = 17$.

3.3. 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 não permite alterações na solução. O segundo critério consiste em alcançar um número máximo de iterações, determinado *a priori*.

3.4. Vizinhanças de Remoção

Cada vizinhança 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 vizinhança de inserção. A Figura 1(b) será utilizada como referência dos exemplos para algumas das quatro vizinhanças de remoção utilizadas.

3.4.1. Remoção de Vértices Críticos

Vértices críticos são aqueles que possuem o valor máximo da largura de corte. Esta vizinhança seleciona como candidatos para remoção todos os vértices críticos da solução. É plausível acreditar em uma melhora da solução ao reinserir todos seus vértices de valor crítico em uma melhor posição. Por exemplo, na Figura 1(b), 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.

3.4.2. Remoção Aleatória

Esta vizinhança 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), proposta por Pereira et al. [2015].

$$q = \lfloor n - \sqrt{(1-u)(n-1)^2} + 0.5 \rfloor \tag{4}$$

A Equação (4) 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 $[0, 1]$. Essa operação pode ter grande impacto quando o número de vértices é grande.

3.4.3. Remoção de Vértices Relacionados

Nesta vizinhança, são removidos vértices que são de alguma forma relacionadas. A similaridade 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 1(b), 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.

3.4.4. 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 vizinhança de remoção propõe remover todos os vértices que estão desbalanceados, ou seja, possuem diferença entre *grauL*() e *grauR*(). O desbalanceamento de um vértice pode ocorrer em duas situações. Na primeira situação, o *grau*(v) é par e *grauL*(v)$\neq$ *grauR*(v). Na segunda situação, *grau*(v) é ímpar e $|$*grauL*(v) $-$ *grauR*(v)$| > 1$.

Por exemplo, na Figura 1(b) 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 1(b) 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; a segunda remove os vértices desbalanceados com número ímpar de vértices adjacentes; e a terceira remove todos os vértices desbalanceados.

3.5. Vizinhanças de Inserção

Cada vizinhança 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 vizinhanças tenham sido consideradas, apenas duas se destacaram em experimentos preliminares.

3.5.1. Inserção Aleatória

A primeira vizinhança de inserção 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.

3.5.2. Inserção na Melhor Posição

Nesta vizinhança, 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 v, conforme [Pardo et al., 2013]. Por exemplo, se o vértice C da Figura 1(b) 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 vizinhança foi decompostas em duas vizinhanças: um delas insere o vértice na posição escolhida somente se houver melhoria na função objetivo, e a outra insere o vértice na posição escolhida independente do valor da função objetivo.

4. Experimentos Computacionais

O ambiente computacional utilizado nos experimentos computacionais consiste em um computador com processador *Intel Core i5* de 3.0 GHz, 8 GB RAM e 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*. Devido ao componente de aleatoriedade do ALNS proposto, foram realizadas 10 execuções independentes para cada uma das instâncias. O número de execuções foi escolhido com base na literatura.

4.1. Experimentos Preliminares

Os parâmetros usados pelo ALNS e seus respectivos valores são: σ_1, σ_2 e σ_3 com valores 50, 15 e 25 respectivamente; fator de reação ρ igual a 0,85; percentual de piora p_s igual a 0,85 e p_e igual a 0,45; número de iterações igual a 3000; e por fim, o tamanho de cada segmento θ igual a 200. Esses parâmetros foram determinados utilizando o método *offline* de configuração automática de algoritmos de otimização *irace* [López-Ibáñez et al., 2016].

Os experimentos computacionais consideraram três conjuntos de instâncias da literatura, *Small*, *Grid* e HB, todos disponíveis em Optsicom [2017]. O conjunto *Small*, proposto por Martí et al. [2008], foi introduzido no contexto do problema *bandwidth*. O conjunto possui 84 grafos com proporções que variam entre $16 \leq n \leq 24$ e $18 \leq m \leq 49$. O conjunto *Grid*, proposto por Raspaud et al. [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 conjunto *Harwell-Boeing* (HB), é um subconjunto derivado do *Harwell-Boeing Sparse Matrix Collection* disponível em domínio público no *Matrix Market Library* [Library, 2007]. Do conjunto original, 87 instâncias foram selecionadas e variam de 30 a 700 vértices e 46 a 41686 arestas.

4.2. Comparação com os Resultados da Literatura

Um resumo dos resultados médios do ALNS podem ser encontrados na Tabela 1. Esta tabela apresenta a melhor solução da literatura (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 da solução ótima e a melhor solução do ALNS (*gap*), o desvio padrão das 10 execuções (σ) e, por fim, o tempo médio de execução em segundos (T).

Tabela 1: Resultados ALNS.

	BKS	$S*$	S	S_0	$gap(\%)$	σ	T
Small	4,92	4,92	4,93	5,32	0,00	0,02	0,12
Grid	11,56	12,94	14,35	16,33	7,54	0,92	14,06
HB	311,55	314,03	318,57	358,86	2,40	3,38	202,71

O ALNS proposto conseguiu alcançar todas as soluções ótimas do conjunto *Small*, o que era esperado, tendo em vista que este conjunto é pequeno e trivial. A solução inicial está a 8% da solução ótima, 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 convergência 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. Tanto o conjunto *Grid* quanto o *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, ainda assim 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 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%.

Na Tabela 2 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 e a coluna $\#OPT$ apresenta a quantidade de soluções ótimas encontradas por cada método.

Tabela 2: Comparação com os métodos estado da arte.

	VFS		SS		ALNS	
	S	$\#OPT$	S	$\#OPT$	S	$\#OPT$
Small	–	–	4,92	84	4,92	84
Grid	12,23	59	13,00	44	12,94	42
HB	314,39	61	315,22	59	314,03	61

Assim como o SS, o ALNS foi capaz de encontrar todas as soluções ótimas do conjunto *Small*, que não foi considerado pelo VFS. No entanto, o ALNS superou as 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 de os testes serem executados em uma arquitetura diferente, a comparação de tempo não pode ser feita de forma justa. Contudo, o ALNS demostra ser competitivo em questão de tempo e qualidade das soluções geradas.

5. Conclusão e Trabalhos Futuros

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 soluções ótimas da literatura e também com os métodos que compõem o estado da arte. O método proposto alcançou uma quantidade significativa de soluções ótimas. Além disso, o ALNS superou o método do estado da arte em um conjunto de instância e demonstrou ser competitivo em qualidade e tempo de execução. Os trabalhos futuros serão concentrados em incluir e aprimorar as vizinhanças utilizadas pelo ALNS, no intuito de obter melhores resultados.

6. Agradecimentos

Esta pesquisa foi apoiada pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pela Fundação de Apoio à Pesquisa do Estado de Minas Gerais e pela Universidade Federal de Ouro Preto.

Referências

Adolphson, D. e Hu, T. C. (1973). Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423.

Andrade, D. V. e Resende, M. G. (2007a). Grasp with evolutionary path-relinking. In *Proc. of Seventh Metaheuristics International Conference (MIC 2007)*.

Andrade, D. V. e Resende, M. G. (2007b). Grasp with path-relinking for network migration scheduling. In *Proceedings of the International Network Optimization Conference*.

Arora, S., Frieze, A., e Kaplan, H. (2002). A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Math Program*, 92(1):1–36.

Bansal, R., Srivastava, K., e Srivastava, S. (2012). A hybrid evolutionary algorithm for the cutwidth minimization problem. In *Evolut Comput (CEC), 2012 IEEE Congress on*, p. 1–8. IEEE.

Botafogo, R. A. (1993). Cluster analysis for hypertext systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, p. 116–125. ACM.

Chung, M.-J., Makedon, F., Sudborough, I. H., e Turner, J. (1985). Polynomial time algorithms for the min cut problem on degree restricted trees. *SIAM Journal on Computing*, 14(1):158–177.

Coudert, D. (2016a). Linear ordering problems on graphs. URL `http://www-sop.inria.fr/members/David.Coudert/code/graph-linear-ordering.shtml`.

Coudert, D. (2016b). *A note on Integer Linear Programming formulations for linear ordering problems on graphs*. PhD thesis, Inria; I3S; Universite Nice Sophia Antipolis; CNRS.

Dıaz, J., Penrose, M. D., Petit, J., e Serna, M. (2001). Approximating layout problems on random geometric graphs. *J. Algorithms*, 39(1):78–116.

Díaz, J., Petit, J., e Serna, M. (2002). A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356.

Duarte, A., Pantrigo, J. J., Pardo, E. G., e Sánchez-Oro, J. (2013). Parallel variable neighbourhood search strategies for the cutwidth minimization problem. *IMA Journal of Management Mathematics*, 27(1):55–73.

Even, S. e Shiloach, Y. (1975). Np-completeness of several arrangement problems. *Department of Computer Science, Israel Institute of Technology, Haifa, Isreal, Tech. Rep*.

Fraire-Huacuja, H. J., López-Locés, M. C., García, N. C., Pecero, J. E., e Rangel, R. P. (2017). Solving the cut width optimization problem with a genetic algorithm approach. In *Nature-Inspired Design of Hybrid Intelligent Systems*, p. 729–738. Springer.

Garey, M. R., Johnson, D. S., e Stockmeyer, L. (1976). Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267.

Gavril, F. (1977). Some np-complete problems on graphs. In *Proc. Conf. on Inform. Sci. and Systems, 1977*, p. 91–95.

Harper, L. H. (1964). Optimal assignments of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, 12(1):131–135.

Kornai, A. e Tuza, Z. (1992). Narrowness, pathwidth, and their application in natural language processing. *Discrete Applied Mathematics*, 36(1):87–92.

Leighton, T. e Rao, S. (1999). Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832.

Lengauer, T. (1982). Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees. *SIAM Journal on Algebraic Discrete Methods*, 3(1):99–113.

Library, M. M. (2007). Matrix market library. URL `http://math.nist.gov/MatrixMarket/index.html`.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

López-Locés, M. C., Castillo-García, N., Huacuja, H. J. F., Bouvry, P., Pecero, J. E., Rangel, R. A. P., Barbosa, J. J., e Valdez, F. (2014). 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*, p. 509–517. Springer.

Makedon, F. e Sudborough, I. H. (1989). On minimizing width in linear layouts. *Discrete Applied Mathematics*, 23(3):243–265.

Makedon, F. S., Papadimitriou, C. H., e Sudborough, I. H. (1985). Topological bandwidth. *SIAM Journal on Algebraic Discrete Methods*, 6(3):418–444.

Martí, R., Campos, V., e Piñana, E. (2008). A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, 186(2):513–528.

Martí, R., Pantrigo, J. J., Duarte, A., e Pardo, E. G. (2013). Branch and bound for the cutwidth minimization problem. *Comput Oper Res*, 40(1):137–149.

Monien, B. e Sudborough, I. H. (1988). Min cut is np-complete for edge weighted trees. *Theoretical Computer Science*, 58(1-3):209–229.

Ohtsuki, T., Mori, H., Kuh, E., Kashiwabara, T., e Fujisawa, T. (1979). One-dimensional logic gate assignment and interval graphs. *IEEE Transactions on Circuits and Systems*, 26(9):675–684.

Optsicom (2017). Optsicom project, university of valencia. URL `http://www.optsicom.es/cutwidth`.

Palubeckis, G. e Rubliauskas, D. (2012). A branch-and-bound algorithm for the minimum cut linear arrangement problem. *Journal of combinatorial optimization*, p. 1–24.

Pantrigo, J. J., Martí, R., Duarte, A., e Pardo, E. G. (2012). Scatter search for the cutwidth minimization problem. *Annals of Operations Research*, p. 1–20.

Pardo, E. G., Mladenović, N., Pantrigo, J. J., e Duarte, A. (2013). Variable formulation search for the cutwidth minimization problem. *Applied Soft Computing*, 13(5):2242–2252.

Pereira, M. A., Coelho, L. C., Lorena, L. A., e De Souza, L. C. (2015). A hybrid method for the probabilistic maximal covering location–allocation problem. *Comp. & Oper. Res.*, 57:51–59.

Raspaud, A., Sỳkora, O., e Vrt'o, I. (1995). Cutwidth of the de bruijn graph. *RAIRO-Theoretical Informatics and Applications*, 29(6):509–514.

Raspaud, A., Schroder, H., Sỳkora, O., Torok, L., e Vrt'o, I. (2009). Antibandwidth and cyclic antibandwidth of meshes and hypercubes. *Discrete Mathematics*, 309(11):3541–3552.

Ropke, S. e Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Santini, A., Ropke, S., e Hvattum, L. M. (2016). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics, Submitted*.

Suderman, M. (2004). Pathwidth and layered drawings of trees. *International Journal of Computational Geometry & Applications*, 14(03):203–225.

Thilikos, D., Serna, M., e Bodlaender, H. (2001). A polynomial time algorithm for the cutwidth of bounded degree graphs with small treewidth. *Algorithms—ESA 2001*, p. 380–390.

Yannakakis, M. (1985). A polynomial algorithm for the min-cut linear arrangement of trees. *Journal of the ACM (JACM)*, 32(4):950–988.