Notas em Matemática Aplicada e-ISSN 2236-5915

Volume 75, 2014

Editores

SBMAC **Sociedade Brasileira de Matemática Aplicada e Computacional**

2014

A Sociedade Brasileira de Matemática Aplicada e Computacional - SBMAC publica, desde as primeiras edições do evento, monografias dos cursos que são ministrados nos CNMAC.

Para a comemoração dos 25 anos da SBMAC, que ocorreu durante o XXVI CNMAC em 2003, foi criada a série **Notas em Matemática Aplicada** para publicar as monografias dos minicursos ministrados nos CNMAC, o que permaneceu até o XXXIII CNMAC em 2010.

A partir de 2011, a série passa a publicar, também, livros nas áreas de interesse da SBMAC. Os autores que submeterem textos à série Notas em Matemática Aplicada devem estar cientes de que poderão ser convidados a ministrarem minicursos nos eventos patrocinados pela SBMAC, em especial nos CNMAC, sobre assunto a que se refere o texto.

O livro deve ser preparado em **Latex (compatível com o Miktex versão 2.9), as figuras em eps** e deve ter entre **80 e 150 páginas**. O texto deve ser redigido de forma clara, acompanhado de uma excelente revisão bibliográfica e de **exercícios de verificação de aprendizagem** ao final de cada capítulo.

Veja todos os títulos publicados nesta série na página
http://www.sbmac.org.br/p_notas.php

SBMAC **Sociedade Brasileira de Matemática Aplicada e Computacional**

2014

INTRODUÇÃO A HEURÍSTICAS PARA REDUÇÃO DE LARGURA DE BANDA DE MATRIZES

Sanderson L. Gonzaga de Oliveira
sanderson@dcc.ufla.br

Guilherme Oliveira Chagas
guilherme.chagas@computacao.ufla.br

Departamento de Ciência da Computação
Universidade Federal de Lavras

Sociedade Brasileira de Matemática Aplicada e Computacional

São Carlos - SP, Brasil
2014

Coordenação Editorial: Maria do Socorro Nogueira Rangel

Coordenação Editorial da Série: Fernando Rodrigo Rafaeli

Editora: SBMAC

Capa: Matheus Botossi Trindade

Patrocínio: SBMAC

Catalogação elaborada pela Biblioteca do IBILCE/UNESP
Bibliotecária: Maria Luiza Fernandes Jardim Froner

Gonzaga de Oliveira, Sanderson L.
Introdução a Heurísticas para Redução de Largura de Banda de Matrizes - São Carlos, SP : SBMAC, 2014, 106 p., 21.5 cm - (Notas em Matemática Aplicada; v. 75)

e-ISBN 978-85-8215-061-0

1. Redução de largura de banda 2. Heurísticas 3. Matrizes esparsas I. Gonzaga de Oliveira, Sanderson L. II. Chagas, Guilherme O. III. Título. IV. Série

CDD - 51

A minha esposa, Tatiana,
ao meu pai, Luiz e
à memória de minha mãe, Selma.
Sanderson.
Aos meus amados pais, Zaqueu e Irene.
Guilherme.

Agradecimentos

Agradecemos ao comitê responsável pela série Notas em Matemática Aplicada da SBMAC a oportunidade de divulgarmos nosso trabalho no CNMAC. Agradecemos também aos revisores anônimos as valiosas contribuições.

Ainda, agradecemos os apoios do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Fapemig). Também, agradecemos ao aluno Alexandre de Abreu as contribuições nas descrições das seções 2.6 e 3.5 e ao aluno Júnior Assis Barreto Bernardes a contribuição na descrição da seção 2.4.

Conteúdo

Prefácio

Este livro foi preparado como material de apoio para o minicurso de mesmo título apresentado pelos autores no XXXV Congresso Nacional de Matemática Aplicada e Computacional (CNMAC) realizado em Natal, Rio Grande do Norte, de 8 a 12 de setembro de 2014. O foco principal deste trabalho é em técnicas para a redução do custo computacional na resolução de sistemas de equações lineares.

Heurísticas para as reduções de largura de banda e de *profile* são particularmente importantes na resolução de equações diferenciais parciais por métodos numéricos, como os métodos dos elementos, volumes e diferenças finitas. Pode-se utilizar uma heurística para as reduções de largura de banda e de *profile* para que a resolução do sistema de equações lineares tenha um custo computacional baixo.

Há centenas de heurísticas para as reduções de largura de banda e de *profile*. Neste livro, apresentamos 12 heurísticas que foram testadas para a redução de largura de banda por seus autores.

As heurísticas para a redução de largura de banda são dependentes das instâncias. Ainda, claramente, o tempo de execução da resolução de um sistema de equações lineares depende do método utilizado e, por sua vez, o tempo de execução do método depende das características da matriz de coeficientes do sistema de equações lineares. Com isso, não é uma tarefa trivial afirmar qual heurística é o estado da arte em redução de largura de banda. Apesar disso, é possível que algumas das heurísticas apresentadas neste livro estejam entre as melhores nessa tarefa.

O livro está dividido em cinco capítulos. Naturalmente, no capítulo 1, há uma introdução ao assunto e são apresentadas noções básicas para a compreensão dos demais capítulos, como conceitos elementares em teoria dos grafos. No capítulo 2, são apresentadas heurísticas clássicas para o problema, heurísticas simples e outras heurísticas projetadas de forma que a numeração é realizada em relação a níveis de vértices a partir de um vértice inicial, tal qual são percorridos os vértices de um grafo na busca em largura. No capítulo 3, são apresentadas heurísticas que reduzem bastante a largura de banda; mas por serem projetadas por meta-heurísticas, inerentemente, são mais lentas que, por exemplo, uma heurística simplesmente baseada na busca em largura. Em suma, apresentamos uma coleção de heurísticas projetadas por técnicas variadas. Supomos que o leitor poderá obter uma introdução adequada ao assunto com essas 12 heurísticas.

Entre as heurísticas apresentadas, há heurísticas que são dependentes do vértice inicial da numeração. Com isso, no capítulo 4, apresentamos algoritmos para se determinar um vértice apropriado para se iniciar a numeração dos vértices. Finalmente, no capítulo 5, são apresentadas simulações com a heurística Cuthill-McKee reverso e com uma variação dessa heurística. Essas formas de redução de largura de banda foram testadas em um projeto computacional para a resolução de equações diferenciais parciais por discretizações por volumes finitos com diagrama de Voronoi e refinamento de Delaunay, nomeadamente, pelos refinamentos de Ruppert e *off-*

centers. Em particular, os sistemas de equações lineares são resolvidos ao se utilizar o método dos gradientes conjugados.

Claramente, o livro não esgota o assunto e, como indicado no título, é apenas uma introdução. Desde já, agradecemos as sugestões para a melhoria do texto que venham a ser encaminhadas para sanderson@dcc.ufla.br.

Lavras, 19 de julho de 2014.

Sanderson L. Gonzaga de Oliveira e Guilherme Oliveira Chagas.

Capítulo 1

Redução de largura de banda de matrizes

1.1 Introdução

A resolução de sistemas de equações lineares do tipo $Ax = b$, em que A é uma matriz esparsa, é central em diversas simulações em ciência e engenharia e é, geralmente, a parte que requer o maior custo computacional na simulação. Como explica Benzi [3], a principal fonte de problemas com matrizes de grande porte é oriunda da discretização (e linearização) de equações diferenciais parciais (EDPs) elípticas ou parabólicas. Por exemplo, a discretização de um modelo matemático pode levar a um sistema grande de equações não lineares. Métodos de linearização reduzem isso a uma série de sistemas de equações lineares de grande porte. Alguns dos métodos numéricos mais populares para a resolução de problemas relacionados a fenômenos físicos modelados por equações diferenciais parciais são os métodos dos elementos finitos, das diferenças finitas e dos volumes finitos. Sistemas de equações lineares são gerados nas aplicações desses métodos. Por outro lado, sistemas algébricos de grande porte também são oriundos de aplicações não modeladas por EDPs, como o projeto e a análise de circuitos integrados, redes de sistemas de potência, processos em engenharia química e modelos econômicos. Benzi [3] apresenta uma boa descrição sobre as principais áreas com problemas em que sistemas algébricos precisam ser resolvidos. Por exemplo, Kaveh [42, p. 221] explica que, na mecânica das estruturas, 30% a 50% do custo computacional pode ser relacionado à resolução de sistemas de equações lineares, para problemas encontrados na prática; e isso pode chegar a 80% em problemas não lineares de otimização de estruturas. Por serem de grande porte, necessitam-se de muita memória e de alto custo de processamento para armazenar e resolver esses sistemas de equações lineares.

Descreve-se sobre a importância das reduções de largura de banda e de *profile* da matriz A na redução do custo computacional na resolução de sistemas de equações lineares nas seções 1.2 e 1.3. Por causa da importância desse tema, foi proposta uma grande quantidade de heurísticas para as reduções de largura de banda, de *profile* e de *frontwidth* (ou *wavefront*). Esses problemas são relacionados, mas independentes. Em particular, *frontwidth* é associado com *Frontal solution techniques* [38], que são resolutores de sistemas de equações lineares baseados na eliminação gaussiana. Principalmente na década de 1970, os sistemas computacionais tinham pouca capacidade de armazenamento. Com a necessidade de precisão atual, siste-

mas computacionais comuns atuais também têm pouca capacidade de armazenamento para se resolver sistemas de grande porte. Nesse tipo de resolutor, realiza-se a fatoração LU ou a decomposição de Cholesky somente em uma "parte" do sistema por vez. Essa "parte" é chamada de *frontwidth* ou *wavefront*. O *frontwidth* $f_i(A)$ são as equações ativas na iteração da técnica frontal para a solução de sistemas de equações lineares. De forma mais técnica, considere uma matriz $A = [a_{ij}]$ simétrica. No i-ésimo passo da fatoração de A, a linha k está ativa se $k \geq i$ para todo $l \leq i$, tal que $a_{kl} \neq 0$. E particular, o *frontwidth* máximo da matriz A é o maior $f_i(A)$.

Heurísticas para renumerar os vértices do grafo têm sido desenvolvidas especificamente para a redução de largura de banda, *profile* ou *frontwidth*. O estudo foi orientado pelo interesse dos pesquisadores no método de resolução do sistema de equações lineares. Em particular, a redução de *profile* foi, por muitas vezes, estudada em conjunto com a redução de largura de banda ou de *frontwidth*. Apesar de estudos específicos em apenas um desses casos, muitas das heurísticas propostas são efetivas, muitas vezes, na redução de todas essas características da matriz A. Entretanto, pode ser que, com a redução da largura de banda, não haja redução do *profile*. O inverso é verdadeiro. Com a redução da largura de banda, ocorre a redução do *frontwidth* máximo; entretanto, pode ser que não haja redução da largura de banda ou do *frontwidth* médios. Com a redução de *profile*, supõe-se que haja redução da largura de banda e do *frontwidth* médios. Como nosso foco é na resolução de sistemas de equações lineares com matrizes simétricas e positivas-definidas, mostramos uma introdução a heurísticas que foram estudadas para as reduções de largura de banda e de *profile*. As heurísticas mostradas nos capítulos 2 e 3 formam uma boa amostra da grande quantidade de heurísticas aplicadas nesses problemas.

Este texto está organizado da seguinte forma. Comenta-se sobre resolução de sistema de equações lineares na seção 1.2. Abordam-se, na seção 1.3, as reordenações de linhas e de colunas de matrizes. Os principais conceitos para o entendimento deste texto, como largura de banda e *profile*, são mostrados na seção 1.4. Na seção 1.5, são mostradas algumas métricas alternativas para as reduções de largura de banda e de *profile*. Apresentam-se heurísticas para as reduções de largura de banda e de *profile* de matrizes nos capítulos 2 e 3. Para isso, considere as definições apresentadas na seção 1.4. No capítulo 2, são apresentadas heurísticas baseadas em níveis de vértices. Apresentam-se, no capítulo 3, heurísticas baseadas em meta-heurísticas. No capítulo 4, são mostrados algoritmos para se encontrar um vértice *pseudo-periférico* (ou vértice com pseudo-diâmetro), que é o vértice inicial para algumas heurísticas para a redução de largura de banda. Os pseudo-códigos estão mostrados de uma forma a se manter a simplicidade na apresentação. Claramente, esquemas e estruturas sofisticadas podem ser utilizadas em uma fase de implementação. No capítulo 5, apresentam-se resultados de implementações da heurística Cuthill-McKee reverso [22] como exemplo de aplicação de uma heurística para as reduções de largura de banda e de *profile*.

1.2 Observações sobre resolução de sistemas de equações lineares

Métodos diretos ou iterativos são utilizados para a resolução de sistemas de equações lineares. Exemplos de métodos diretos são as decomposições LU ou de Choleski. Em geral, os métodos diretos são baseados na eliminação gaussiana. Segundo Tarjan [82], a eliminação gaussiana necessita de $O(n \cdot \beta^2)$ operações para a resolução de um sistema de equações lineares com n elementos e *largura de banda* β. Se

$\beta \cong n$, então, são realizadas $O(n^3)$ operações. Ainda, a eliminação gaussiana requer $O(n \cdot \beta)$ espaços na memória ao se utilizar armazenamento baseado em vetores [82]. Define-se largura de banda na seção 1.4, na página 6. Veja, por exemplo, Davis [10], para detalhes sobre métodos diretos para a solução de sistemas de equações lineares. Benzi [3] explica que métodos diretos têm sido tradicionalmente preferidos em análises de estruturas e modelagens de periféricos semicondutores, em grande parte da área de dinâmica de fluidos computacional e em diversas aplicações em que o problema não é modelado por EDPs, como circuitos e redes de sistemas de potência. Entretanto, métodos diretos têm escalabilidade ruim em relação ao tamanho do problema, em termos de contagem de operações e exigências de memória, especialmente em problemas tridimensionais oriundos da discretização de EDPs. Com isso, sistemas de equações lineares com matrizes de grande porte são, muitas vezes, resolvidos por métodos iterativos. Veja, por exemplo, Saad [77], para detalhes sobre métodos iterativos para a resolução de sistemas de equações lineares.

Na resolução de sistemas de equações lineares, métodos iterativos exigem menos memória e, geralmente, exigem menos operações que métodos diretos, mas não são considerados tão confiáveis quanto os métodos diretos. Há aplicações em que métodos iterativos falham e o pré-condicionamento é necessário, apesar de nem sempre necessário, para que haja convergência em um tempo razoável [3]. Benzi [3] ainda explica que a classificação entre métodos diretos e iterativos é uma simplificação que pode não ser satisfatória atualmente. Após diversas ideias e técnicas da área de métodos diretos serem transferidas, na forma de pré-condicionadores, para métodos iterativos, os métodos iterativos passaram a ser cada vez mais confiáveis. Para a resolução de sistemas de equações lineares com matrizes esparsas de grande porte, um método iterativo eficiente é o método dos gradientes conjugados [49, 37], que pode ser utilizado se a matriz do sistema de equações lineares é simétrica e positiva-definida.

No método dos elementos finitos, ao se utilizar funções bases com suporte compacto, pode-se fazer com que a largura de banda da matriz seja correspondente à ordem do polinômio sendo empregado. Por sua vez, no método dos volumes finitos, a largura de banda da matriz correspondente, em geral, depende de como a malha é percorrida para se montar o sistema de equações lineares. Há formas de se percorrer os volumes de controle em tempo linear ao se utilizar uma curva de preenchimento de espaço, por exemplo, como mostrado por Gonzaga de Oliveira e Kischinhevsky [28, 30]. Entretanto, essa forma de se percorrer os vértices da malha pode gerar um sistema de equações lineares com matriz com largura de banda grande. É necessário encontrar meios adequados para se montar o sistema de equações lineares para que seja resolvido com custo computacional baixo. A forma de montagem do sistema de equações lineares pode deixar de ser uma fonte de preocupação com a utilização de uma heurística para as reduções de largura de banda e de *profile* da matriz de coeficientes. Na maioria dos casos, as reduções de largura de banda e de *profile* afetam a taxa de convergência de métodos baseados no subespaço de Krylov [48], como é o caso do método dos gradientes conjugados [49, 37]. Ainda, o reordenamento das linhas e das colunas de matrizes é um dos itens mais importantes em implementações paralelas de resoluções diretas ou iterativas [77, p. 72].

1.3 Reordenações de linhas e de colunas de matrizes

As heurísticas para as reduções de largura de banda e de *profile* realizam permutações de linhas e colunas das matrizes, deixando-as com uma estrutura compacta e com coeficientes não nulos próximos à diagonal principal. Para uma sistema de equações lineares $Ax = b$, essas heurísticas produzem uma matriz de permutação P, tal que PAP^{T} tem uma largura de banda pequena. O sistema de equações lineares permutado $PAP^{\mathrm{T}}(P\overline{x}) = P\bar{b}$ continua esparso, simétrico e positivo definido. As possíveis vantagens de resolver o sistema de equações lineares permutado é que a solução pode exigir um número menor de operações aritméticas e/ou exigência de armazenamento do que o sistema original [58]. Do ponto de vista da teoria dos grafos, pode-se considerar uma permutação de linhas e colunas em uma matriz simétrica como equivalente a se renumerar os vértices do grafo correspondente sem que as adjacências sejam alteradas.

Como descrito, busca-se baixo custo computacional na resolução de sistemas de equações lineares e no armazenamento desses sistemas com heurísticas para as reduções de largura de banda e de *profile* de matrizes. Em relação a armazenamento, muitas vezes não é prático armazenar todas as subdiagonais da banda de uma matriz porque, geralmente, pode ocorrer que poucas linhas da banda tenham largura de banda grande. Nesses casos, pode ser melhor utilizar o esquema de armazenamento por banda variável ou por *envelope* [39]. Define-se envelope na seção 1.4, na página 6. Uma variação do esquema de Jennings [39] é obtida ao se armazenar a transposta do envelope da matriz triangular inferior. Esse esquema é chamado de *skyline* [18]. Ocorrem, ainda, casos em que mesmo o envelope tem um número grande de zeros. Nesses casos, um esquema de armazenamento bastante utilizado é, para cada linha, armazenam-se os elementos não nulos em uma lista encadeada e também há outra lista para se ligar as listas de cada linha.

Ao se trocar as linhas da matriz de um sistema de equações lineares, altera-se a ordem em que as equações são escritas; ao se trocar as colunas, as incógnitas são renumeradas ou reordenadas. Duff e Meurant [15] compararam 17 ordenações de incógnitas de sistemas de equações lineares oriundas de discretizações de diferenças finitas para a convergência do método dos gradientes conjugados. Eles concluíram que é uma condição suficiente que ordenações que são "locais" fornecem os melhores resultados. Ordenações que são "locais" são aquelas em que vértices (ou incógnitas no sistema original) vizinhos na malha têm posições na ordenação que não são muito distantes entre si. Esse foi o caso das ordenações produzidas pelas heurísticas de Cuthill e McKee [9] e Cuthill-McKee reverso (CMr) [22], que são descritas na seção 2.2, na página 14. Essa é uma condição suficiente, mas não necessária, já que há ordenações que não são locais, mas fornecem resultados bons como, por exemplo, a ordenação em espiral [14]. Os autores deixaram um problema não trivial: dada uma matriz, encontre a ordenação mais local. Isso é bastante similar ao problema de *minimização de largura de banda*.

A minimização de largura de banda de uma matriz simétrica A é aproximar, ao máximo possível, todos os coeficientes não nulos da diagonal principal de A. Isso significa que minimizar a largura de banda é encontrar a menor largura de banda de uma matriz simétrica. Esse é um problema NP-difícil, conforme foi demonstrado por Papadimitriou [70]. O autor mostrou que o problema de se determinar se os vértices de um grafo têm uma ordenação que produz uma largura de banda de tamanho B ou menor é NP-Completo. Garey et al. [20] provaram que o problema

é NP-completo mesmo para árvores com grau máximo 3. Petit [71] mostrou que o problema de se encontrar o *profile* mínimo é equivalente ao problema da soma de cortes em grafos. Esse problema foi mostrado ser NP-Difícil em Díaz et al. [11], Lin e Yuan [56, 57] e Golovach [27]. Petit [71] revisou resultados da resolução de vários problemas em grafos: em minimização de largura de banda, pesquisadores identificaram muitas classes particulares de grafos em que esse problema permanece NP-Difícil.

Apesar de os estudos de Duff e Meurant [15] terem sido somente em discretizações por diferenças finitas, conclusões mais gerais são válidas. Por exemplo, ao se utilizar o método dos gradientes conjugados para resolver sistemas algébricos oriundos do método dos elementos finitos, os autores sugerem que deve ser benéfico ordenar os vértices pela heurística CMr. Dutto [16] também considerou padrões esparsos baseados em ordenações em matrizes jacobianas oriundas da discretização de equações de Navier-Stokes em malhas irregulares e os resultados coincidiram com os de Duff e Meurant [15], indicando que a ordenação pela heurística CMr ou pela heurística de Gibbs [24] são boas escolhas.

O problema de minimização de largura de banda de matrizes surgiu, possivelmente, na década de 1950. Engenheiros de estruturas analisavam a resolução computacional de sistemas de equações lineares com matrizes de grande porte e já tinham a preocupação com a largura de banda das matrizes (por exemplo, veja Livesley [61]).

Os métodos que realizam a redução de largura de banda podem ser divididos em métodos exatos e heurísticas. Como explica Skiena [78, p. 399], métodos por força bruta encontram a menor largura de banda de matrizes por *backtracking* no conjunto de $n!$ permutações possíveis de vértices. Exemplos de métodos exatos podem ser obtidos em Harary [36], Gurari e Sudborough [32], Corso e Manzini [7] e Martí, Campos e Piñana [63]. Ainda, Caprara e Salazar-González [5] desenvolveram e melhoraram métodos baseados em programação inteira.

Neste texto, abordam-se *heurísticas* para as reduções de largura de banda e de *profile* de matrizes. O foco neste trabalho é em matrizes simétricas. Entretanto, há heurísticas (por exemplo, TS-*band* [64], GRASP$_{PR}$ [72], GA-HC [54], NS-HC [54], PSO-HC [52], NC-HC [55], FNC-HC [55], SA-δ [74] e VNS-*band* [66]) que foram testadas também para matrizes $A = [a_{ij}]$ assimétricas. Para heurísticas para a renumeração de matrizes retangulares, veja a seção 7.13 de Kaveh [42, p. 260-264]. Para sub-ordenações de blocos de vértices, veja a seção 7.14 de Kaveh [42, p. 265-267]. Isso pode ser útil, por exemplo, para particionamento de grafos em resoluções paralelas de sistemas de equações lineares.

Como é claramente impraticável verificar todas as $n!$ sequências possíveis associadas com uma matriz de ordem n, desde a década de 1960, foram propostas, provavelmente, centenas de heurísticas aplicadas nos problemas de redução de largura de banda, de *profile* ou de *wavefront*. Everstine [17] referenciou 49 heurísticas para as reduções de largura de banda, de *profile* ou de *wavefront*. Gibbs [25] afirmou que de forma alguma a lista de Everstine [17] foi exaustiva. Gibbs [25] supôs que existiriam de 50 a 100 heurísticas para as reduções de largura de banda, *profile* e de *wavefront* até 1980.

Com a grande quantidade de heurísticas para as reduções de banda e de *profile*, têm-se evidências da importância desses problemas. Inclusive, no *MATLAB* [83], constam versões das heurísticas CMr de George [22] e *Spectral* de Barnard, Photen e Simon [2]. Na *Netlib* (*http://www.netlib.org*), encontram-se implementações em *Fortran* da heurística GPS de Gibbs, Poole e Stockmeyer [26] em *http://www.netlib.org/toms/508* [8] e da heurística Gibbs-King de Gibbs [24] em *http://www.*-

netlib.org/toms/509 [24]. Também, em *http://www.netlib.org/toms/582* [50], há implementações das heurísticas GPS e Gibbs-King.

Como exemplo, na figura 1.1, tem-se a representação de um grafo por meio de uma matriz de adjacências. Cada linha da matriz A corresponde a um vértice no grafo da figura 1.1, ou seja, para $1 \leq i \leq 10$, a linha i da matriz corresponde ao vértice i do grafo.

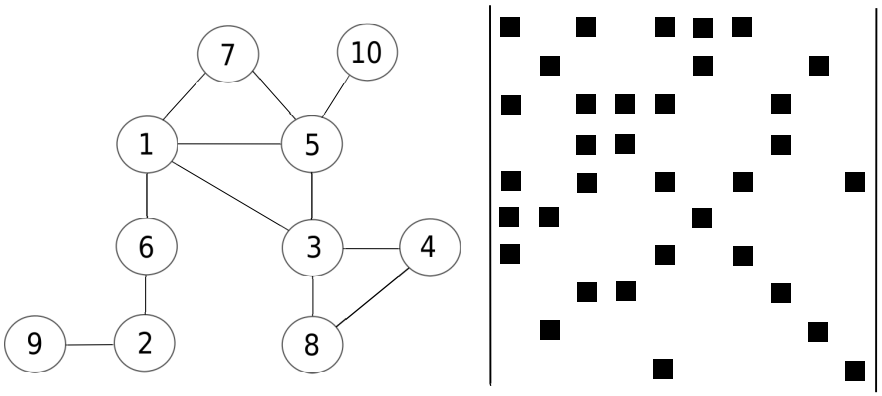

Figura 1.1: Representação do grafo por uma matriz de adjacências.

Para o grafo da figura 1.2, utilizou-se uma ordem diferente da anterior para numerar os vértices do grafo e obteve-se uma representação matricial também diferente. Nota-se que a disposição dos coeficientes não nulos na representação matricial do grafo depende da ordem em que se numeram os vértices do grafo para a montagem da matriz. É da ordem dessa numeração de que trata este texto. Para isso, há heurísticas que definem essa ordem e o vértice inicial da numeração.

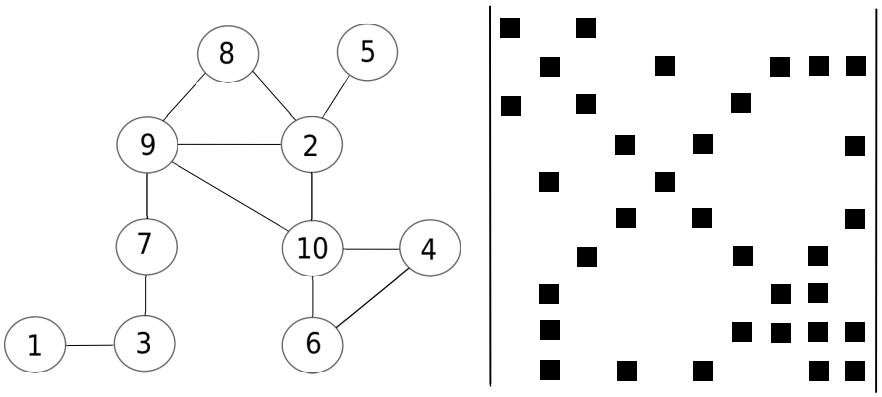

Figura 1.2: Grafo da figura 1.1 com a numeração dos vértices alterada.

1.4 Conceitos básicos

Os sistemas de equações lineares considerados aqui são da forma $Ax = b$, em que A é uma matriz $n \times n$ esparsa, simétrica e positiva-definida. Uma matriz A é

positiva-definida se, para qualquer vetor x, tem-se, $x^{\mathrm{T}}Ax > 0$. A matriz $A = [a_{ij}]$ é positiva-definida se e, somente se, todos os autovalores são positivos.

Adiante, considere um grafo $G = (V, E)$ não orientado e não ponderado, constituído por um conjunto finito de vértices $V = \{v_1, v_2, v_3, \cdots, v_n\}$ e um conjunto finito de arestas $E = \{e_1, e_2, e_3, \cdots, e_m\}$ representadas por pares não ordenados de vértices $\{v_i, v_j\}$. Tem-se as seguintes definições [29, p. 215-219]:

- dois vértices são adjacentes quando existe uma aresta entre eles e $Adj(G, v) = \{u \in V : \{v, u\} \in E\}$ é o conjunto de vértices adjacentes ao vértice v;
- o grau de um vértice v é o número de vértices adjacentes a v, ou seja, $Grau(G, v) = |\{u \in V : \{v, u\} \in E\}|$;
- caminho é uma sequência de vértices tal que, de cada um dos vértices parte uma aresta para o vértice seguinte, até o último vértice da sequência;
- o tamanho de um caminho é o seu número de arestas;
- em um grafo conexo, existe um caminho entre qualquer par de vértices;
- árvore é um grafo conexo acíclico;
- componentes de um grafo $G = (V, E)$ são os subgrafos conexos maximais de $G = (V, E)$, ou seja, são os subgrafos conexos que não estão estritamente contidos em outros subgrafos conexos.

Definição 1 (largura de banda)**.** *Seja A uma matriz simétrica $n \times n$, com coeficientes a_{ij} e $1 \leq i, j \leq n$. A largura de banda da i-ésima linha é $\beta_i(A) = i - \min_{1 \leq j < i} (j \mid a_{ij} \neq 0)$. A largura de banda $\beta(A)$ é a maior distância de coeficiente não nulo da matriz triangular inferior até a diagonal principal, considerando-se todas as n linhas da matriz, ou seja, $\beta(A) = \max_{1 \leq i \leq n} (\beta_i(A)) = \max_{1 \leq i \leq n, 1 \leq j < i} (i - j \mid a_{ij} \neq 0)$.*

A definição 1 significa que a largura de banda de uma matriz simétrica A é o número de subdiagonais da matriz triangular inferior. Caso seja necessária uma definição para matrizes assimétricas, pode-se utilizar $\beta(A) = \max_{1 \leq i, j \leq n} (|i - j| \mid a_{ij} \neq 0)$ ou $\beta_i(A) = i - \min_{1 \leq j \leq n} (j \mid a_{ij} \neq 0)$. Isso é o mesmo que definir $diam(s(v)) = \max_{\{u,v\} \in E} |s(u) - s(v)|$, em que $s(u)$ é a numeração do vértice u. Quase todas as heurísticas mostradas neste texto foram testadas somente para matrizes simétricas por seus autores. Somente as heurísticas NC-HC [55] e FNC-HC [55], mostradas na seção 3.2, na página 43, e a heurística VNS-*band* [66], mostrada na seção 3.4, na página 51, foram testadas também para matrizes assimétricas.

Definição 2 (banda)**.** *A banda de A é definida como $banda(A) = \{(1 \leq i \leq n)\ (1 \leq j < i)\ (i, j) \mid (0 < i - j \leq \beta(A))\}$.*

A definição 2 significa que a banda é formada pelas diagonais da matriz triangular inferior que contenham coeficientes não nulos. Note que a diagonal principal não compõe a banda. Ainda, note que só se considera a matriz triangular inferior porque A é simétrica.

Definição 3 (envelope)**.** *O envelope de A é definido como $Env(A) = \{(1 \leq i \leq n)\ (1 \leq j < i)\ (i, j) \mid 0 < i - j \leq \beta_i(A)\}$.*

A definição 3 significa que os coeficientes nulos mais à esquerda da i-ésima linha, em β_i, não compõem o envelope.

Definição 4 (*profile*). *O profile de A é definido como $|Env(A)| = \sum_{i=1}^{n} \beta_i(A)$.*

A definição 4 significa que o *profile* é a soma das distâncias, em cada linha, do primeiro coeficiente não nulo até a diagonal principal. O estudo da redução de *profile* pode ser mais interessante do que o estudo da largura de banda. Isso porque pode ser que $\beta_i(A)$ de uma só linha i torne a matriz com largura de banda grande, mas as demais $n-1$ linhas podem ter $\beta_j(A)$ pequeno, com $j \neq i$. Com isso, o *profile* de A pode ser pequeno. Portanto, apesar de que com a redução da largura de banda pode-se reduzir o *profile* de A, a redução de largura de banda *não* é condição suficiente *nem* necessária para a redução do *profile* de A. O inverso é verdadeiro. Com isso, heurísticas foram estudadas para reduções de largura de banda ou de *profile* ou ambas.

Definição 5 (distância). *A distância $d(v,u)$ entre dois vértices v e $u \in V$ é o tamanho do menor caminho entre eles.*

Definição 6 (excentricidade). *A excentricidade $\ell : V \to \mathbb{N}$ de um vértice $v \in V$ é dada por $\ell(v) = \max_{u \in V}(d(v,u))$.*

Definição 7 (diâmetro do grafo). *O diâmetro $\Phi(G) = \max_{v \in V}(\ell(v)) = \max_{u,v \in V}(d(v,u))$ do grafo $G = (V,E)$ é a maior excentricidade encontrada em $G = (V,E)$.*

Definição 8 (vértice periférico). *Um vértice v é considerado periférico se a sua excentricidade é igual ao diâmetro do grafo, ou seja, $\ell(v) = \Phi(G)$.*

Para Gibbs, Poole e Stockmeyer [26], um vértice é considerado *pseudo-periférico* se sua excentricidade for próxima ao diâmetro do grafo.

Definição 9 (estrutura de nível enraizada). *Seja $G = (V,E)$ conexo e simples. Dado um vértice $v \in V$, a estrutura de nível com raiz v e profundidade $\ell(v)$ é o particionamento $\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{\ell(v)}(v)\}$, em que $L_0(v) = \{v\}$ e $L_i(v) = Adjc(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, para $i = 1, 2, 3, \ldots, \ell(v)$ e $Adjc(.)$ retorna os vértices adjacentes aos vértices do argumento.*

Isso significa que uma estrutura de nível enraizada de um vértice de um grafo conexo e simples é um particionamento do conjunto de vértices em classes (ou *níveis*, *levels*) de vértices com as mesmas distâncias para um vértice determinado na raiz da árvore. Note que o número de partições (ou níveis) de $\mathscr{L}(v)$ é $\ell(v)+1$ e $\bigcup_{i=0}^{\ell(v)} L_i(v) = V$. Mostra-se um exemplo de montagem de uma estrutura de nível enraizada na figura 1.3. Para isso, considere o grafo da figura 1.1. Para a estrutura de nível enraizada mostrada na figura 1.3, o vértice inicial 1 foi escolhido arbitrariamente. Note que $\ell(1) = 3$. O particionamento dos vértices em níveis, partindo-se do vértice 1, é $L_0(1) = \{1\}$, $L_1(1) = \{3, 5, 6, 7\}$, $L_2(1) = \{2, 4, 8, 10\}$ e $L_3(1) = \{9\}$.

Definição 10 (largura de nível). *A largura de nível $b(\mathscr{L}(u))$ é o número de vértices do nível com mais vértices, ou seja, $b(\mathscr{L}(u)) = \max_{0 \leq i \leq \ell(u)} |L_i(u)|$.*

Definição 11 (estrutura de nível). *Seja $G = (V,E)$ conexo e simples. A estrutura de nível é o particionamento $\mathscr{K}(v, \cdots) = \{K_0(v, \cdots), K_1(v, \cdots), \cdots, K_{\ell(v)}(v, \cdots)\}$,*

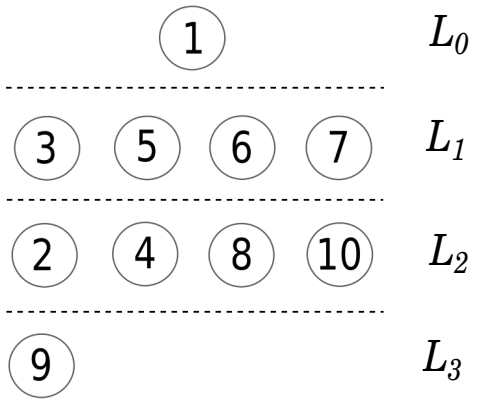

Figura 1.3: Estrutura de nível enraizada $\mathscr{L}(1)$ do grafo da figura 1.1.

em que $v \in K_0(v, \cdots)$ *e* $K_i(v, \cdots) = Adjc(K_{i-1}(v, \cdots)) - \bigcup_{j=0}^{i-1} K_j(v, \cdots)$*, para* $i = 1, 2, 3, \ldots, \ell(v)$ *e* $Adjc(.)$ *retorna os vértices adjacentes aos vértices do argumento.*

Essa estrutura é uma estrutura de nível *não* enraizada, ou seja, pode ter mais de um vértice na raiz. Note que consta, pelo menos, um vértice $v \in V$ no primeiro nível da estrutura, ou seja, no nível $K_0(v, \cdots)$.

Segundo Cuthill e Mckee [9], se a renumeração dos vértices de $G = (V, E)$ é realizada nível a nível de $\mathscr{K}(v, \cdots)$, então, $\beta(A) \leq 2b(\mathscr{K}(v, \cdots)) - 1$. Ainda, se a estrutura de nível é enraizada, a largura de banda de A não pode ser menor que a largura de nível da estrutura de nível enraizada, ou seja, $b(\mathscr{L}(v)) \leq \beta(A) \leq 2b(\mathscr{L}(v)) - 1$.

Definição 12 (aresta crítica)**.** *Uma solução* S *é o conjunto das numerações dos vértices, ou seja,* $S = \{s(1), s(2), \cdots, s(|V|)\}$*, em que* $s(i)$ *é a numeração do vértice* i *na solução* S*. A matriz* A_S *é a matriz obtida pela numeração* S *do grafo* $G = (V, E)$*. Uma aresta* $\{u, v\} \in E$ *é crítica se* $|s(u) - s(v)| = \beta(A_S)$*.*

Definição 13 (vértice crítico)**.** *Um vértice* $v \in V$ *é um vértice crítico de* $G = (V, E)$ *se* $(\exists u \in V)$ *tal que* $\{u, v\}$ *é uma* aresta crítica *de* $G = (V, E)$*.*

1.5 Métricas alternativas para a redução de largura de banda

Em heurísticas para a redução de largura de banda, alguns autores utilizam métricas (funções objetivo) diferentes da largura de banda β para verificar a qualidade da solução corrente. Rodriguez-Tello e Torres-Jimenez [75] afirmam que a métrica β quase não proporciona informações detalhadas em relação à qualidade da solução. Por definição, β de uma matriz simétrica A é a maior distância de coeficiente não nulo da matriz triangular inferior até a diagonal principal, ou a maior diferença entre as numerações de vértices adjacentes. Portanto, com a métrica β, uma melhora só é constatada se é trocada a numeração de um vértice crítico.

Considere que $\beta(A_S)$ é a largura de banda da matriz simétrica A obtida pela numeração S. A numeração (ou solução) S é o conjunto das numerações dos vértices,

ou seja, $S = \{s(1), s(2), \cdots, s(|V|)\}$, em que $s(i)$ é a numeração do vértice $i \in V$ da solução S. A seguir, apresentam-se algumas métricas alternativas para a redução de largura de banda.

- Na métrica de Torres-Jimenez e Rodriguez-Tello [84], denominada γ, leva-se em consideração todas as arestas do grafo. Define-se γ por

$$\gamma = \sum_{i,j|\{i,j\}\in E} P(|V|, |i-j|),$$

em que

$$P(N,k) = \begin{cases} 1, & \text{se } k = 0 \\ (N+1) \cdot \prod_{j=2}^{k} (2N - 2j + 3), & \text{caso contrário} \end{cases}.$$

Com γ, pequenas melhoras em relação à solução corrente são notadas. Entretanto, segundo Rodriguez-Tello e Torres-Jimenez [75], essa métrica só é eficiente em grafos com até 150 vértices.

- Na métrica de Rodriguez-Tello e Torres-Jimenez [75], identificam-se pequenas melhoras em relação à solução corrente. Essa métrica é definida por

$$\delta(S) = \beta(A_S) + \sum_{i=0}^{\beta(A_S)} \left(\frac{d_i}{\prod_{j=i}^{\beta(A_S)} (|V| + (\beta(A_S) - j) + 1)} \right),$$

em que d_i é o número de diferenças, com valor i, entre as numerações dos vértices adjacentes. Rodriguez-Tello e Torres-Jimenez [75] realizaram testes com *Simulated Anneling*, utilizando as métricas β e δ, e as heurísticas foram denominadas de SA-β e de SA-δ, respectivamente. Rodriguez-Tello e Torres-Jimenez [75] compararam a heurística SA-δ com as heurísticas: GPS de Gibbs, Poole e Stockemeyer [26], de Dueck e Jeffs [13], TS (*tabu search*) de Martí et al. [64], GRASP$_{PR}$ (*Greedy Randomized Adaptive Search Procedure with Path Relinking*) de Piñana et al. [72] e GA-HC (*Genetic Algorithm with Hill Climbing*) de Lim, Rodrigues e Xiao [53]. Com a heurística SA-δ, os autores obtiveram resultados melhores, em quase todos os testes, em relação aos resultados dessas cinco heurísticas. A heurística de Rodriguez-Tello, Kao e Torres-Jimenez [74] utiliza a métrica δ. Por outro lado, essa heurística foi superada na redução de largura de banda pela heurística VNS-*band* [66], mostrada na seção 3.4, na página 51.

- Mafteiu-Scai [62], utilizaram a métrica denominada *mbw*, com foco em paralelização. A vantagem de se utilizar essa métrica é que se promove uma distribuição uniforme dos coeficientes não nulos ao redor da diagonal principal. Utiliza-se $mbw(A_S) = \frac{1}{m} \cdot \sum_{a_{ij} \neq 0} |i-j|$, em que m é o número de coeficientes não nulos da matriz A e com $i, j = 1, 2, \cdots, n$. A heurística de Mafteiu-Scai [62] obteve resultados comparáveis aos resultados da heurística Cuthill-McKee reverso, mostrada na seção 2.2, na página 14.

1.6 Exercícios

1. Qual é a largura de banda máxima do grafo mostrado na figura 1.1?
2. Qual é a largura de banda da matriz mostrada na figura 1.1?
3. Qual é a aresta crítica do grafo mostrado na figura 1.1?
4. Qual é a largura de banda β_i de cada linha da matriz mostrada na figura 1.1?
5. Qual é o *profile* da matriz mostrada na figura 1.1?
6. Mostre as estruturas de nível dos vértices 2 a 10 do grafo da figura 1.1.
7. Qual a largura de nível $b(\mathscr{L}(v))$, em que v é cada vértice do grafo da figura 1.1.
8. Qual é a excentricidade de cada vértice do grafo da figura 1.1?
9. Qual é o diâmetro do grafo da figura 1.1?
10. Qual é a largura de banda da matriz mostrada na figura 1.2?
11. Qual é a largura de banda β_i de cada linha da matriz mostrada na figura 1.2?
12. Qual é o *profile* da matriz mostrada na figura 1.2?

Capítulo 2

Heurísticas baseadas em níveis de vértices

2.1 Introdução

Neste capítulo, são apresentadas heurísticas que utilizam a estrutura de nível ou variações dessa estrutura para as reduções de largura de banda e de *profile*. Algumas heurísticas utilizam abordagens simples para a renumeração dos vértices do grafo, como, por exemplo, a busca em largura. Esse é o caso das heurísticas Cuthill-McKee [9] e da heurística de Boutora et al. [4]. As heurísticas Cuthill-McKee (CM) [9], Cuthill-McKee reverso (CMr) [22] e GPS [26] são apresentadas por serem as heurísticas clássicas para a redução de largura de banda. A heurística de Snay [80] é apresentada por ser um exemplo simples de heurística estudada para a redução de *profile*. A heurística quatro-etapas [40] é descrita por ser simples e por utilizar um exemplo de estrutura similar à estrutura de nível, a estrutura *Shortest Route Tree* [42]. A heurística de Boutora et al. [4] é descrita por ser simples e por ser uma heurística baseada na busca em largura. A heurística *Generalized* GPS (GGPS) [88] é apresentada porque é um exemplo de heurística recente e baseada na heurística GPS [26].

A heurística CM foi proposta para a redução de largura de banda e a heurística CMr [22] produz matrizes com as mesmas larguras de banda da heurística CM, mas com *profiles* pelo menos tão bons quanto a heurística CM. Depois da publicação da heurística CMr, projetistas de diversas heurísticas passaram a realizar a renumeração que ao final é *invertida*: isso faz com que a largura de banda seja mantida, mas os *profiles* podem ser menores que na renumeração sem a inversão. As heurísticas GPS [26], a heurística de Boutora et al. [4] e a heurística GGPS [88] foram propostas para as reduções de largura de banda e de *profile* de matrizes simétricas. A heurística quatro-etapas [40] foi proposta para a redução de largura de banda.

Este capítulo está dividido como a seguir. Na seção 2.2, são apresentadas as heurísticas CM [9] e CMr [22]. Na seção 2.3, aborda-se a heurística GPS [26]. Mostra-se a heurística de Snay [80] na seção 2.4. A heurística quatro-etapas [40] é apresentada na seção 2.5. A heurística de Boutora et al. [4] é descrita na seção 2.6. Apresenta-se a heurística GGPS [88] na seção 2.7.

2.2 Heurísticas Cuthill-McKee e Cuthill-McKee reverso

Com a heurística de Cuthill e McKee (CM) [9], reduz-se a largura de banda de uma matriz simétrica por meio da reordenação dos vértices do grafo que correspondem à matriz. Essa heurística é similar à busca em largura, em que todos os vértices do grafo são percorridos. A ordem de visitação dos vértices é o que difere a heurística CM da busca em largura. Na heurística CM, a visitação dos vértices se dá de forma crescente ao grau dos vértices, isto é, dado um vértice inicial, seus vértices adjacentes serão visitados em ordem crescente de grau e isso ocorrerá para os vértices adjacentes desses adjacentes e assim, sucessivamente, até que todos os vértices sejam percorridos. A visitação dos vértices adjacentes em ordem crescente de grau proporciona uma boa configuração da matriz correspondente em relação a sua largura de banda, pois os vértices de maior grau ficam posicionados, no possível, nas linhas centrais da matriz.

Cuthill e McKee [9] constataram que a escolha do vértice inicial interferia na qualidade da solução: na largura de banda e no *profile* da matriz resultante. Então, propuseram que a heurística começasse pelo vértice de grau mínimo do grafo. Nas heurísticas CM e CMr originais, eram geradas as estruturas de nível de todos os vértices de grau mínimo. Em seguida, eram selecionadas as estruturas de nível que apresentassem a menor largura de nível. A renumeração era realizada com base em todas essas estruturas de nível e era escolhida a renumeração que gerasse a menor largura de banda. Embora apresente bons resultados, isso não garante que a largura de banda ou o *profile* sejam menores que uma solução em que o vértice inicial seja o vértice de maior excentricidade.

Como há um custo computacional grande para se encontrar o vértice de maior excentricidade do grafo, ou seja, um vértice com excentricidade igual ao diâmetro do grafo, a partir da publicação da heurística GPS [26], os pesquisadores começaram a utilizar um vértice pseudo-periférico em vez de se testar as renumerações com vértices iniciais de menor grau. Como exemplos, em Liu e Sherman [59, 60] e Liu [58], em que é mostrada a complexidade da heurística CM, omite-se o passo de se escolher o vértice inicial.

A seguir, na subseção 2.2.1, é apresentada a heurística CM. Apresenta-se a heurística Cuthill-McKee reverso (CMr) na subseção 2.2.2. Na subseção 2.2.3, é apresentada a análise de complexidade dessas heurísticas. Na subseção 2.2.4, um exemplo é mostrado. No capítulo 5, são descritas implementações da heurística CMr, como exemplo de aplicação de heurística para as reduções de largura de banda e de *profile* de matrizes.

2.2.1 Pseudo-código para a heurística Cuthill-McKee

A heurística CM é mostrada no algoritmo 1 [58]. O algoritmo recebe um grafo $G = (V, E)$ conexo, em que V é o conjunto de vértices e E é o conjunto de arestas. Outro parâmetro do algoritmo é o vértice inicial $v \in V$. Na linha 2, inicializa-se $s(1)$ com o vértice inicial, em que S é a sequência dos vértices a ser retornada. Nas linhas 3 e 4, inicializam-se as variáveis i e j, respectivamente. A variável i indica a última entrada ocupada de S e a variável j é utilizada para percorrer os vértices do grafo. Em particular, mostra-se no algoritmo 1 como percorrer os vértices do grafo com uma variável, j, em vez de se utilizar uma fila, como ocorre na busca em largura. Na linha 6, percorrem-se os vértices adjacentes a $s(j)$ ainda não renumerados.

Algoritmo 1: Cuthill-McKee.

Entrada: grafo $G = (V, E)$ conexo; vértice inicial $v \in V$;
Saída: renumeração S com $|V|$ entradas $s(1), s(2), \cdots, s(|V|)$;

```
1  início
2      s(1) ← v;
3      i ← 1;
4      j ← 1;
5      enquanto ( i < |V| ) faça
6          para cada ( vértice w ∈ Adj(G, s(j)) − {s(1), ··· , s(i)}, em ordem
           crescente de grau ) faça
7              i ← i + 1;
8              s(i) ← w;
9          fim-para-cada;
10         j ← j + 1;
11     fim-enquanto;
12     retorna S;
13 fim.
```

Na estrutura de repetição das linhas 5 a 11, enquanto i for menor que o número de vértices do grafo, a cada iteração, renumeram-se os vértices adjacentes a $s(j)$, em ordem crescente de grau, exceto aqueles que já foram renumerados. Após cada vértice w adjacente a $s(j)$ ser renumerado, uma referência a w é inserida na i-ésima entrada de S, após incremento de i. A saída do algoritmo é a sequência de vértices em S.

2.2.2 Cuthill-Mckee reverso

George [22] descobriu que renumerar os vértices na ordem inversa em que foram visitados, em vez da renumeração habitual da heurística CM, diminui, frequentemente, o tamanho dos $profiles$ das matrizes resultantes; porém, a largura de banda é mantida. A heurística CMr é a heurística CM, mas com ordenação *inversa*. Liu e Sherman [59, 60] e Liu [58] provaram que a heurística CMr resulta em matrizes com $profiles$, pelo menos, tão bons quanto a heurística CM.

Na versão original da heurística CMr [22], também são selecionados vértices iniciais com grau mínimo do grafo e são geradas as estruturas de nível desses vértices. Em seguida, renumeram-se os vértices pelas estruturas de nível enraizadas que possuírem as menores larguras de nível e escolhe-se a renumeração que gerar a menor largura de banda.

As reduções de largura de banda e de *profile* da matriz resultante, obtidas pelas heurísticas CM e CMr, dependem da escolha do vértice inicial. Essa tarefa rendeu trabalhos, que são abordados no capítulo 4, na página 69.

2.2.3 Análise de complexidade

Suponha que o grafo é representado por listas de adjacências. Para encontrar o vértice de grau mínimo, necessita-se percorrer todos os vértices. Para cada vértice, percorre-se toda a lista de adjacências para se verificar o grau do vértice. De uma forma similar, utilizando-se a busca em largura, ao percorrer a lista de adjacências de cada vértice, pode-se verificar o grau de cada vértice. Dessa forma, o custo para

se encontrar o vértice de grau mínimo é $O(|V| + |E|)$ no grafo $G = (V, E)$. Em utilizando-se uma matriz de adjacências, pode-se percorrer as $|V|^2$ entradas para se encontrar o vértice de grau mínimo.

Conforme demonstrado por Liu [58], a complexidade no pior caso das heurísticas Cuthill-McKee e Cuthill-McKee reverso é dada pelo produto entre o grau m do vértice de maior grau do grafo e o número de arestas $|E|$. Liu [58] obteve a complexidade, no pior caso, considerando que, no passo de ordenação dos vértices na linha 6 do algoritmo 1, fosse utilizado o algoritmo por inserção. Com isso, considere que todas as ordenações na linha 6 exijam, no máximo, $\frac{n^2}{c}$ operações, para n vértices e uma constante c. Esse número de operações é menor que $\frac{1}{c}\sum_{v\in V} Grau(G,v)^2 \leq \frac{m}{c}\sum_{v\in V} Grau(G,v) \leq \frac{2m|E|}{c}$ porque, para um grafo não direcionado, ocorre $\sum_{v\in V} Grau(G,v) = 2|E|$. Com isso, $\frac{2m|E|}{c}$ é o limite superior do número de operações na linha 6. O número de operações da estrutura de repetição na linha 5 é $\Theta(|V|)$. Com isso, a complexidade no pior caso desse algoritmo é $O(m \cdot |E|)$. O limite inferior desse algoritmo, considerando-se um grafo em que o vértice inicial tem ligações com todos os demais vértices e utilizando-se, por exemplo, o *merge sort* no passo da ordenação, é $\Omega(|V| \lg |V|)$.

2.2.4 Exemplo

A seguir, é mostrado um exemplo da execução da heurística Cuthill-McKee. Apresenta-se apenas a heurística Cuthill-McKee, pois a heurística CM e a heurística CMr diferem apenas na ordem da renumeração final. Considere o grafo da figura 2.1 e, ao seu lado, sua representação matricial M. Neste exemplo, o vértice inicial não é o de grau mínimo, como ocorre nas heurísticas CM e CMr.

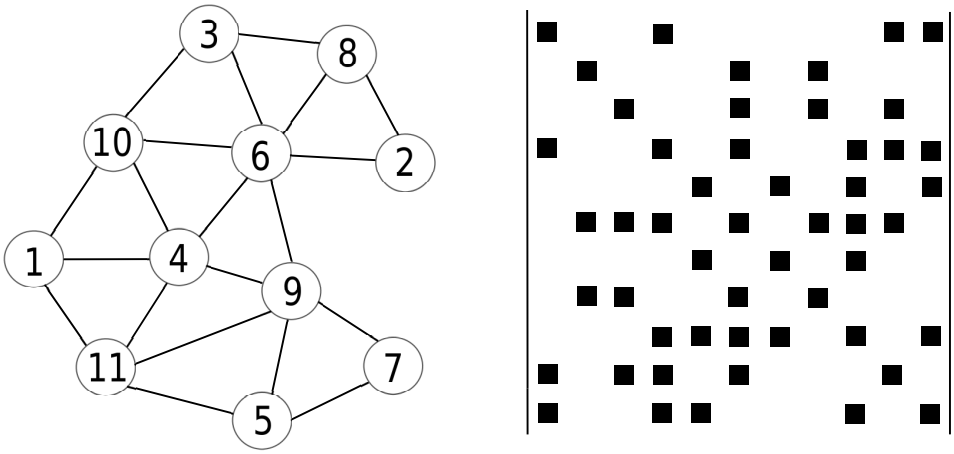

Figura 2.1: Grafo original para um exemplo da heurística CM e sua representação matricial M.

Escolheu-se, arbitrariamente, o vértice 9 como o inicial. Insere-se uma referência a esse vértice na primeira posição da saída S. Em seguida, visitam-se os vértices adjacentes ao vértice 9, referenciado-os, em ordem crescente de grau, em posições da saída S.

Na figura 2.2, mostram-se, em destaque, o vértice 9 e suas arestas incidentes. Na tabela 2.1, mostram-se os vértices adjacentes ao vértice 9, ordenados por seus graus. Os vértices são atribuídos a cada posição de S nessa ordem.

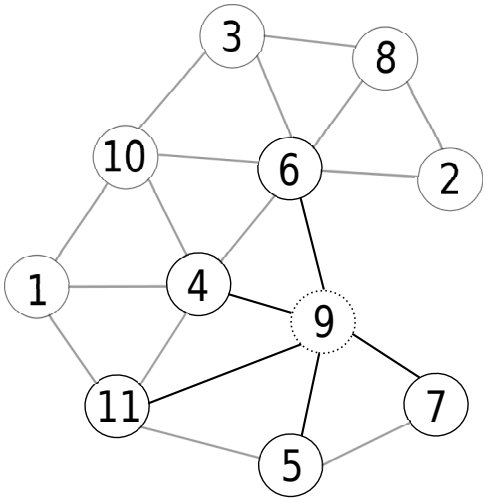

Figura 2.2: Passo inicial da ordenação pela heurística Cuthill-McKee.

Vértice	7	5	11	4	6
Grau	2	3	4	5	6

Tabela 2.1: Vértices adjacentes ao vértice 9 e seus graus.

O próximo vértice a ser processado é o vértice 7 e a segunda entrada em S é uma referência para esse vértice. Não são referenciados novos vértices no vetor porque os vértices adjacentes ao vértice 7 já foram visitados. O mesmo ocorre com o vértice 5. Ao processar o vértice 11, o vértice 1 é referenciado na próxima entrada disponível do vetor S. Isso é realizado até que todos os vértices sejam visitados. Veja a ordem de visitação na figura 2.3.

O resultado da renumeração é mostrado na figura 2.4, juntamente com a matriz M_1 correspondente. Na figura 2.5, representa-se o grafo com a renumeração pela heurística Cuthill-McKee reverso e sua representação matricial M_2. Nesse exemplo, $\beta(M) = 10$, $\beta(M_1) = \beta(M_2) = 5$, $|Env(M)| = 39$, $|Env(M_1)| = 33$ e $|Env(M_2)| = 24$. Na tabela 2.2, esses dados são sumarizados. Note que, com a numeração inversa, as colunas da matriz M_1 transformam-se nas linhas da matriz M_2. Conforme destacado nas linhas tracejadas nas representações de matrizes das figuras 2.4 e 2.5, as larguras de banda das linhas 4, 5 e 6, isto é, $\beta_4(M_1)$, $\beta_5(M_1)$ e $\beta_6(M_1)$, não são contadas no *profile* de M_2 na heurística CMr. Os primeiros coeficientes dessas três linhas de M_1 pertencem à última linha de M_2. Com isso, há redução no *profile* de M_2 em relação ao *profile* de M_1.

Matriz	Largura de banda	$Profile$
M (original)	10	39
M_1 (CM)	5	33
M_2 (CMr)	5	24

Tabela 2.2: Largura de banda e *profile* de matrizes de exemplos para as heurísticas CM e CMr.

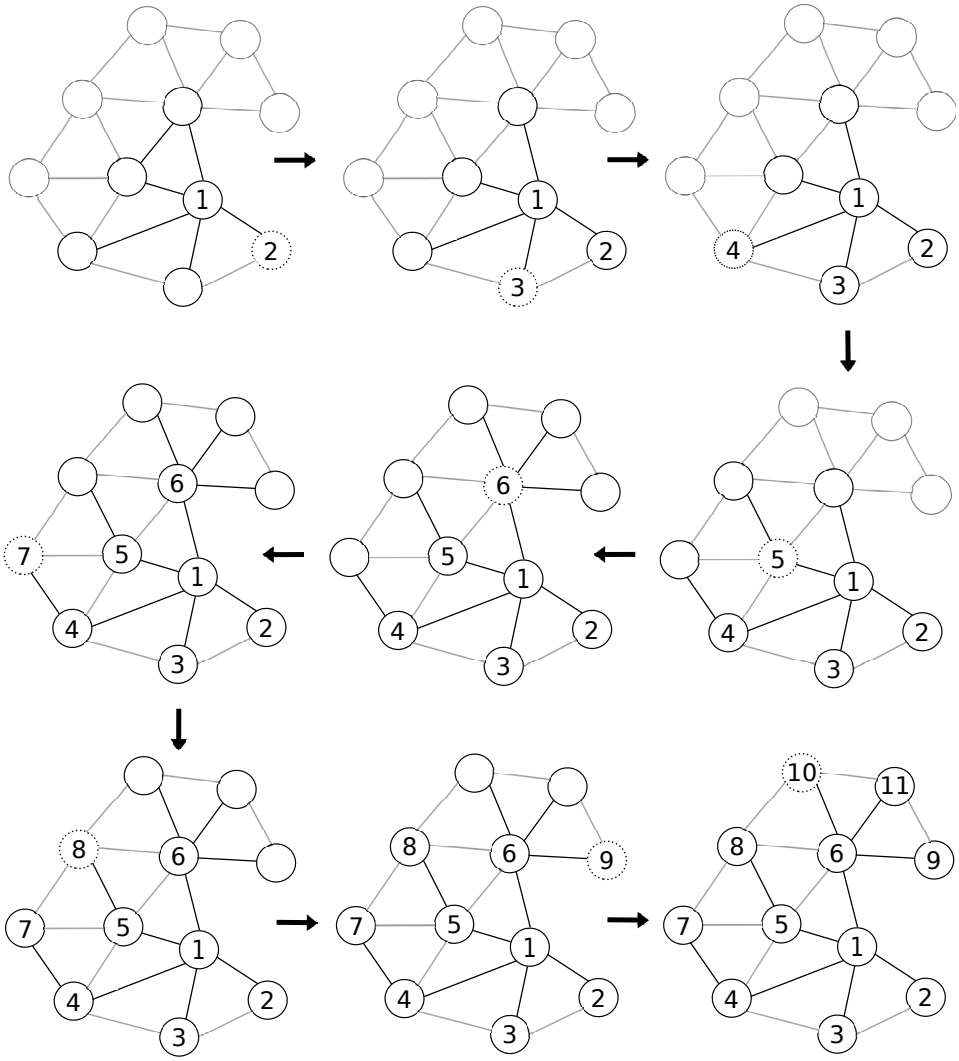

Figura 2.3: Um exemplo, passo a passo, da renumeração de vértices de um grafo pela heurística Cuthill-McKee.

2.3 Heurística GPS

A heurística GPS [26] foi desenvolvida após estudo detalhado da heurística Cuthill-McKee reverso (CMr) [22]. Gibbs, Poole e Stockmeyer [26] verificaram que a heurística CMr [22] não era adequada em situações em que há um custo computacional excessivo para se encontrar o vértice inicial. Nas heurísticas CM e CMr, encontram-se vértices com grau mínimo do grafo, geram-se as estruturas de nível de todos esses vértices e escolhem-se as estruturas de nível que apresentarem a menor largura de nível. A heurística CMr [22] realizaria esforço desnecessário nas situações em que todos os vértices do grafo possuíssem o mesmo grau, por exemplo. Isso implicaria na montagem da estrutura de nível enraizada por meio da busca em largura para cada vértice do grafo e seriam realizadas as renumerações por todas essas estruturas de nível.

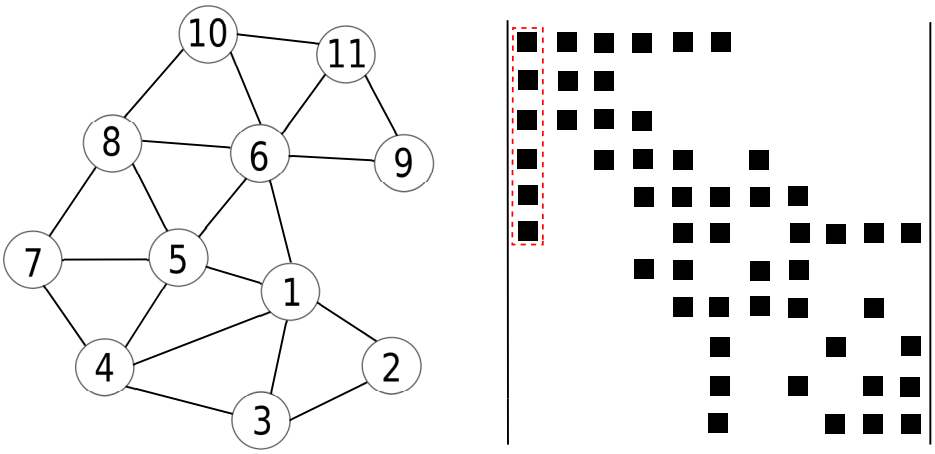

Figura 2.4: Grafo reordenado pela heurística Cuthill-McKee e sua representação matricial M_1.

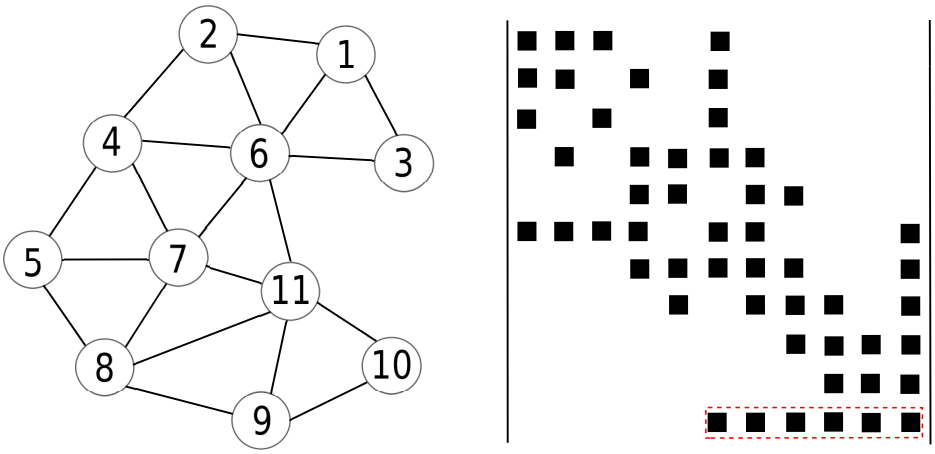

Figura 2.5: Grafo reordenado pela heurística Cuthill-McKee reverso e sua representação matricial M_2.

A heurística de Gibbs, Poole e Stockmeyer [26] pode ser dividida em três passos. No primeiro passo, encontra-se um par de vértices iniciais. Esse passo é apresentado na subseção 2.3.1. No segundo passo, reduz-se a largura de nível da estrutura montada no passo anterior. Esse passo é descrito na subseção 2.3.2. Por último, realiza-se a renumeração dos vértices do grafo de uma forma similar à heurística Cuthill-McKee reverso, iniciando-se no vértice encontrado no primeiro passo. Esse passo é descrito na subseção 2.3.3.

2.3.1 Escolha dos vértices iniciais

Cuthill e Mckee [9] constataram que, quanto mais níveis uma estrutura de nível tem, menor será sua largura de banda, se a renumeração for realizada por meio dessa estrutura de nível. Gibbs, Poole e Stockmeyer [26] utilizaram essa noção para a escolha por qual vértice inicializar a renumeração dos vértices.

Considere um grafo $G = (V, E)$ e um vértice $v \in V$ com grau mínimo do grafo. O algoritmo gera a estrutura de nível enraizada $\mathscr{L}(v)$. Em seguida, o algoritmo também gera estruturas de nível enraizadas para vértices folhas de $\mathscr{L}(v)$, em ordem crescente de grau, e determina um vértice $w \in L_{\ell(v)}(v)$ com excentricidade $\ell(w) > \ell(v)$. Se $\ell(w) > \ell(v)$, então, atribui-se w a v e repete-se o processo até que não tenha vértice em $L_{\ell(v)}(v)$ com excentricidade maior que a excentricidade de v. O vértice v será um dos dois vértices iniciais do algoritmo. Para se obter o segundo vértice inicial u, dentre os vértices folhas de $\mathscr{L}(v)$, escolhe-se o vértice que apresentar $\mathscr{L}(u)$ com a menor largura de nível.

Mostra-se, no algoritmo 2, um pseudo-código para a escolha do primeiro vértice inicial. Na linha 2, é escolhido o vértice com grau mínimo do grafo e o atribui a v. Em seguida, é montada a estrutura de nível enraizada $\mathscr{L}(v)$, utilizando-se a busca em largura. Constroem-se as estruturas de nível enraizadas dos nós folhas de $\mathscr{L}(v)$ até que algum vértice folha w tenha excentricidade maior que a excentricidade de v. Então, atribui-se w a v e repete-se o processo. Itera-se na estrutura de repetição nas linhas 5 a 17 enquanto a excentricidade de algum vértice folha de $\mathscr{L}(v)$ for maior que a excentricidade de v. Ao sair da estrutura de repetição *enquanto* externa, o vértice v é um dos vértices iniciais da heurística GPS.

Para se determinar o segundo vértice periférico, utiliza-se a sub-rotina *Vertice-MenorLarguraDeNivel*, mostrada no algoritmo 3. Note que as estruturas de nível enraizadas dos nós folhas de $\mathscr{L}(v)$ foram construídas na linha 10 do algoritmo 2. A sub-rotina *VerticeMenorLarguraDeNivel* recebe $L_{\ell(v)}(v)$. Com a estrutura de repetição *para* externa, nas linhas 4 a 10, percorrem-se os nós folhas w_i de $\mathscr{L}(v)$, para $i = 1, 2, \cdots, |L_{\ell(v)}(v)|$. Uma forma de se determinar a largura de nível de $\mathscr{L}(w_i)$ é mostrada no algoritmo 4. O algoritmo 4 é utilizado em diversos pseudo-códigos deste texto. No algoritmo 3, ao sair da estrutura de repetição *para* externa, estará armazenado em u o vértice folha de $\mathscr{L}(v)$ com a menor largura de nível, que é retornado.

Na figura 2.6, mostra-se um grafo e a sua representação matricial com os vértices iniciais determinados pela heurística GPS. O vértice 1 foi determinado como um vértice inicial porque possui grau três, que é um vértice com grau mínimo do grafo (além de outros vértices com grau três). Por meio da montagem da estrutura de nível enraizada do vértice 1, obtém-se a excentricidade $\ell(1) = 5$ e os vértices folhas de $\mathscr{L}(1)$ são os vértices 12, 14 e 18. Como esses vértices possuem excentricidade 5, então, um dos vértices iniciais é o vértice 1. Calculando-se a largura de nível das folhas de $\mathscr{L}(1)$, encontra-se $b(\mathscr{L}(14)) = b(\mathscr{L}(18)) = 4$ e $b(\mathscr{L}(12)) = 5$. Com isso, escolheu-se o vértice 18 como o segundo vértice inicial da heurística GPS.

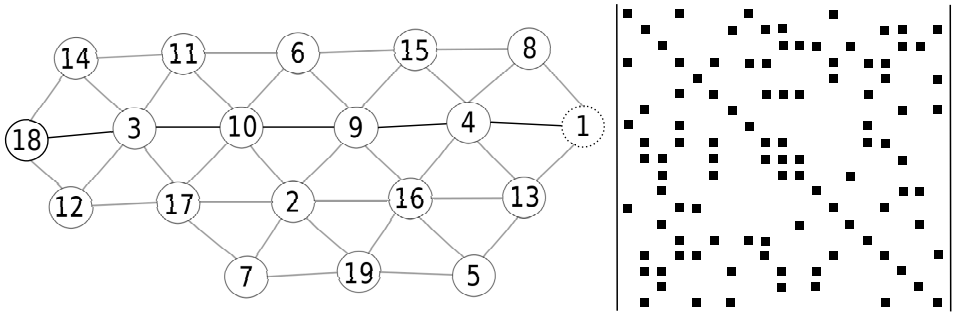

Figura 2.6: Um grafo de exemplo para a renumeração pela heurística GPS com o vértices iniciais 1 e 18 destacados e a representação matricial M_G do grafo.

Algoritmo 2: Escolha_GPS (escolha do primeiro vértice inicial para a heurística GPS).

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $v \in V$;

```
1  início
2      v ← VerticeComGrauMinimo(G);
       // constrói-se estrutura de nível enraizada
3      ℒ(v) ← BuscaEmLargura(v);
4      trocou ← verdadeiro;
5      enquanto ( trocou = verdadeiro ) faça
6          trocou ← falso;
           // cria fila de prioridades em ordem crescente de grau
7          F ← FilaPrioridades(L_ℓ(v)(v)); // dos vértices em L_ℓ(v)(v)
8          repita
9              w ← F.Remove();
               // constrói-se estrutura de nível enraizada
10             ℒ(w) ← BuscaEmLargura(w);
               // ℓ(w) e ℓ(v) são as excentricidades dos vértices
11             se ( ℓ(w) > ℓ(v) ) então
12                 v ← w;
13                 ℒ(v) ← ℒ(w);
14                 trocou ← verdadeiro;
15             fim-se;
16         até que ((trocou = verdadeiro) ∨ (F = ∅)) ;
17     fim-enquanto;
18     retorna v;
19 fim.
```

Algoritmo 3: *VérticeMenorLarguraDeNível* (escolhe vértice com menor largura de nível).

Entrada: conjunto composto pelos nós folhas $w_i \in L_{\ell(v)}(v)$;
Saída: segundo vértice pseudo-periférico $u \in V$ para a heurística GPS;

```
1  início
2      u ← ∅; // u recebe nulo
3      largura ← +∞;
4      para ( i ← 1; i ≤ |L_ℓ(v)(v)|; i ← i + 1 ) faça
           // constrói-se estrutura de nível enraizada
5          ℒ(w_i) ← BuscaEmLargura(w_i);
           // a sub-rotina b é mostrada no algoritmo 4
6          se ( b(ℒ(w_i)) < largura ) então
7              largura ← b(ℒ(w_i));
8              u ← w_i;
9          fim-se;
10     fim-para;
11     retorna u;
12 fim.
```

Algoritmo 4: b.

Entrada: estrutura de nível $\mathscr{L}(v)$;
Saída: largura de nível de $\mathscr{L}(v)$;

```
1 início
    // a largura de nível de L(v) é, pelo menos, 1,
    // porque o número de vértices no nível 0 é 1 (a raiz)
2   l ← 1;
    // percorrem-se os níveis de L(v)
3   para ( i ← 1; i ≤ ℓ(v); i ← i + 1 ) faça
4     se ( |L_i(v)| > l ) então l ← |L_i(v)|;
5   fim-para;
6   retorna l;
7 fim.
```

2.3.2 Redução da largura de nível

No segundo passo da heurística, cria-se uma nova estrutura de nível a partir das estruturas de nível enraizadas dos vértices iniciais v e u. A largura de nível da nova estrutura de nível será, no máximo, a menor largura de nível entre as estruturas de nível enraizadas de v e de u.

Seja $k = \ell(v) = \ell(u)$. Note que a excentricidade de v é igual às excentricidades dos nós folhas de $\mathscr{L}(v)$. Dessa forma, $(\forall u \in L_{\ell(v)}(v))\ \ell(v) = \ell(u)$, pois $v \in L_{\ell(u)}(u)$. $\mathscr{L}(v) = \{L_0(v), L_1(v), \cdots, L_k(v)\}$ e $\mathscr{L}(u) = \{L_0(u), L_1(u), \cdots, L_k(u)\}$ são as respectivas estruturas de nível enraizadas de v e u geradas no primeiro passo da heurística.

Associa-se, a cada vértice w do grafo $G = (V, E)$, um par ordenado (i, j), chamado de par associado do nível, em que i é o índice associado ao nível de $\mathscr{L}(v)$ que contém w e j é o índice associado, em ordem decrescente, de $\mathscr{L}(u)$ que contém w. Note que $k - j$ é o índice do nível em $\mathscr{L}(u)$ que contém w. O par (i, j) é associado ao vértice w se e, somente se, $w \in L_i(v) \cap L_{k-j}(u)$. Dessa forma, os pares associados aos vértices v e u são $(0, 0)$ e (k, k), respectivamente. Os passos do processo que resultarão em uma nova estrutura de nível $\mathscr{K}(\nu, \cdots) = \{K_0(\nu, \cdots), K_1(\nu, \cdots), \cdots, K_k(\nu, \cdots)\}$ são descritos a seguir.

1. A cada par na forma (i, i), associado a algum vértice w, w é colocado em $K_i(\nu, \cdots)$. Note que $(0, 0)$ é associado a v, então, $v \in K_0(\nu, \cdots)$; e (k, k) é associado a u, então, $u \in K_{\ell(\nu)}(\nu, \cdots)$. Cada vértice w e as arestas incidentes a w são removidas de $G = (V, E)$. Se $V = \varnothing$, então, pare.

2. Após o primeiro passo, tem-se $V = \bigcup_{\iota=1}^{t} \mathscr{C}_\iota$, em que os conjuntos $\mathscr{C}_\iota$ são disjuntos e t é o número de componentes. As componentes são dispostas em ordem decrescente de número de vértices, isto é, $|\mathscr{C}_1| \geq |\mathscr{C}_2| \geq \cdots \geq |\mathscr{C}_t|$.

3. Para cada $\mathscr{C}_\iota$, $\iota = 1, 2, \cdots, t$ e considerando-se $i = 0, 1, \cdots, k$; e $j = k, k - 1, \cdots, 0$, faça:

   - calcule $(n_0, n_1, \cdots, n_k)$, em que n_r é o número de vértices em $K_r(\nu, \cdots)$, para $0 \leq r \leq k$;

- calcule $(h_0, h_1, \cdots, h_k)$, em que h_r é n_r mais o número de vértices que $K_r(\nu, \cdots)$ teria se o índice i do par associado do nível fosse utilizado como a renumeração de nível;
- calcule $(l_0, l_1, \cdots, l_k)$, em que l_r é n_r mais o número de vértices que $K_r(\nu, \cdots)$ teria se o índice j do par associado do nível fosse utilizado como a renumeração de nível;
- encontre $h_{max} = \max_{0 \leq r \leq k} (h_r : h_r - n_r > 0)$, $l_{max} = \max_{0 \leq r \leq k} (l_r : l_r - n_r > 0)$ e
  - se $h_{max} < l_{max}$, então, numere o nível de cada vértice de $\mathscr{C}_\iota$ de acordo com o índice i do par associado do nível;
  - se $h_{max} > l_{max}$, então, numere o nível de cada vértice de $\mathscr{C}_\iota$ de acordo com o índice j do par associado do nível;
  - se $h_{max} = l_{max}$, então:
    - se $b(\mathscr{L}(u)) \geq b(\mathscr{L}(v))$, então, numere o nível de cada vértice de $\mathscr{C}_\iota$ de acordo com o índice i do par associado do nível;
    - se $b(\mathscr{L}(u)) < b(\mathscr{L}(v))$, então, numere o nível de cada vértice de $\mathscr{C}_\iota$ de acordo com o índice j do par associado do nível.

Utilizam-se os níveis resultantes da combinação das estruturas de nível enraizadas dos vértices v e u. Para isso, verifica-se qual quantidade é menor, h_{max} ou l_{max}.

Um exemplo da execução da segunda parte da heurística GPS, considerando-se os vértices iniciais $v = 1$ e $u = 18$ do grafo da figura 2.6, é mostrado nas figuras 2.7, 2.8 e 2.10. Na figura 2.7, associados aos vértices, são mostrados os índices (i, j) referentes aos níveis das estruturas de nível dos vértices 1 e 18.

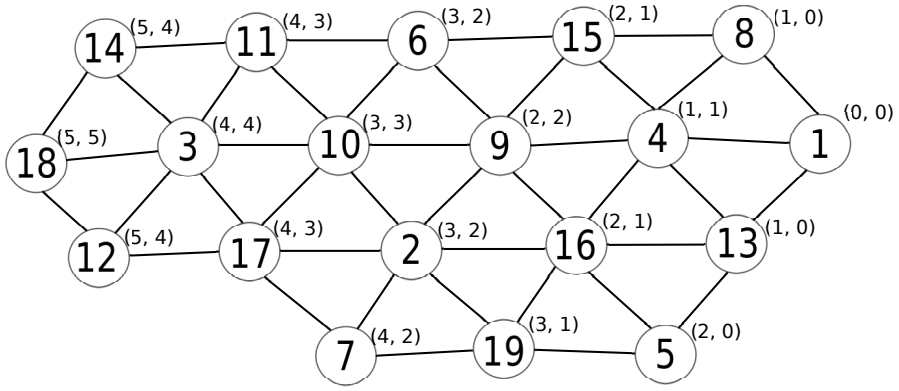

Figura 2.7: Grafo com os vértices associados aos índices das estruturas de nível de $v = 1$ e de $u = 18$. O primeiro índice do par (i, j) indica o nível do vértice na estrutura de nível de $v = 1$ e o segundo índice indica o nível do vértice na estrutura de nível de $u = 18$.

Na figura 2.8, mostram-se os índices dos vértices 1, 3, 4, 9, 10, e 18 na nova estrutura de nível $\mathscr{K}(\nu, \cdots)$. Esses vértices são dispostos na estrutura de nível $\mathscr{K}(1, \cdots)$ porque $i = j$ em (i, j). Também, na figura 2.8, mostram-se os componentes $\mathscr{C}_1 = \{2, 5, 7, 12, 13, 16, 17, 19\}$ e $\mathscr{C}_2 = \{6, 8, 11, 14, 15\}$. Note que, na figura 2.8, os vértices de $\mathscr{C}_1$ e de $\mathscr{C}_2$ não possuem índices no estágio corrente do algoritmo.

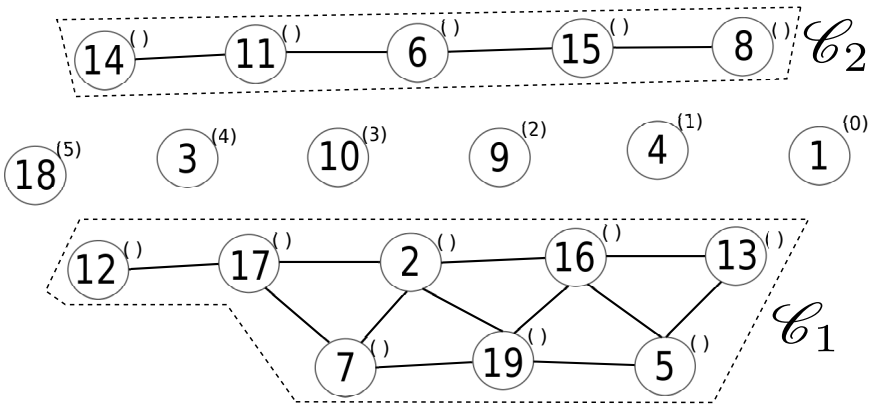

Figura 2.8: Os vértices que não estão em $\mathscr{C}_1$ ou $\mathscr{C}_2$ estão nos níveis $K_0(\nu,\cdots), K_1(\nu,\cdots),\cdots,K_k(\nu,\cdots)$, em que $k = \ell(1) = \ell(18)$. Os índices estão mostrados juntos desses vértices. O grafo corrente é $G = \mathscr{C}_1 \cup \mathscr{C}_2$.

Em seguida, gera-se $(n_0, n_1, n_2, n_3, n_4, n_5) = (1,\ 1,\ 1,\ 1,\ 1,\ 1)$ porque tem um vértice em cada nível de $\mathscr{K}(1,\cdots)$, ou seja, cada um dos vértices 1, 4, 9, 10, 3 e 18 estão em um nível de $\mathscr{K}(1,\cdots)$ (veja os índices associados a esses vértices na figura 2.8). Com isso, de $\mathscr{C}_1$, geram-se:

- $(h_0, h_1, h_2, h_3, h_4, h_5) = (1,\ 1,\ 1,\ 1,\ 1,\ 1) + (0,\ 1,\ 2,\ 2,\ 2,\ 1) = (1,\ 2,\ 3,\ 3,\ 3,\ 2)$, pois para $0 \leq r \leq k$, h_r é n_r adicionado do número de vértices de $\mathscr{C}_1$, no nível r, considerando-se o índice i (veja os índices i associados aos vértices em $\mathscr{C}_1$ na figura 2.7);

- $(l_0, l_1, l_2, l_3, l_4, l_5) = (1,\ 1,\ 1,\ 1,\ 1,\ 1) + (2,\ 2,\ 2,\ 1,\ 1,\ 0) = (3,\ 3,\ 3,\ 2,\ 2,\ 1)$, pois para $0 \leq r \leq k$, l_r é n_r adicionado do número de vértices de $\mathscr{C}_1$, no nível r, considerando-se o índice j (veja os índices j associados aos vértices em $\mathscr{C}_1$ na figura 2.7).

Com isso, $h_{max} = l_{max} = 3$ e $b(\mathscr{L}(u)) = b(\mathscr{L}(v))$. Então, os vértices de $\mathscr{C}_1$ têm os seus índices da estrutura de nível organizados e incorporados ao grafo de acordo com o índice i, como pode ser visto na figura 2.9.

Após associar os vértices de $\mathscr{C}_1$ na nova estrutura de nível, repetem-se os passos para $\mathscr{C}_2$. Nesta etapa do algoritmo, tem-se $(n_0, n_1, n_2, n_3, n_4, n_5) = (1,\ 2,\ 3,\ 3,\ 3,\ 2)$, pois $\mathscr{C}_1$ foi incorporado à estrutura de nível $\mathscr{K}(\nu,\cdots)$, de acordo com o índice i do par associado do nível. Por isso, de $\mathscr{C}_2$, geram-se:

- $(h_0, h_1, h_2, h_3, h_4, h_5) = (1,\ 2,\ 3,\ 3,\ 3,\ 2) + (0,\ 1,\ 1,\ 1,\ 1,\ 1) = (1,\ 3,\ 4,\ 4,\ 4,\ 3)$, pois para $0 \leq r \leq k$, h_r é n_r adicionado do número de vértices de $\mathscr{C}_2$, no nível r, considerando-se o índice i (veja os índices i associados aos vértices em $\mathscr{C}_2$ na figura 2.7);

- $(l_0, l_1, l_2, l_3, l_4, l_5) = (1,\ 2,\ 3,\ 3,\ 3,\ 2) + (1,\ 1,\ 1,\ 1,\ 1,\ 0) = (2,\ 3,\ 4,\ 4,\ 4,\ 2)$, pois para $0 \leq r \leq k$, h_r é n_r adicionado do número de vértices de $\mathscr{C}_2$, no nível r, considerando-se o índice j (veja os índices j associados aos vértices em $\mathscr{C}_2$ na figura 2.7.

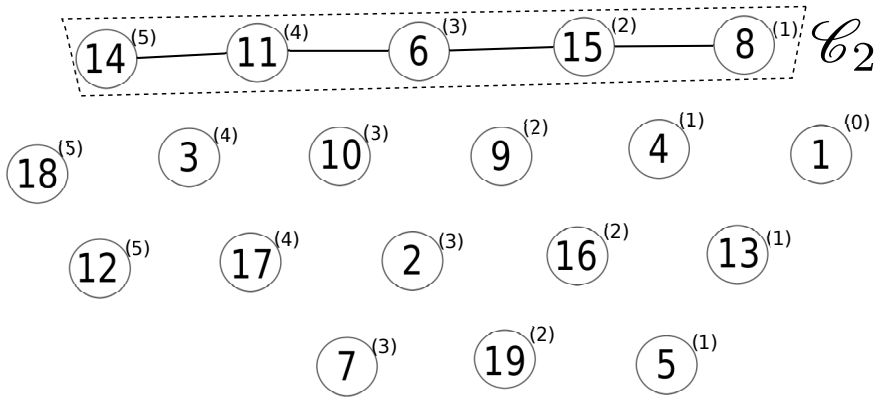

Figura 2.9: Os vértices de $\mathscr{C}_1$, mostrados na figura 2.8, foram incorporados na nova estrutura de nível de acordo com o índice i de (i, j), mostrados na figura 2.7.

Com isso, $h_{max} = l_{max} = 4$ e $b(\mathscr{L}(u)) = b(\mathscr{L}(v))$. Então, os vértices de $\mathscr{C}_2$ têm os seus índices da estrutura de nível organizados de acordo com o índice i, como pode ser visto na figura 2.10.

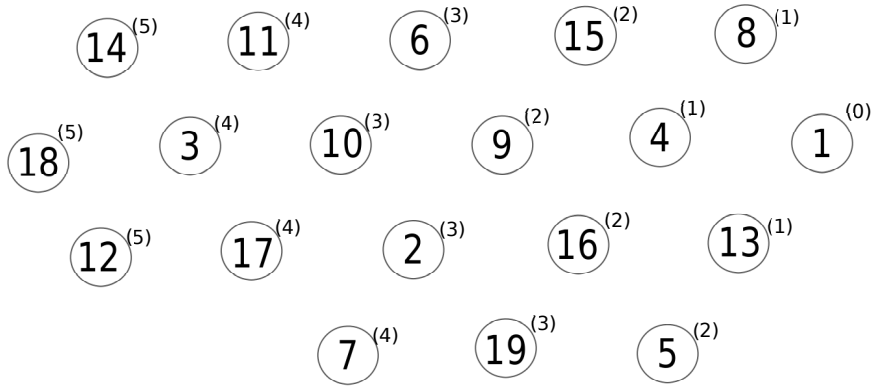

Figura 2.10: Os vértices de $\mathscr{C}_2$, mostrado na figura 2.8, foram incorporados na nova estrutura de nível de acordo com o índice i de (i, j), mostrados na figura 2.7.

Com essa estrutura de nível, obtêm-se os vértices com a ordem:

- 1 porque está em $K_0(\nu, \cdots)$;
- 8, 13 e 4 porque estão em $K_1(\nu, \cdots)$ e possuem graus 3, 4 e 6, respectivamente;
- 5, 15, 9 e 16 porque estão em $K_2(\nu, \cdots)$ e possuem graus 3, 4, 6 e 6, respectivamente;
- $\cdots$;
- 12, 18 e 14 porque estão em $K_5(\nu, \cdots)$ e possuem o mesmo grau, com isso, a ordem entre esses vértices é arbitrária.

2.3.3 Renumeração

A ordem para percorrer os vértices é dada considerando-se a estrutura de nível obtida no passo mostrado na subseção 2.3.2. Realiza-se a renumeração dos vértices iniciando-se pelo nível $K_k(\nu, \cdots)$ até o nível $K_0(\nu, \cdots)$. Para vértices de mesmo nível, primeiramente, percorre-se o que tem grau maior. No exemplo mostrado nas figuras 2.6 a 2.10, a renumeração é dada na ordem inversa ao que foi apresentado no final da subseção 2.3.2.

A renumeração final do grafo da figura 2.6 e a sua representação matricial são mostradas na figura 2.11. Para uma comparação, mostra-se, na figura 2.12, a renumeração, pela heurística CMr, do grafo da figura 2.6, considerando-se o vértice 1 como o vértice inicial. Na tabela 2.3, observa-se que bons resultados (reduções de largura de banda e de *profile* em relação à matriz M_G da figura 2.6) são obtidos pela heurística CMr ao se escolher um vértice inicial com excentricidade aproximada ao diâmetro do grafo. Veja o capítulo 4, na página 69, para detalhes sobre vértice pseudo-periférico. No caso da figura 2.6, $\ell(1) = 5$, que é o diâmetro do grafo. Na tabela 2.3, também mostram-se as larguras de banda e os *profiles* de M_{G_1} e de M_{G_2} das figuras 2.11 e 2.12, obtidos pelas renumerações do grafo da figura 2.6 por meio das heurísticas GPS e CMr, respectivamente.

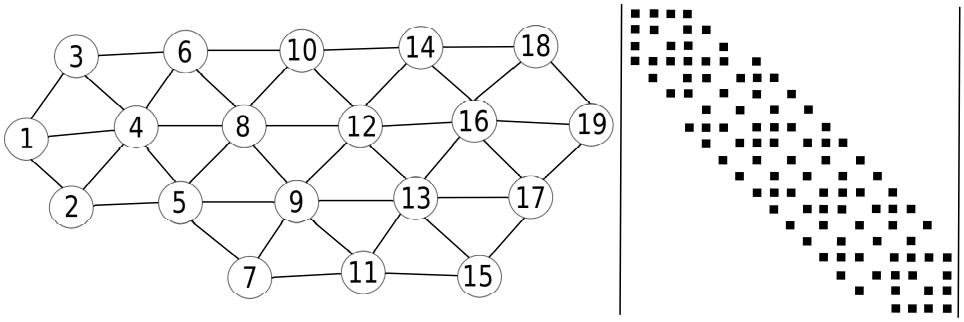

Figura 2.11: Grafo da figura 2.6 renumerado pela heurística GPS e sua representação matricial M_{G_1}.

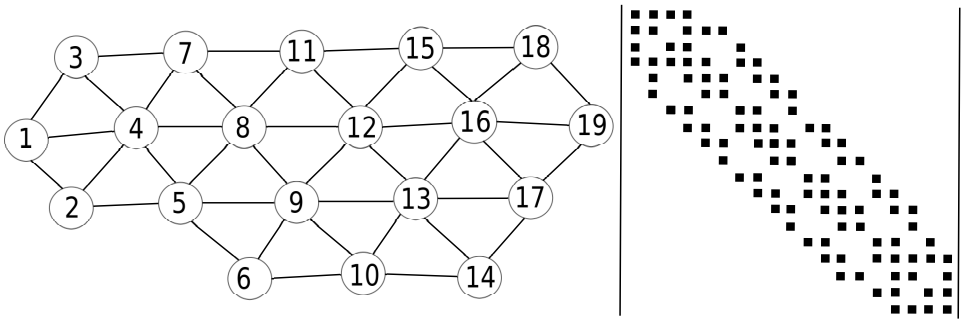

Figura 2.12: Grafo da figura 2.6 renumerado pela heurística CMr e sua representação matricial M_{G_2}.

Matriz	Largura de banda	$Profile$
M_G (original) da figura 2.6	17	142
M_{G_1} (GPS) da figura 2.11	4	61
M_{G_2} (CMr) da figura 2.12	4	63

Tabela 2.3: Largura de banda e de *profile* das matrizes mostradas nas figuras 2.6, 2.11 e 2.12.

2.4 Heurística de Snay

Uma heurística para a permutação de linhas e colunas de uma matriz simétrica para a redução do *profile* foi proposta por Snay [80]. Descreve-se o algoritmo de Snay na subseção 2.4.1. É mostrada a heurística de Snay [80] para a determinação de vértices iniciais na subseção 2.4.2.

2.4.1 Descrição do algoritmo de Snay

Considere o grafo $G = (V, E)$. Considere, também, um conjunto de vértices H composto por vértices *esperançosos*, que são vértices que ainda não foram processados e que são adjacentes a vértices já processados.

A heurística de Snay [80], mostrada no algoritmo 5, recebe o vértice v inicial e referencia-se esse vértice na primeira entrada da renumeração $S = \{s(1), s(2), \cdots, s(|V|)\}$. Isso é representado como $s(i) \leftarrow v$ na linha 2 do algoritmo 5. Nas linhas 3 e 4, estabelecem-se como vazios os conjuntos de vértices esperançosos (H) e candidatos.

Percorrem-se todos os vértices $u \in V - \{v\}$ na estrutura de repetição das linhas 5 a 25. O algoritmo de Snay [80] escolhe o candidato com menor grau como o i-ésimo vértice da renumeração, sem considerar as adjacências a vértices não esperançosos e não processados. No início da iteração i, na linha 6, os novos vértices esperançosos são os vértices adjacentes a $s(i-1)$ e que não pertencem a S. O conjunto H é acrescido desses vértices adjacentes ao vértice $s(i-1)$ e que não pertencem a S. Na linha 8, o conjunto de vértices candidatos é acrescido dos novos vértices esperançosos. Nas linhas 9 a 11, o conjunto de vértices candidatos é acrescido dos vértices adjacentes dos novos vértices esperançosos, isto é, incluem-se no conjunto de vértices candidatos os vértices no segundo nível de adjacência de $s(i-1)$. Note que não se incluem no conjunto de vértices candidatos os vértices pertencentes a S. A limitação dos vértices candidatos é para tornar o algoritmo mais rápido do que considerar todos os vértices não renumerados como candidatos.

Na estrutura de repetição nas linhas 14 a 21, o vértice candidato escolhido para ser renumerado em uma iteração é o vértice que possui o menor grau, sem considerar as adjacências a vértices não esperançosos e não processados na linha 15. Além disso, os vértices esperançosos são favorecidos em relação aos demais vértices candidatos, na linha 16. O vértice v_{min} a ser renumerado na iteração i é referenciado na entrada $s(i)$, na linha 22. Após ser renumerado, o vértice v_{min} deixa de ser tanto um vértice esperançoso, na linha 23, como um vértice candidato, na linha 24. O algoritmo retorna a renumeração S na linha 26.

Algoritmo 5: *AlgoritmoSnay.*

Entrada: grafo conexo $G = (V, E)$ e um vértice inicial $v \in V$;
Saída: renumeração S para V, com $|V|$ entradas $s(1), s(2), \cdots, s(|V|)$;

início
  $s(1) \leftarrow v$; `// inicia-se a renumeração pelo vértice inicial`
  $H \leftarrow \varnothing$; `// inicializam-se os conjuntos de vértices`
  $candidatos \leftarrow \varnothing$; `// esperançosos e candidatos como vazios`
  para ($i \leftarrow 2; i \leq |V|; i \leftarrow i + 1$ **) faça**
    `// vértices não renumerados adjacentes a` $s(i-1)$
    `// tornam-se vértices esperançosos`
    $NovosEsperançosos \leftarrow Adj(G, s(i-1)) - \{s(1), s(2), \cdots, s(i-1)\}$;
    $H \leftarrow H \cup NovosEsperançosos$;
    `// vértices não processados e adjacentes`
    `// a` $s(i-1)$ `tornam-se esperançosos`
    $candidatos \leftarrow candidatos \cup NovosEsperançosos$;
    para cada ($u \in NovosEsperançosos$ **) faça**
      `// torna candidatos os vértices não processados do`
      `// segundo nível de adjacência de` $s(i-1)$
      $candidatos \leftarrow candidatos \cup (Adj(G, u) - \{s(1), ..., s(i-1)\})$;
    fim-para-cada;
    $min \leftarrow +\infty$;
    $v_{min} \leftarrow \emptyset$; `//` v_{min} `recebe nulo`
    `// escolhe-se o candidato` w `com menor grau, sem contar as`
    `// adjacências a não esperançosos e não processados`
    para cada ($w \in candidatos$ **) faça**
      `// número de vértices não processados e`
      `// não esperançosos adjacentes a` w
      $grauRel\_w \leftarrow |Adj(G, w) - (\{s(1), ..., s(i-1)\} \cup H)|$;
      `// os vértices esperançosos são favorecidos`
      `// em relação aos demais vértices candidatos`
      se ($w \in H$ **) então** $grauRel\_w \leftarrow grauRel\_w - 1$;
      se ($grauRel\_w < min$ **) então**
        $min \leftarrow grauRel\_w$;
        $v_{min} \leftarrow w$;
      fim-se;
    fim-para-cada;
    $s(i) \leftarrow v_{min}$; `// inclui o candidato escolhido na sequência`
    `// se o escolhido era vértice esperançoso, retira-o de` H
    $H \leftarrow H - \{v_{min}\}$;
    `// o vértice escolhido é retirado também do conjunto`
    $candidatos \leftarrow candidatos - \{v_{min}\}$; `// de candidatos`
  fim-para;
  retorna S*;*
fim.

2.4.2 Vértices iniciais pseudo-periféricos

Considere que a distância entre dois vértices $u, v \in V$ é o tamanho do menor caminho entre u e v, ou seja, *d(v,u)*, conforme mostrado na definição 5, na página 8. O

método para a determinação dos candidatos a vértice inicial, mostrado em Snay [80], consiste em considerar, a partir de um vértice aleatório $v \in V$, um conjunto D com os *cinco* vértices mais distantes de v. Considera-se, também, o conjunto Q dos cinco vértices mais distantes do vértice $x \in S$, em que x é o vértice mais distante de v. Os conjuntos de vértices D e Q contêm os candidatos a vértice inicial da heurística de Snay [80].

Esse método de determinação dos candidatos a vértice inicial é similar à heurística GPS [26]. A heurística GPS determina dois vértices cuja distância entre os mesmos se aproxima do diâmetro do grafo (vértices pseudo-periféricos). Para isso, a heurística GPS utiliza a estrutura de nível. Para detalhes sobre a heurística GPS, veja a seção 2.3, na página 18.

Ao escolher os vértices mais distantes de um vértice inicial v aleatório, a heurística de Snay [80] considera, de maneira implícita, os vértices do(s) último(s) nível(is) da estrutura de nível enraizada de v. Porém, há uma escolha arbitrária de apenas *cinco* desses vértices, sem estabelecer critérios de desempate caso dois ou mais vértices tenham a mesma distância de v. A mesma arbitrariedade e falta de critério da escolha anterior acontecem tanto na determinação do vértice mais distante de v como na determinação do segundo conjunto de vértices candidatos a vértice inicial. Ao se desconsiderar as arbitrariedades e faltas de critérios de desempate, pode-se considerar que a heurística de Snay [80] executa exatamente duas iterações da heurística GPS para determinar os vértices *pseudo-periféricos*.

No algoritmo 6, utiliza-se a estrutura de nível para se determinar os vértices mais distantes a um determinado vértice. O algoritmo 6 recebe um grafo $G = (V,E)$ e tem como saída um conjunto de vértices pseudo-periféricos $D \cup Q$. Na condição da linha 2, exige-se que o grafo tenha, pelo menos, 10 vértices. Um vértice aleatório é atribuído a v na linha 3. Na linha 4, constrói-se a estrutura de nível enraizada de v. A construção do conjunto de vértices mais distantes a v inicia-se na linha 5. A variável k, inicializada na linha 6, é utilizada para que $|D| = 5$, pois o(s) último(s) nível(is) da estrutura de nível enraizada de v pode(m) não ter cinco vértices. A variável u, inicializada na linha 7, é utilizada para armazenar um vértice mais distante a v. A variável i, inicializada na linha 8, é utilizada na estrutura de repetição das linhas 9 a 20, em que se inserem os cinco vértices em D na linha 11. Constrói-se a estrutura de nível enraizada de u na linha 15. Da mesma forma, cinco vértices mais distantes a u são inseridos em Q, nas linhas 21 a 31. Por simplidade da apresentação, mostram-se os dois trechos de código similares em seguida, das linhas 6 a 20 e das linhas 22 a 31. Finalmente, retorna-se $D \cup Q$, na linha 32.

No algoritmo 7, avalia-se a largura de banda de $G = (V,E)$ para cada vértice pseudo-periférico. O algoritmo recebe o grafo $G = (V, E)$ e tem como saída uma renumeração dos vértices em V.

Na condição da linha 2, exige-se que o grafo tenha, pelo menos, 10 vértices. Chama-se o algoritmo 6 e inserem-se os vértices pseudo-periféricos no conjunto C, na linha 3. As variáveis *menorLarguraBanda* e *melhorOrdem* são inicializadas nas linhas 4 e 5, respectivamente. Na estrutura de repetição das linhas 6 a 13, para cada vértice $w \in C$, avalia-se a renumeração com w como o vértice inicial na linha 7 e a largura de banda dessa renumeração na linha 8. Atualizam-se as variáveis *menorLarguraBanda* e *melhorOrdem* nas linhas 9 a 12. Finalmente, a melhor renumeração é retornada na linha 14.

Algoritmo 6: *VerticesPseudoPerifericosSnay* (vértices pseudo-periféricos para o algoritmo de Snay [80]).

Entrada: grafo $G = (V, E)$;
Saída: conjunto de vértices pseudo-periféricos $D \cup Q$;

```
1  início
2      se ( |V| < 10 ) então retorna;
3      v ← VerticeAleatorio(V);
       // constrói-se estrutura de nível enraizada
4      L(v) ← BuscaEmLargura(v);
5      D ← ∅; // conterá cinco vértices mais distantes de v
6      k ← 0; // inserem-se, pelo menos, cinco vértices em D
7      u ← ∅; // um dos vértices mais distantes de v
8      i ← 0;
9      enquanto ( i < 5 ) faça
10         para cada ( w ∈ L_{ℓ(v)−k}(v) ) faça
11             D ← D ∪ {w};
12             i ← i + 1;
               // determina-se um vértice u mais distante de v
13             se ( u = ∅ ) então
14                 u ← w;
                   // constrói-se estrutura de nível enraizada
15                 L(u) ← BuscaEmLargura(u);
16             fim-se;
               // sai da estrutura de repetição para cada
17             se ( i ≥ 5 ) então break;
18         fim-para-cada;
19         k ← k + 1;
20     fim-enquanto;
       // cinco vértices mais distantes a um vértice
21     Q ← ∅; // mais distante de v são inseridos em Q
22     k ← 0;
23     i ← 0;
24     enquanto ( i < 5 ) faça
25         para cada ( w ∈ L_{ℓ(u)−k}(u) − D ) faça
26             Q ← Q ∪ {w};
27             i ← i + 1;
               // sai da estrutura de repetição para cada
28             se ( i ≥ 5 ) então break;
29         fim-para-cada;
30         k ← k + 1;
31     fim-enquanto;
32     retorna D ∪ Q;
33 fim.
```

2.5 Heurística quatro-etapas

Essa heurística foi proposta por Kaveh [40], que utiliza a estrutura *Shortest Route Tree* (SRT). A SRT é utilizada em todas as quatro etapas da heurística. Essa estrutura é comentada na subseção 2.5.1.

Algoritmo 7: calcula largura de banda para cada vértice pseudo-periférico do algoritmo de Snay.

Entrada: grafo $G = (V, E)$;
Saída: renumeração S para V;
1 **início**
2 **se** ($|V| < 10$) **então retorna**;
3 $C \leftarrow VerticesPseudoPerifericosSnay(G)$; `// algoritmo 6`
4 $menorLarguraBanda \leftarrow +\infty$;
5 $melhorOrdem \leftarrow \varnothing$;
6 **para cada** ($w \in C$) **faça**
7 $S \leftarrow AlgoritmoSnay(G, w)$; `// algoritmo 5`
8 $larguraBanda \leftarrow CalculaLarguraBanda(G, S)$;
9 **se** ($larguraBanda < menorLarguraBanda$) **então**
10 $menorLarguraBanda \leftarrow larguraBanda$;
11 $melhorOrdem \leftarrow S$;
12 **fim-se;**
13 **fim-para-cada;**
14 **retorna** $melhorOrdem$;
15 **fim.**

A qualidade dos resultados da heurística quatro-etapas depende da escolha de um vértice inicial adequado. Com isso, a heurística quatro-etapas inicia por um vértice pseudo-periférico. Para a primeira etapa, Kaveh [40] propôs seis algoritmos para encontrar vértice pseudo-periférico. Esses algoritmos são abordados na seção 4.3, na página 71.

A segunda etapa consiste em montar a estrutura SRT do vértice inicial v_0, selecionado na primeira etapa. Na prática, a SRT pode ser montada na primeira etapa, já que pode-se utilizar a SRT para se determinar o vértice inicial.

Na terceira etapa, encontra-se um caminho para percorrer cada nível (ou contorno) $C_0(v_0), C_1(v_0), \cdots, C_{\ell(v_0)}(v_0)$ em $\mathscr{R}(v_0)$, isto é, a estrutura SRT do vértice inicial v_0. Encontrar esse caminho é possível porque armazenam-se as conectividades na SRT, o que não seria possível na estrutura de nível enraizada. Nesse caminho, há somente um vértice em cada nível da SRT. Na subseção 2.5.2, apresenta-se a escolha do *caminho transversal*.

Ao se considerar a heurística CMr [22] começando por um vértice pseudo-periférico, a renumeração por essa heurística é uma busca em largura modificada, em que os vértices de cada nível são renumerados na ordem crescente de grau. Na heurística quatro-etapas, os primeiros vértices renumerados de cada nível são os *vértices representativos do nível*, ou seja, os vértices do caminho transversal. A renumeração de cada nível é dada pela distância de cada nível em relação ao vértice representativo do nível. São essas as diferenças da heurística quatro-etapas em relação à heurística CMr.

Na última etapa da heurística, realiza-se a renumeração dos vértices do grafo na ordem inversa. Essa renumeração é realizada separadamente em cada nível da estrutura de nível. Descreve-se, na subseção 2.5.3, a renumeração realizada pela heurística quatro-etapas. Finalmente, apresenta-se, na subseção 2.5.4, um exemplo da heurística quatro-etapas.

2.5.1 Estrutura *Shortest Route Tree*

A estrutura de nível enraizada, apresentada na definição 9, na seção 1.4, na página 8, é uma árvore com um vértice inicial na raiz que mostra duas características principais: a distância de cada vértice para o vértice inicial e a quantidade de vértices em cada nível. Com isso, pode-se determinar a excentricidade do vértice inicial.

Pode-se afirmar que a estrutura de nível enraizada é um caso especial da *Shortest Route Tree* (SRT). Além das características da estrutura de nível enraizada, também armazenam-se as conectividades dos vértices na SRT. Isso significa que a SRT é uma forma de representação do grafo original. Isso facilita e tornam eficientes as heurísticas de Kaveh [42] que utilizam essa estrutura ao custo de mais espaço de armazenamento em relação à estrutura de nível enraizada.

A estrutura SRT é particionada como $\mathscr{R}(v_0) = \{C_0(v_0) \cup C_1(v_0) \cup \cdots \cup C_{\ell(v_0)}(v_0)\}$, em que v_0 é o vértice inicial da renumeração e armazenam-se as adjacências entre os vértices de $C_i(v_0)$ e de $C_{i+1}(v_0)$, para $0 \leq i < |V|$, para o grafo $G = (V, E)$. Neste texto, utilizam-se conceitos e termos similares à estrutura de nível enraizada. Em particular, os conjuntos $C_i(v_0)$, para $0 \leq i \leq |V|$, são os *contornos* em torno do vértice inicial e é possível determinar um caminho (ou rota) na SRT $\mathscr{R}(v_0)$ de cada vértice $v \in V - \{v_0\}$ para o vértice v_0.

2.5.2 Caminho transversal

Na terceira etapa, é selecionado um caminho transversal mínimo $p = \{v_{\ell(v)}, v_{\ell(v)-1}, \cdots, v_0\}$ que possui um vértice em cada contorno de $\mathscr{R}(v_0)$. Um caminho transversal mínimo é um caminho em que os vértices escolhidos em cada contorno de $\mathscr{R}(v_0)$ possuem grau mínimo. Note que a escolha do caminho transversal mínimo se inicia no contorno $C_{\ell(v_0)}(v_0)$ e vai até o contorno $C_0(v_0)$. A seleção desse caminho é importante para a etapa de renumeração, que é abordada na subseção 2.5.3.

Mostra-se um pseudo-código para a escolha do caminho transversal mínimo no algoritmo 8. Considere v_0 como o vértice inicial da renumeração, selecionado na primeira etapa da heurística. O algoritmo recebe uma estrutura SRT $\mathscr{R}(v_0) = \{C_0(v_0), C_1(v_0), \cdots, C_{\ell(v_0)}(v_0)\}$ proveniente da segunda etapa da heurística. No contorno 0 da estrutura SRT só há o vértice v_0. Esse vértice é o último vértice a ser referenciado no caminho transversal.

A construção do caminho transversal mínimo é iniciada pelo contorno $C_{\ell(v_0)}(v_0)$. Na linha 2, escolhe-se um vértice de $C_{\ell(v_0)}(v_0)$ com grau mínimo para ser o primeiro vértice a ser referenciado no caminho transversal.

Na estrutura de repetição *para* nas linhas 4 a 9, escolhe-se, a cada contorno, um vértice adjacente ao vértice do contorno anterior com grau mínimo para ser inserido no caminho transversal. Nessa estrutura de repetição, percorrem-se os vértices pertencentes ao contorno $C_{i-1}(v_0)$ que são adjacentes ao vértice determinado no passo anterior. O algoritmo retorna o caminho transversal, referenciado pelo vértice inicial $v_{\ell(v_0)}$, com os vértices de grau mínimo de cada contorno da estrutura SRT.

2.5.3 Renumeração

Na última etapa da heurística, renumeram-se os vértices, percorrendo-se os contornos da SRT. Cada vértice pertencente ao caminho transversal mínimo, obtido na etapa anterior, é o vértice inicial para a renumeração do seu nível. Após todos os vértices do grafo terem sido renumerados, a renumeração é invertida.

Algoritmo 8: escolha do caminho transversal mínimo.

Entrada: estrutura SRT $\mathscr{R}(v_0) = \{C_0(v_0), C_1(v_0), \cdots, C_{\ell(v_0)}(v_0)\}$;
Saída: um caminho transversal mínimo com $\ell(v_0)+1$ vértices;

início
  // o caminho transversal mínimo é iniciado por um vértice
  // $v \in C_{\ell(v_0)}(v_0)$ com grau mínimo; considera-se o caminho
  // transversal mínimo como uma referência a v e há uma
  // referência em cada vértice do caminho para o vértice
  // seguinte até o último vértice do caminho transversal
  $v_{\ell(v_0)} \leftarrow VerticeComGrauMinimo(C_{\ell(v_0)}(v_0))$;
  $v_{\ell(v_0)}.proximo \leftarrow \emptyset$;
  // como v_0 é conhecido, basta iterar a sequência até v_1
  para ($i \leftarrow \ell(v_0)$; $i > 1$; $i \leftarrow i-1$) **faça**
    // percorrem-se os vértices pertencentes ao contorno
    // $C_{i-1}(v_0)$ que são adjacentes ao vértice escolhido no
    // contorno $C_i(v_0)$; considere que $C_i^E(v_0) \cup C_{i-1}^E(v_0)$
    // contém as arestas entre os dois contornos
    $W \leftarrow \{(\forall w' \in C_{i-1}(v_0))\ w' : \{v_i, w'\} \in C_i^E(v_0) \cup C_{i-1}^E(v_0)\}$;
    $v_{i-1} \leftarrow VerticeComGrauMinimo(W)$;
    $v_i.anterior \leftarrow v_{i-1}$;
    $v_{i-1}.proximo \leftarrow v_i$; // lista duplamente encadeada
  fim-para;
  // v_0 é o único vértice em $C_0(v_0)$ e, necessariamente,
  $v_1.anterior \leftarrow v_0$; // é adjacente a v_1
  $v_0.proximo \leftarrow v_1$;
  $v_0.anterior \leftarrow \emptyset$; // término da lista encadeada
  retorna $v_{\ell(v_0)}$; // retorna referência a $v_{\ell(v_0)}$
fim.

O primeiro contorno da SRT a ser visitado é o contorno $C_0(v_0)$. Como somente há $v_0 \in C_0(v_0)$, v_0 recebe a numeração 1. Em seguida, renumeram-se os vértices de $C_1(v_0)$. Nesse contorno, a renumeração é iniciada pelo vértice v_1 do caminho transversal mínimo. Esse vértice recebe a numeração 2. Feito isso, é gerada a estrutura de nível $\mathscr{L}(v_1)$. Note que pode ser utilizada a estrutura de nível enraizada do vértice em vez da SRT porque as conectividades não são necessárias. Só se necessita criar $\mathscr{L}(v_1)$ até os vértices de $C_1(v_0)$. Renumeram-se os vértices pertencentes a $C_1(v_0)$ na sequência em que aparecerem em $\mathscr{L}(v_1)$. O mesmo é realizado com v_2. Gera-se $\mathscr{L}(v_2)$ e renumeram-se os vértices pertencentes a $C_2(v_0)$ na sequência em que aparecerem em $\mathscr{L}(v_2)$. Repete-se esse processo para se renumerarem os vértices em $C_i(v_0)$ na sequência em que aparecem em $\mathscr{L}(v_i)$, com $3 \leq i \leq \ell(v_0)$, ou seja, numeram-se os vértices $C_3(v_0), C_4(v_0), \cdots, C_{\ell v_0}(v_0)$ na sequência em que aparecem em $\mathscr{L}(v_3), \mathscr{L}(v_4), \cdots, \mathscr{L}(v_{\ell(v_0)})$, respectivamente. Em seguida, a numeração dos vértices do grafo é invertida.

2.5.4 Exemplo para a etapa de renumeração

Um exemplo de execução da etapa de renumeração é mostrado a seguir. Mostra-se, na figura 2.13, a estrutura SRT do grafo da figura 2.6 com os vértices do caminho transversal mínimo destacados. Foi montada a estrutura SRT do vértice periférico

1, supondo-se que esse vértice tenha sido escolhido na primeira etapa da heurística. Os vértices que compõem o caminho transversal mínimo são: 1, 8, 15, 6, 11 e 14.

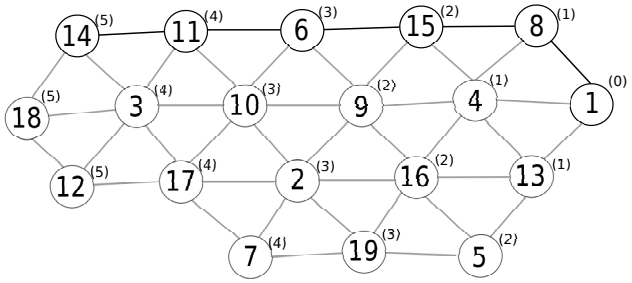

Figura 2.13: Estrutura SRT da figura 2.6 com os vértices do caminho transversal mínimo destacados.

O vértice inicial é o vértice 1, que recebe a numeração 1. O segundo vértice numerado é o vértice 8, que é o segundo vértice do caminho transversal mínimo. Mostra-se, na figura 2.14, parte da a estrutura de nível do vértice 8 com os vértices do contorno $C_1(1)$ destacados. Como o vértice $4 \in L_1(8)$ e o vértice $13 \in L_2(8)$, renumera-se o vértice 4 e depois renumera-se o vértice 13, como pode ser visto na figura 2.15. A renumeração final é invertida, e é mostrada na figura 2.16. Também é mostrada a representação matricial do grafo renumerado na figura 2.16. Mostram-se, na tabela 2.4, as reduções de largura de banda e de *profile* pela renumeração da heurística quatro-etapas. Também são mostradas as larguras de banda e *profiles* obtidas nas aplicações das heurísticas GPS e CMr no grafo da figura 2.6 na tabela 2.4.

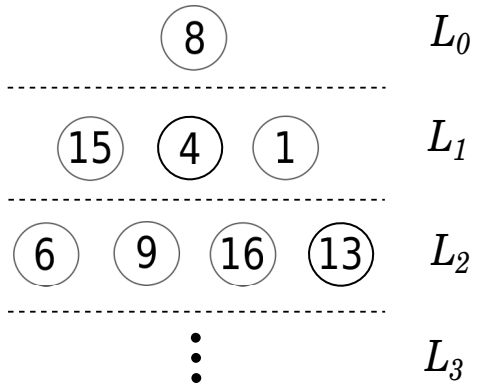

Figura 2.14: Parte da estrutura de nível do vértice 8.

Matriz	Aplicação da heurística	Largura de banda	*Profile*
M_G (original) da figura 2.6 (ou 2.13)	(matriz original)	17	142
M_{G_1} da figura 2.11	GPS	4	61
M_{G_2} da figura 2.12	CMr	4	63
M_{G_3} da figura 2.16	quatro-etapas	5	65

Tabela 2.4: Largura de banda e de *profile* das matrizes mostradas nas figuras 2.6 (e 2.13), 2.11, 2.12 e 2.16.

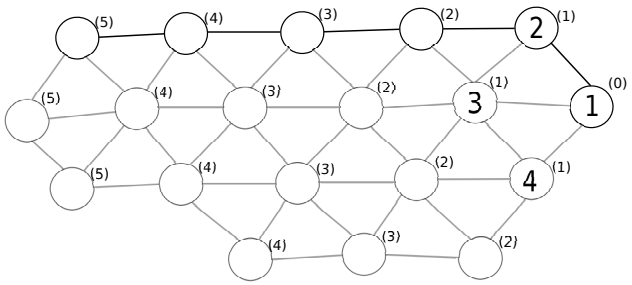

Figura 2.15: Grafo da figura 2.13 com os vértices dos contornos $C_0(1) = \{1\}$ e $C_1(1) = \{4, 8, 13\}$ renumerados.

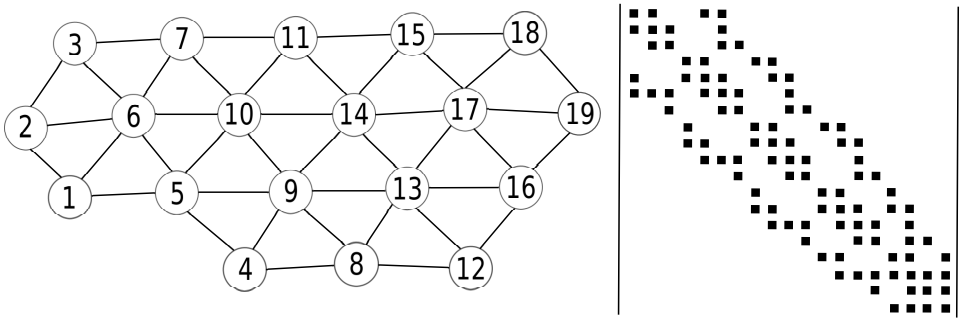

Figura 2.16: Grafo da figura 2.6 renumerado pela heurística quatro-etapas e sua representação matricial M_{G_3}.

2.6 Heurística de Boutora-Takorabet-Ibtiouen-Mezani

Boutora et al. [4] aplicaram a busca em largura para a renumeração de vértices de malhas triangulares. O grafo da figura 2.16 é um exemplo de uma malha triangular. Boutora et al. [4] iniciaram a renumeração por um vértice pseudo-periférico.

Esta seção é dividida como a seguir. Na subseção 2.6.1, algumas definições para o entendimento da implementação da heurística de Boutora et al. [4] são descritas. Na subseção 2.6.2, apresenta-se um pseudocódigo para essa implementação.

2.6.1 Definições

Considere um elemento triangular $T = (T_V, T_E)$ composto pelos vértices v, u e w, isto é, $T_V = \{u, v, w\}$ e $T_E = \{\{u, v\}, \{v, w\}, \{w, u\}\}$. Um vértice $v \in T_V$ é considerado a cúpula e os outros dois vértices, $u, w \in T_V$, são considerados a base $B(T) = \{u, w\}$.

Ainda, elementos triangulares adjacentes possuem a mesma cúpula. Considere v_i a cúpula de T_i e v_j a cúpula de T_j. Se $v_i = v_j$, então, o elemento triangular T_j é adjacente ao elemento triangular T_i. Por exemplo, suponha que um elemento triangular T_i é composto pelos vértices 17, 18 e 19 e um elemento triangular T_j é composto pelos vértices 13, 14 e 17 da malha triangular mostrada na figura 2.16. Nesse exemplo, considere que o vértice 17 é a cúpula de ambos os elementos triangulares T_i e T_j; logo, o elemento triangular T_j é adjacente ao elemento triangular

T_i.

Considere, também, um grafo $G = (V, E)$, em que V é um conjunto de vértices $V = \{v_1, v_2, \cdots, v_n\}$, conectados pelas arestas em E. Define-se $Struc(v) = \{(\forall T \subset G)\ T : v \in T_V\}$. Isso significa que $Struc(v)$ é formada pelos elementos triangulares que possuem v como um de seus vértices.

2.6.2 Pseudocódigo

O pseudocódigo dessa busca em largura aplicada para a renumeração de vértices de malhas triangulares é mostrado no algoritmo 9. O algoritmo recebe $G = (V, E)$, isto é, recebe um conjunto V de vértices para a ordenação, conectados pelas arestas em E.

Inicialmente, na linha 2, seleciona-se um vértice inicial pseudo-periférico para a numeração. Esse vértice inicial é atribuído a $s(1)$. A numeração de cada vértice é dada pela posição em $S = \{s(1), s(2), \cdots, s(|V|)\}$. Se um vértice v é colocado na posição i em S, então, v receberá a numeração i.

A estrutura de repetição das linhas 5 a 20 é repetida, no máximo, $|V|$ vezes. Nas linhas 6 e 7, atribui-se a $\mathscr{E}$ as estruturas triangulares com $s(k)$ como um de seus vértices, a menos da(s) estrutura(s) triangular(es) com cúpula $s(k-1)$ porque já foi(ram) considerada(s) na iteração anterior. Em particular, como o vértice inicial é atribuído a $s(1)$, somente $Struc(s(1))$ é selecionado na linha 6. Isso para que não ocorresse $Struc(s(0))$ na linha 7, ou seja, tentativa de criação do conjunto com argumento inválido. Na estrutura de repetição das linhas 8 a 18, para cada estrutura triangular $T \in \mathscr{E}$, inserem-se os vértices $u, w \in B(T)$ em S, caso ainda não tenham sido inseridos. Finalmente, quando todos os vértices de V estiverem em S, o algoritmo retorna a renumeração inversa de S.

2.7 Heurística GGPS

A heurística *Generalized GPS* (GGPS) de Wang, Guo e Shi [87, 88] foi proposta para as reduções de largura de banda e de *profile*. Essa heurística, assim como a heurística GPS de Gibbs, Poole e Stockmeyer [26], é dividida em três etapas: escolha de vértices pseudo-periféricos, redução da largura de nível e renumeração. Para um melhor entendimento da heurística GGPS, veja as definições 8 e 10, na seção 1.4, na página 8, para vértice pseudo-periférico e largura de nível, respectivamente.

A heurística GGPS tem duas diferenças principais em relação à heurística GPS. A primeira diferença é que, na heurística GPS, apenas dois vértices pseudo-periféricos são encontrados na etapa de inicialização. Na heurística GGPS, *dois conjuntos* de vértices pseudo-periféricos são encontrados na etapa de inicialização. Seja o grafo $G = (V, E)$ conexo, não ponderado e não direcionado, em que V é o conjunto de vértices e E é o conjunto de arestas. Selecionam-se dois vértices pseudo-periféricos u e w, em que w é o vértice com menor largura de nível do último nível da estrutura de nível de u. Essa escolha é realizada da mesma forma que ocorre no passo de inicialização da heurística GPS. Veja a subseção 2.3.1, na página 19, para detalhes sobre esse passo na heurística GPS.

No primeiro conjunto de vértices, selecionam-se os vértices do último nível da estrutura de nível de u que possuem a mesma excentricidade do vértice u. Nesse conjunto, haverá, pelo menos, um vértice w. Isso porque o vértice u foi selecionado conforme o passo de inicialização da heurística GPS. Com isso, a excentricidade do

Algoritmo 9: Boutora-Takorabet-Ibtiouen-Mezani.

Entrada: triangulação $G = (V, E)$, com vértices em V conectados por arestas em E;
Saída: solução $S = \{s(1), s(2), \cdots, s(|V|)\}$ com a ordem dos vértices;

```
1  início
       // seleciona um vértice na fronteira da triangulação
2      s(1) ← VerticePseudoPeriferico(V);
3      k ← 1; // percorrem-se os vértices adjacentes ao vértice s(k),
       // em que s(k) já foi renumerado
4      i ← 1; // já foram renumerados i vértices
5      enquanto ( i < |V| ) faça
6          ℰ ← Struc(s(k));
7          se ( k > 1 ) então ℰ ← ℰ − (Struc(s(k)) ∩ Struc(s(k − 1)));
8          para cada ( estrutura triangular T ∈ ℰ ) faça
               // {u, w} é a base de T, pois s(k) é a cúpula de cada T
9              {u, w} ← B(T); // T ∈ ℰ
10             se ( u ∉ S ) então
11                 i ← i + 1;
12                 s(i) ← u;
13             fim-se;
14             se ( w ∉ S ) então
15                 i ← i + 1;
16                 s(i) ← w;
17             fim-se;
18         fim-para-cada;
19         k ← k + 1;
20     fim-enquanto;
21     retorna RenumeracaoInversa(S);
22 fim.
```

vértice u é igual às excentricidades de todos os vértices do último nível da estrutura de nível de u.

Para o segundo conjunto de vértices, são selecionados todos os vértices do grafo que possuem o grau e a excentricidade de u, excetuando-se os vértices que não foram selecionados para o conjunto anterior. Apenas o vértice u pertencerá a esse conjunto se nenhum outro vértice tiver essas características.

Com mais vértices iniciais, mais estruturas de nível são geradas. Com isso, pode ser que se obtenham larguras de nível menores que as larguras de nível das estruturas de nível obtidas pela heurística GPS.

A segunda diferença entre as heurísticas GPS e GGPS é que, na etapa de renumeração, em vez de se renumerar os vértices nível a nível da estrutura de nível em ordem crescente de grau como na heurística GPS, renumeram-se os vértices em ordem crescente do somatório dos rótulos dos seus vértices adjacentes já renumerados. Assim como na heurística GPS, a numeração final é invertida na heurística GGPS.

Esta seção é dividida como a seguir. Na subseção 2.7.1, descrevem-se detalhes da etapa da escolha dos vértices pseudo-periféricos da heurística GGPS. Apresenta-se, na subseção 2.7.2, a etapa de redução de largura de nível da heurística GGPS. Na subseção 2.7.3, apresenta-se a etapa de renumeração da heurística GGPS.

2.7.1 Escolha dos vértices iniciais

Nesta etapa, dois conjuntos U e W de vértices pseudo-periféricos são selecionados. Primeiramente, obtém-se um vértice de grau mínimo $u \in V$ e gera-se a estrutura de nível enraizada $\mathscr{L}(u)$. Constroem-se as estruturas de nível enraizadas dos vértices $u' \in L_{\ell(u)}(u)$, em ordem crescente de grau. Se $\ell(u) < \ell(u')$, então, u' é atribuído a u. Repete-se esse processo até que $(\forall u' \in L_{\ell(u)}(u))\ \ell(u) = \ell(u')$. Com isso, o vértice u será o primeiro vértice a compor o conjunto U de vértices pseudo-periféricos. Para o conjunto W de vértices pseudo-periféricos, os autores indicaram que fosse selecionado um vértice w com a menor largura de nível entre os vértices de $L_{\ell(u)}(u)$ e selecionam-se, também de $L_{\ell(u)}(u)$, os vértices com a excentricidade de u ou w. Ocorre que todos os vértices w em $L_{\ell(u)}(u)$ têm excentricidade $\ell(u) = \ell(w)$. Por isso, $W = L_{\ell(u)}(u)$. Finalmente, para o conjunto de vértices pseudo-periféricos U, são selecionados os vértices $u' \in V - W$ que possuírem o mesmo grau e a mesma excentricidade do vértice u.

Apresenta-se o pseudo-código da geração do conjunto U de vértices pseudo-periféricos no algoritmo 10. O algoritmo recebe um grafo $G = (V, E)$. A primeira parte é o algoritmo 2, na página 21.

Algoritmo 10: escolha dos vértices pseudo-periféricos.

Entrada: grafo $G = (V, E)$;
Saída: conjunto de vértices pseudo-periféricos $U \subseteq V$;

```
1 início
2     u ← EscolhaGPS(G = (V, E)); // algoritmo 2, na página 21
3     U ← {u};
4     ℒ(u) ← BuscaEmLargura(u); // estrutura de nível enraizada
5     para cada ( vértice u' ∈ V − {u} − L_ℓ(u)(u) ) faça
6         se ( Grau(G, u') = Grau(G, u) ∧ ℓ(u') = ℓ(u) ) então U ← U ∪ {u'};
7     fim-para-cada;
8     retorna U;
9 fim.
```

Para o conjunto U de vértices pseudo-periféricos, são selecionados os vértices pertencentes a V que possuírem a mesma excentricidade e o mesmo grau que u. Esse passo é realizado na estrutura de repetição das linhas 5 a 7. O algoritmo retorna o conjunto U de vértices pseudo-periféricos. Para o conjunto W de vértices pseudo-periféricos, determina-se $W = L_{\ell(u)}(u)$.

2.7.2 Redução da largura de nível

Na segunda etapa da heurística GGPS [88], procura-se combinar as estruturas de nível enraizadas dos vértices encontrados na etapa anterior, apresentada na subseção 2.7.1. A renumeração dos vértices de um grafo pela estrutura de nível limita a largura de banda da matriz de adjacências A por $\beta(A) \leq 2b(\mathscr{K}(\nu, \cdots)) - 1$ [26]. Com isso, quanto menor a largura de nível de uma estrutura de nível, menor a largura de banda que pode ser obtida.

Considere que as estruturas de nível enraizadas dos vértices pseudo-periféricos obtidos na etapa mostrada na subseção 2.7.1 são $\mathscr{L}(v_1), \mathscr{L}(v_2), \cdots, \mathscr{L}(v_m)$, em que $v_1 = u$ e $m = |U \cup W| = |U| + |W|$, pois $U \cap W = \varnothing$. Associa-se, a cada vértice $v \in V$, uma n-tupla ordenada $(i_1, \cdots, i_{|U|}, i_{|U|+1}, \cdots, i_m)$, em que

i_j, com $j = 1, \cdots, |U|$ são os índices associados ao nível de $\mathscr{L}(v_j)$ que contém $v \in V$. Os índices $\ell(u) - i_r$, com $r = |U| + 1, \cdots, m$, são os índices associados aos níveis de $\mathscr{L}(v_r)$, em ordem decrescente de níveis, que contém v. A n-tupla ordenada $(i_1, \cdots, i_{|U|}, i_{|U|+1}, \cdots, i_m)$ é associada ao vértice v se e, somente se, $v \in \{L_{i_1}(v_1) \cap L_{i_2}(v_2) \cap \cdots \cap L_{i_{|U|}}(v_{|U|}) \cap L_{\ell(u)-i_{(|U|+1)}}(v_{|U|+1}) \cap \cdots \cap L_{\ell(u)-i_m}(v_m)\}$. Os passos do processo que resultarão em uma nova estrutura de nível $\mathscr{K}(\nu, \cdots) = \{K_0(\nu, \cdots), K_1(\nu, \cdots), \cdots, K_{\ell(u)}(\nu, \cdots)\}$ são descritos a seguir.

1. Os vértices com as n-tuplas ordenadas associadas dos níveis na forma $(i, i, \cdots, i)$ são organizados em $K_i(\nu, \cdots)$. Cada vértice v e as arestas incidentes a v são removidas de $G = (V, E)$. Se $V = \varnothing$, então, pare.

2. Após o primeiro passo, tem-se $V = \bigcup_{\iota=1}^{t} \mathscr{C}_\iota$, em que os conjuntos $\mathscr{C}_\iota$ são disjuntos e t é o número de componentes. As componentes são dispostas em ordem decrescente de número de vértices, isto é, $|\mathscr{C}_1| \geq |\mathscr{C}_2| \geq \cdots \geq |\mathscr{C}_t|$.

3. Para cada $\mathscr{C}_\iota$, $\iota = 1, 2, \cdots, t$ e considerando-se $m = |U| + |W|$ e $i = 0, 1, \cdots, k$, em que $k = \ell(u) = \ell(w)$ (logo, k é o número de níveis de $\mathscr{K}(\nu, \cdots)$ decrementado de um), faça:

   - calcule $(n_0, n_1, \cdots, n_k)$, em que n_i é o número de vértices em $K_i(\nu, \cdots)$;
   - gere m n-tuplas ordenadas $(h_0(v_j), h_1(v_j), \cdots, h_k(v_j))$, para $1 \leq j \leq m$, isto é, geram-se:
     - $(h_0(v_1), h_1(v_1), \cdots, h_k(v_1))$,
     - $(h_0(v_2), h_1(v_2), \cdots, h_k(v_2))$,
     - $\cdots$,
     - $(h_0(v_m), h_1(v_m), \cdots, h_k(v_m))$,

     em que $h_i(v_j)$ é n_i mais o número de vértices que $K_i(\nu, \cdots)$ teria se o índice referente a v_j fosse utilizado como a renumeração de nível;
   - encontre $h_{\max}(v_j) = \max_{0 \leq r \leq k} (h_r(v_j) : h_r(v_j) - n_r > 0)$ com $j = 1, 2, \cdots, m$. Em seguida, calcule $h_{\min} = \min(h_{\max}(v_1), h_{\max}(v_2), \cdots, h_{\max}(v_m))$. Dessa maneira, coloque os vértices de $\mathscr{C}_\iota$ nos níveis da estrutura de nível $\mathscr{K}(\nu, \cdots)$ de acordo com o índice associado do nível correspondente a $h_{\min}$. Em ocorrendo mais de um $h_{\max}(v_j)$ mínimo, escolha o primeiro encontrado.

4. Se, no final dos passos desta etapa, o grafo não foi particionado em componentes, então, utiliza-se a estrutura de nível do vértice pertencente a $U \cup W$ que possua a menor largura de nível.

2.7.3 Renumeração

As primeiras etapas da heurística GGPS [88], mostradas nas subseções 2.7.1 e 2.7.2, são baseadas nas etapas correspondentes da heurística GPS, mostradas nas subseções 2.3.1 e 2.3.2. Porém, a etapa de renumeração da heurística GGPS é diferente da etapa de renumeração da heurística GPS. Na heurística GPS, a renumeração é realizada nível a nível da estrutura de nível obtida na etapa de redução de largura nível. Os vértices são renumerados em ordem crescente de grau. Finalmente, essa renumeração é invertida. Na heurística GGPS, os vértices são renumerados de acordo com a soma das numerações dos vértices adjacentes. Essa renumeração pode

ser invertida no final. Para a numeração da heurística GGPS, considere a estrutura de nível $\mathscr{K}(\nu, \cdots)$ montada na etapa anterior.

1. Mostramos aqui, uma pequena variação na descrição mostrada por Wang, Guo e Shi [88]. Considere o vértice pseudo-periférico u escolhido na primeira etapa. Se o somatório da numeração dos vértices adjacentes de u não for a menor entre as numerações dos vértices pseudo-periféricos pertencentes a W, troque o vértice $w \in W$ que possuir o menor somatório das numerações dos vértices adjacentes com o vértice u. Note que, antes de uma heurística para a redução de largura de banda ser executada, há uma numeração original dos vértices, justamente para que, com a renumeração, haja a *redução* de largura de banda. Se essa troca de vértices for realizada, inverta os níveis da estrutura de nível $\mathscr{K}(\nu, \cdots)$ de $K_i(\nu, \cdots)$ por $K_{k-i}(\nu, \cdots)$, com $i = 0, 1, \cdots, \lfloor \frac{k}{2} \rfloor$.

2. Associe o número 1 ao vértice u.

3. Para a renumeração dos vértices pertencentes a $K_j(\nu, \cdots)$, com $i = 0, 1, \cdots, k$, faça os seguintes passos.

   (a) No nível $K_0(\nu, \cdots)$, selecione o vértice com a menor renumeração na iteração corrente. Se houver vértices pertencentes a $K_0(\nu, \cdots)$ e adjacentes a esse vértice que ainda não foram renumerados, renumere esses vértices em ordem crescente do somatório das numerações dos vértices adjacentes. Repita esses passos até que todos os vértices pertencentes a $K_0(\nu, \cdots)$, alcançáveis dessa maneira, sejam renumerados. Se restarem vértices não renumerados em $K_0(\nu, \cdots)$, então, renumere-os em ordem crescente do somatório das numerações.

   (b) Seja $v' \in K_{i-1}(\nu, \cdots)$ o vértice com a menor renumeração do nível $K_{i-1}(\nu, \cdots)$. Se v' possui vértices adjacentes no nível $K_i(\nu, \cdots)$, renumere esses vértices em ordem crescente do somatório das numerações dos vértices adjacentes. Repita esse processo até que todos os vértices de $K_i(\nu, \cdots)$, que possuam adjacências a outros vértices de $K_i(\nu, \cdots)$, sejam numerados.

   (c) Seja $v'' \in K_i(\nu, \cdots)$ o vértice com a menor renumeração na iteração corrente. Se houver vértices pertencentes a $K_i(\nu, \cdots)$ e adjacentes a v'' que ainda não foram numerados, numere esses vértices em ordem crescente do somatório das numerações dos vértices adjacentes. Repita esses passos até que todos os vértices pertencentes a $K_i(\nu, \cdots)$, alcançáveis dessa maneira, sejam numerados.

   (d) Se restarem vértices pertencentes ao nível $K_i(\nu, \cdots)$ ainda não renumerados, numere o vértice com menor somatório das numerações dos vértices adjacentes e volte ao passo (b).

4. A renumeração final é invertida se uma das duas condições a seguir são atendidas.

   (a) Se, no passo 1 desta etapa, o vértice u foi trocado por outro vértice e foi selecionado o índice de um vértice de W na etapa de redução de largura de nível da componente $\mathscr{C}_1$.

   (b) Se, no passo 1 desta etapa, o vértice u não foi trocado por outro vértice e foi selecionado o índice de um vértice de U na etapa de redução de largura de nível da componente $\mathscr{C}_1$.

2.8 Exercícios

1. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.1 após a renumeração pela heurística GPS [26].

2. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.1 após a renumeração pela heurística de Snay [80].

3. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.6 após a renumeração pela heurística de Snay [80].

4. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.1 após a renumeração pela heurística quatro-etapas [40].

5. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.1 após a renumeração pela heurística de Boutora et al. [4].

6. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.6 após a renumeração pela heurística de Boutora et al. [4].

7. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.1 após a renumeração pela heurística GGPS [88].

8. Mostre a largura de banda e o *profile* da matriz correspondente ao grafo mostrado na figura 2.6 após a renumeração pela heurística GGPS [88].

9. Considere a seguinte variação da heurística CMr. Considere um vértice inicial v_1 e a sua estrutura de nível enraizada $\mathscr{L}(v_1)$. Nesta variação do CMr, não se numeram os vértices de cada nível na ordem crescente de grau. Em vez disso, comece a renumeração dos vértices do nível $L_i(v_1)$ por um vértice $v \in L_i(v_1)$ qualquer, para $1 \leq i \leq \ell(v_1)$. Em seguida, numere os demais vértices de $L_i(v_1)$ na ordem da distância em relação ao vértice v, ou seja, numere os vértices de $L_i(v_1)$ por uma busca em largura em v. Essa heurística não garante redução de largura de banda nem redução de *profile* em relação ao CMr. Ainda, essa heurística não garante redução de largura de banda em relação à heurística quatro-etapas, mas pode apresentar redução de *profile* em relação à heurística quatro-etapas.

   (a) Mostre as diferenças dessa variação em relação à heurística quatro-etapas.

   (b) Qual heurística necessita de mais capacidade de armazenamento: quatro-etapas ou essa variação do CMr?

   (c) Mostre que a complexidade no pior caso dessa heurística é $O(|V| + |E|)$.

   (d) Qual heurística é mais rápida: quatro-etapas ou essa variação do CMr?

   (e) Apresente uma variação da heurística CMr.

Capítulo 3

Heurísticas baseadas em meta-heurísticas

3.1 Introdução

Neste capítulo, são apresentadas heurísticas baseadas em meta-heurísticas para as reduções de largura de banda e de *profile* de matrizes. Descrevem-se as heurísticas *Node Centroid with Hill Climbing* (NC-HC) [55] e *Fast Node Centroid with Hill Climbing* (FNC-HC) [55] porque são exemplos de heurísticas que utilizam o *Hill Climbing* como busca local e porque foram testadas também para a redução de largura de banda de matrizes assimétricas.

Apresenta-se a heurística de Kaveh e Sharafi [43] (ACO-KS) como exemplo de heurística baseada na meta-heurística Otimização por Colônia de Formigas. A heurística baseada em *Variable Neighbourhood Search* (VNS-*band*) de Mladenovic et al. [66] é apresentada porque, segundo os próprios autores, é a provável heurística no estado da arte para o problema de redução de largura de banda de matrizes. Os testes realizados por Mladenovic et al. [66] foram no conjunto de instâncias de matrizes esparsas Harwel-Boeing (*http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/*). A heurística de Kaveh e Sharafi [44] (CSS-*band*) é apresentada, pois é um exemplo de heurística baseada em uma meta-heurística recente: *Charged System Search* [46].

Nas publicações em que foram apresentadas, as heurísticas mostradas neste capítulo foram estudadas para a redução da largura de banda de matrizes por seus autores. O CSS-*band* foi estudado também para a redução de *profile* por Kaveh e Sharafi [44].

Este capítulo é dividido como a seguir. Na seção 3.2, apresentam-se as heurísticas NC-HC e FNC-HC [55]. Na seção 3.3, apresenta-se a heurística ACO-KS [43]. Descreve-se a heurística VNS-*band* [66] na seção 3.4. A heurística CSS-*band* [44] é apresentada na seção 3.5.

3.2 Heurísticas NC-HC e FNC-HC

Proposto por Lim, Rodrigues e Xiao [55], a heurística *Node Centroid with Hill Climbing* (NC-HC) baseia-se no ajuste de numeração para a busca global e no *Hill Climbing* para a busca local. Foi proposto com o intuito de ser uma heurística rápida, simples e que gerasse bons resultados.

De forma geral, a heurística NC-HC inicia por uma solução gerada pela busca em largura. Altera-se essa numeração inicial pelo ajuste de numeração em relação a um determinado vértice e seus vértices adjacentes, denominado de *Node Centroid* (NC). Finalmente, aplica-se a busca local pelo *Hill Climbing* para obter um mínimo local.

Considere o grafo $G = (V, E)$. Na heurística NC-HC, uma solução S é o conjunto das numerações dos vértices, ou seja, $S = \{s(1), s(2), \cdots, s(|V|)\}$, em que $s(i)$ é a numeração do vértice i da solução S.

A heurística é dividida em três etapas, que são descritas nas subseções seguintes. Na subseção 3.2.1, apresentam-se a etapa de inicialização da heurística NC-HC e a etapa do NC. Descreve-se, na subseção 3.2.2, a etapa da aplicação da busca local pelo *Hill Climbing*. A heurística NC-HC completa é descrita na subseção 3.2.3. Na subseção 3.2.4, apresenta-se a heurística FNC-HC.

3.2.1 Inicialização e ajuste de numeração

Na primeira etapa da heurística NC-HC, Lim, Rodrigues e Xiao [55] utilizaram uma solução inicial gerada pela numeração da estrutura de nível enraizada de um vértice aleatório. Para a montagem dessa estrutura de nível enraizada, utilizou-se a busca em largura, iniciando-se por um vértice aleatório. Na segunda etapa, associa-se um peso $\mathfrak{p}$ a cada vértice pertencente a V. Em seguida, renumeram-se os vértices pela ordem crescente de peso $\mathfrak{p}$ de cada vértice.

Assim como na etapa de busca local da heurística VNS-*band*, mostrada na subseção 3.4.3, na página 58, a heurística NC utiliza a noção de *vértice crítico* de $G = (V, E)$ (veja definição 13 da seção 1.4, na página 9). Para essa etapa, define-se, para cada vértice $v \in V$, o diâmetro $diam(s(v)) = \max_{\{u,v\} \in E} |s(u) - s(v)|$, chamado de *diâmetro do vértice* v veja a definição 1, na seção 1.4, na página 7). Note que, para matrizes simétricas, $diam(s(i)) = \beta_i(A)$. Porém, isso não acontece para matrizes assimétricas. Como as heurísticas NC-HC e FNC-HC foram testadas em matrizes simétricas e assimétricas, Lim, Rodrigues e Xiao [55] definiram $diam(s(v))$, com $v \in V$, considerando-se todas as arestas de E.

Um vértice $v \in V$ é um vértice crítico de $G = (V, E)$ se existe um vértice $u \in V$ adjacente a v, em que $\{u, v\} \in E$ é uma *aresta crítica* de $G = (V, E)$ ou se $diam(s(v)) = \beta(A_s)$, em que A_S é a matriz obtida pela numeração S. Uma aresta $\{u, v\} \in E$ é crítica se $|s(u) - s(v)| = \beta(A_S)$ (veja definições 12 e 13 da seção 1.4 na página 9).

Na heurística NC-HC, um vértice v é dito λ-crítico se $diam(s(v)) \geq \lambda \cdot \beta(A_S)$, com $\lambda \in [0, 1]$. Ainda, para o grafo $G = (V, E)$, definem-se os vértices $\lambda-$adjacentes de v por $V_\lambda(v) = Adj(G, v) \cap \{u : (\{v, u\} \in E)\ |s(u) - s(v)| \geq \lambda \cdot \beta(A_S)\}$. Com o conjunto $\lambda-$adjacentes, têm-se o *feixe* do vértice v, que é definido por $f_\lambda(v) = V_\lambda(v) \cup \{v\}$ e v é o vértice centroide do feixe $f_\lambda(v)$.

Na etapa de inicialização e ajuste de numeração, deseja-se diminuir o diâmetro dos vértices $\lambda-$críticos, com valores próximos a 1. Procura-se realizar essa diminuição trocando-se os vértices centroides de seus respectivos feixes. Essa troca é realizada ao se associar um peso $\mathfrak{p}$ a cada vértice $v \in V$ para, depois, numerar os vértices em ordem crescente desse peso. O peso $\mathfrak{p}(v)$ é definido por $\mathfrak{p}(v) = \frac{\sum_{u \in f_\lambda(v)} s(u)}{|f_\lambda(v)|}$.

Mostra-se a sub-rotina *Node Centroid* no algoritmo 11. O algoritmo recebe um grafo $G = (V, E)$, a numeração S e o parâmetro λ. Na estrutura de repetição *para* das linhas 2 a 5, os vetores $\mathfrak{p}$ e c são inicializados, em que $c(i)$ é o número de vértices pertencentes a $f_\lambda(v)$. Na estrutura de repetição *para cada* das linhas 6

a 13, atualizam-se os vetores $\mathfrak{p}$ e c de cada vértice que seja um vértice $\lambda-$crítico. Atualiza-se o valor de $\mathfrak{p}$ pela razão de $\mathfrak{p}$ por c para cada vértice $v \in V$ na estrutura de repetição *para* da linha 14. Em seguida, na linha 15, renumeram-se o vértices em ordem crescente do peso $\mathfrak{p}$. O algoritmo retorna a renumeração S'.

Algoritmo 11: *Node Centroid.*

Entrada: grafo $G = (V, E)$; renumeração S; parâmetro λ;
Saída: renumeração S';

```
1  início
2      para ( i ← 1; i ≤ |V|; i ← i + 1 ) faça
3          p(i) ← s(i); // p(i) é o peso associado a cada vértice i
4          c(i) ← 1; // c(i) é o número de vértices do feixe do vértice i
5      fim-para;
6      para cada ( aresta {u, v} ∈ E ) faça
7          se ( |s(u) − s(v)| ≥ λ · β(A_S) ) então
8              p(u) ← p(u) + s(v);
9              c(u) ← c(u) + 1;
10             p(v) ← p(v) + s(u);
11             c(v) ← c(v) + 1;
12         fim-se;
13     fim-para-cada;
14     para ( i ← 1; i ≤ |V|; i ← i + 1 ) faça p(i) ← p(i)/c(i);
       // renumeram-se os vértices pertencentes a V em ordem
15     S' ← NumeraCrescente(S); // crescente do peso p
16     retorna S';
17 fim.
```

3.2.2 Busca local por *Hill Climbing*

Nesta etapa, seleciona-se um conjunto de vértices críticos do grafo para que suas numerações sejam trocadas. O conjunto E_c de arestas críticas de $G = (V, E)$ é definido por $E_c = \{\{u, v\} \in E \ : \ |s(u) - s(v)| = \beta(A_s)\}$. O conjunto de vértices críticos V_c de $G = (V, E)$ é todo vértice de arestas de E_c. Para a aplicação da busca local, define-se a vizinhança reduzida para a troca de numeração de v como $\mathrm{N}(v, S)$ do vértice crítico v. $\mathrm{N}(v, S)$ é definido como $\mathrm{N}(v, S) = \{u : (\{u, v\} \in E_c) \ |med(v) - s(u)| < |med(v) - s(v)|\}$ e $med(v) = \left\lfloor \frac{s_{max}(v)+s_{min}(v)}{2} \right\rfloor$, em que $s_{\max}(u) = \max\limits_{\{u,u'\}\in E}(s(u'))$ e $s_{\min}(u) = \min\limits_{\{u,u'\}\in E}(s(u'))$. Isso significa que, entre os vértices críticos, $\mathrm{N}(v, S)$ é o conjunto de vértices que têm diferença de sua numeração média de v em relação à numeração de u menor que a diferença da numeração média de v pela numeração de v. O conjunto de vértices para a aplicação da busca local, ou o conjunto de vizinhanças reduzidas para a troca de numeração em relação a solução S, é definido por $N(S) = \bigcup\limits_{v\in V_c} \mathrm{N}(v, S)$.

Como a etapa da busca local por *Hill Climbing* é depois da etapa de ajuste de numeração, a solução S' é a solução corrente. Com $N(S')$, forma-se um conjunto de vértices apropriados para a troca. A busca local é aplicada no conjunto $N(S')$. Troca-se a numeração do vértice crítico v com cada numeração dos vértices

$u \in \mathrm{N}(v, S') \subseteq N(S')$ até que uma melhora em relação à solução corrente S' seja constatada. Com isso, gera-se a solução S''.

Verifica-se se uma melhora é realizada quando um vértice crítico se torna um vértice não crítico. Para isso, considere o *valor crítico*

$$\Gamma_{S'}(v) = \begin{cases} 0, & \text{se } diam_{S'}(s(v)) < \beta(A_{S'}) \\ 1, & \text{se } diam_{S'}(s(v)) = \beta(A_{S'}) \\ 2, & \text{se } diam_{S''}(s(v)) > diam_{S'}(s(v)) \end{cases},$$

em que $diam_{S'}(s(v))$ é o diâmetro do vértice v na solução S' (veja comentário da definição 1, seção 1.4, na página 7) e S'' é a solução gerada pela troca da numeração do vértice v. Se $\Gamma(v) = 1$, então, v é um vértice crítico. Note que, se $\Gamma(v) = 2$, então, a troca da numeração do vértice v aumentou o $diam_{S'}(s(v))$ e isso pode aumentar a largura de banda, o que não se deseja. Seja $u \in \mathrm{N}(v, S')$. Uma melhora em relação à solução corrente é constatada se $\Gamma_{S''}(u) \leq \Gamma_{S'}(u)$, $\Gamma_{S''}(v) \leq \Gamma_{S'}(v)$ e $\Gamma_{S''}(u) + \Gamma_{S''}(v) < \Gamma_{S'}(u) + \Gamma_{S'}(v)$.

Mostra-se a etapa de busca local por *Hill Climbing* no algoritmo 12. O algoritmo recebe um grafo $G = (V, E)$ e a numeração S'. Na linha 3, inicializa-se a variável lógica *houveMelhora*. Na estrutura de repetição *para cada* das linhas 6 a 17, selecionam-se todos os vértices críticos do grafo. Em seguida, para cada vértice crítico v, selecionam-se vértices $u \in \mathrm{N}(v, S'')$ e troca-se a numeração de u com v, gerando-se a solução auxiliar S_{aux}. Com a solução S_{aux}, identifica-se se uma melhora em relação à solução corrente é gerada. Isso é verificado na condição da linha 10. Caso essa condição seja satisfeita, troca-se a numeração de u com v e atualiza-se a solução corrente S''. O algoritmo para quando melhorias em relação à solução corrente não forem identificadas. O algoritmo retorna a renumeração S''. Se em nenhuma iteração do algoritmo 12 a condição da linha 10 for satisfeita, então, nenhuma melhoria em relação à solução S' foi realizada. Com isso, o algoritmo retorna a renumeração S', que foi atribuída à numeração S'' na linha 2.

3.2.3 Pseudocódigo para a heurística NC-HC

A heurística NC-HC é mostrada no algoritmo 13. O algoritmo recebe o grafo $G = (V, E)$, a numeração S, o número máximo de iterações i_{max} e o número máximo de iterações para a etapa do ajuste de numeração NC_{max}. Na linha 4, modifica-se a numeração original S pela busca em largura. Com isso, gera-se a solução inicial. Altera-se a numeração S pelo NC em NC_{max} vezes.

Lim, Rodrigues e Xiao [55] constataram que aplicar a busca local em apenas algumas iterações e não em todas torna a heurística mais rápida e proporciona boas soluções. Para isso, utiliza-se a estrutura condicional da linha 7 para aplicar a busca local em metade das iterações. O algoritmo retorna a renumeração com a menor largura de banda encontrada.

3.2.4 Heurística FNC-HC

A heurística *Fast Node Centroid with Hill Climbing* (FNC-HC) é uma variação da heurística NC-HC de Lim, Rodrigues e Xiao [55]. Na heurística FNC-HC, Lim, Rodrigues e Xiao [55] fizeram com que alguns parâmetros fossem definidos e ajustados automaticamente, de acordo com o tamanho da matriz do problema. Com isso, uma redução do custo computacional pode ser alcançada em relação à heurística NC-HC.

Algoritmo 12: *Hill Climbing.*

Entrada: grafo $G = (V, E)$; numeração S';
Saída: renumeração S'';

```
1  início
2      S'' ← S';
3      houveMelhora ← verdadeiro;
4      enquanto ( houveMelhora ) faça
5          houveMelhora ← falso;
6          para cada ( vértice v ∈ V ) faça
7              se ( Γ(v) = 1 ) então
8                  para cada ( vértice u ∈ N(v, S'') ) faça
                       // troca a numeração de v pela numeração de u
                       // e gera-se a solução auxiliar S_aux
9                      S_aux ← TrocaNumeracao(s(v), s(u), S'');
10                     se ( (Γ_{S_aux}(u) ≤ Γ_{S''}(u)) ∧ (Γ_{S_aux}(v) ≤
                       Γ_{S''}(v)) ∧ ((Γ_{S_aux}(v) + Γ_{S_aux}(u)) < (Γ_{S''}(u) + Γ_{S''}(v))) )
                       então
                           // atualiza-se a solução S'' com a troca
                           // das numerações realizadas na linha 9
11                         S'' ← S_aux;
                           // realiza-se a busca local enquanto
                           // melhorias são constatadas
12                         houveMelhora ← verdadeiro;
13                         break 2; // vai para o teste da linha 4
14                     fim-se;
15                 fim-para-cada;
16             fim-se;
17         fim-para-cada;
18     fim-enquanto;
       // se a condição da linha 10 não for satisfeita em nenhuma
       // iteração, retorna-se S', pois nenhuma melhoria foi
       // encontrada
19     retorna S'';
20 fim.
```

Para a heurística FNC-HC, Lim, Rodrigues e Xiao [55] definiram, empiricamente, bons valores para o parâmetro λ. O parâmetro λ é explicado na subseção 3.2.1, na página 44. Segundo Lim, Rodrigues e Xiao [55], nas instâncias testadas, os melhores resultados foram obtidos pela heurística FNC-HC com $\lambda = 0{,}95$, para a redução de largura de banda em relação aos resultados obtidos com outros valores de λ. Os melhores tempos de execução foram obtidos pela heurística FNC-HC com $\lambda = 1$ em relação aos resultados obtidos com outros valores de λ.

Algoritmo 13: NC-HC.

Entrada: grafo $G = (V, E)$; numeração S; número máximo de iterações i_{max} da heurística NC-HC; número máximo de iterações NC_{max} para a sub-rotina $NodeCentroid$;
Saída: S renumerado;

```
1  início
2      S_cor ← S;
3      para ( i ← 1; i ≤ i_max; i ← i + 1 ) faça
           // constrói-se uma solução inicial pela busca em largura
4          S ← NumeraBuscaEmLargura(S);
5          para ( j ← 1; j ≤ NC_max; j ← j + 1 ) faça
6              S' ← NodeCentroid(S); // altera-se a numeração por NC
               // altera-se a numeração por HC
7              se ( (j mod 2) = 1 ) então S'' ← HillClimbing(S');
8          fim-para;
           // A_{S_cor} e A_{S''} são as formações das matrizes de
           // adjacências de G = (V, E) pelas numerações S_Cor corrente
           // e S'', respectivamente
9          se ( β(A_{S_cor}) < β(A_{S''}) ) então S_cor ← S'';
10     fim-para;
11     retorna S_cor;
12 fim.
```

3.3 Heurística de Kaveh-Sharafi por Otimização por Colônia de Formigas

A meta-heurística que utiliza o conceito de colônia de formigas (CF) para problemas de otimização foi proposta por Dorigo, Maniezzo e Colorni [12]. Essa meta-heurística consiste em simular o comportamento real de uma colônia de formigas, em que a construção do caminho até um alimento se dá de forma coletiva. Quando uma formiga traça um rota até um alimento, ela deposita um certa quantidade de feromônio nesse caminho. Quando outra formiga encontra esse caminho e o segue, essa formiga deposita seu próprio feromônio, reforçando o caminho. Trabalhos como de Lim et al. [51] aplicaram a Otimização por CF na redução de largura de banda. Kaveh e Sharafi [43] propuseram uma heurística por Otimização por CF e busca local utilizando a estrutura *Shortest Route Tree* (SRT) para a redução de largura de banda. Essa estrutura é explicada na subseção 2.5.1, na página 32.

Para um problema de otimização em um grafo, pode-se dividir a Otimização por CF em duas etapas: a construção da solução pelas formigas e a atualização do feromônio. Segundo Kaveh e Sharafi [43], uma forma de inicializar o feromônio contido nas arestas é configurá-lo para um valor um pouco maior do que a quantidade esperada de feromônio depositado pelas formigas em uma iteração. A opção de iniciar com esses valores deve-se ao fato de que, se os valores iniciais forem muito baixos, então, a construção da solução será influenciada pelo primeiro caminho traçado pelas formigas. Da mesma forma, se os valores do feromônio forem muito altos, muitas iterações serão perdidas, esperando-se que o feromônio evapore o suficiente para que as formigas possam construir a solução depositando os próprios feromônios.

Após as definições dos valores dos parâmetros iniciais, como, por exemplo, o valor do feromônio, as formigas iniciam a construção dos caminhos no grafo. Inicialmente, as formigas são colocadas arbitrariamente nos vértices. A cada iteração, a formiga k decide qual vértice visitar por meio da *regra de escolha probabilística*. A probabilidade da formiga k, posicionada no vértice i, escolher visitar o vértice j, é dada pela regra da escolha probabilística

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^{\omega_\tau} \cdot [\eta_{ij}]^{\omega_\eta}}{\sum_{j \in P_i^k} [\tau_{ij}]^{\omega_\tau} \cdot [\eta_{ij}]^{\omega_\eta}}, & \text{se } j \in P_i^k \\ 0, & \text{se } j \notin P_i^k \end{cases}. \tag{3.3.1}$$

Na equação (3.3.1), τ_{ij} é o valor do feromônio na aresta que liga o vértice i ao vértice j e η_{ij} é o coeficiente da matriz de heurística η. Os coeficientes η_{ij} dependem do problema. No problema do caixeiro viajante, por exemplo, frequentemente, atribui-se $\eta_{ij} = 1/d_{ij}$, em que d_{ij} é a distância entre as cidades i e j. Com os parâmetros ω_τ e ω_η, determinam-se a relevância de τ_{ij} e η_{ij}, respectivamente, na equação (3.3.1). P_i^k é o conjunto de vértices que a formiga k, posicionada no vértice i, ainda não visitou.

Depois que todas as formigas construíram os respectivos caminhos no grafo, ocorre a atualização do feromônio. Essa etapa é dividida em dois passos. O primeiro consiste na evaporação do feromônio de todas as arestas do grafo com taxa ρ ($0 \leq \rho < 1$),

$$\tau_{ij} \leftarrow \tau_{ij} \cdot (1 - \rho), \ \forall (i,j) \in E. \tag{3.3.2}$$

O segundo passo é o depósito do feromônio no caminho de menor custo da iteração corrente, que é atualizado por

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \ \forall (i,j) \in L, \tag{3.3.3}$$

em que $\Delta\tau_{ij}^k$ é o feromônio adicionado a τ_{ij} pela formiga k,

$$\Delta\tau_{ij}^k = \begin{cases} 1/R, & \text{se } (i,j) \in S^k \\ 0, & \text{se } (i,j) \notin S^k \end{cases}, \tag{3.3.4}$$

m é o número de formigas e L é o conjunto das arestas que compõem o caminho de menor custo. O valor atribuído a R é o custo do melhor caminho percorrido no grafo na iteração corrente. S^k é a solução da formiga k na iteração corrente e S é o conjunto de soluções válidas, dependendo do problema.

A heurística de Kaveh e Sharafi [43] para a redução de largura de banda utiliza dois grafos: o grafo real $G = (V, E)$ e um pseudo-grafo $G' = (V', E')$. $G = (V, E)$ é o grafo que se deseja renumerar e obter-se a redução de largura de banda da matriz que o representa. O pseudo-grafo possui o mesmo número de vértices que o grafo real. Todos os vértices do pseudo-grafo são adjacentes aos vértices do grafo real por meio de *pseudo-arestas*. Coloca-se, inicialmente, uma formiga em cada vértice do pseudo-grafo para a construção da solução.

A heurística de Kaveh e Sharafi [43] é dividida em cinco etapas, que são descritas nas subseções seguintes. Na subseção 3.3.1, apresentam-se os cálculos preliminares e inicializações necessárias para a heurística. Descreve-se, na subseção 3.3.2, a numeração realizada pela heurística. Apresenta-se, na subseção 3.3.3, a etapa da busca local. Na subseção 3.3.4, descreve-se a atualização das informações durante a heurística. Apresenta-se, na subseção 3.3.5, a etapa da atualização do feromônio no grafo.

3.3.1 Primeira etapa: cálculos preliminares

Considere A uma matriz simétrica que se deseja reduzir a largura de banda. Nesta etapa, valores e dados iniciais são estabelecidos. Os parâmetros ω_τ, ω_η, ρ e o critério de parada são definidos. Segundo Kaveh e Sharafi [43], para esta heurística, valores dos parâmetros $\omega_\tau = 1,5$ e $\omega_\eta = 2,5$ produzem bons resultados.

Os valores dos feromônios iniciais serão a razão de um pela maior largura de banda possível. O maior valor de largura de banda de uma matriz com $|V|$ vértices é $|V| - 1$. Com isso, todos os coeficientes das matrizes τ e η recebem, inicialmente, $1/(|V| - 1)$. Os coeficientes da matriz τ permanecerão inalterados até a etapa de atualização do feromônio, descrita na subseção 3.3.5.

Em relação à matriz heurística η, ajustes devem ser realizados e depois permanecerá inalterada. Para isso, utiliza-se a estrutura SRT do grafo real, iniciando-se em um vértice pseudo-periférico. Considere que v_0 é esse vértice inicial. Com os valores da matriz heurística, deseja-se fazer com que as formigas escolham mais os vértices de primeiros contornos da estrutura SRT do que os vértices que constam nos últimos contornos. Para isso, considere $I = \{l, l+1, \cdots, |C_l(v_0)|+1\}$, em que l é índice do contorno corrente. Considere, também, $J = \{j \ : \ v_j \in C_l(v_0)\}$. Adiciona-se $1/(2 \cdot |C_l(v_0)|)$ a cada coeficiente η_{ij}, em que $i \in I$ e $j \in J$. Entretanto, para o vértice inicial, considera-se $I = \{0\}$. Por exemplo, no contorno $C_0(v_0)$, soma-se $1/(2 \cdot |C_0(v_0)|)$ ao coeficiente η_{00}. Como há apenas o vértice inicial v_0 no contorno $C_0(v_0)$, então, $|C_0(v_0)| = 1$. Logo, $\eta_{00} = 1/(|V|-1)+1/2$. Como segundo exemplo, para o contorno $C_1(v_0)$, com $I = \{1, \cdots, |C_1(v_0)| + 1\}$ e $J = \{j \mid v_j \in C_1(v_0)\}$, adiciona-se $1/(2 \cdot |C_1(v_0)|)$ para todas as entradas η_{ij}, em que $i \in I$ e $j \in J$.

3.3.2 Segunda etapa: construção da solução

Essa etapa é subdividida em duas fases. Na primeira, as formigas escolhem qual vértice visitar, periodicamente, por meio da equação (3.3.1). Uma formiga k, posicionada em um vértice v_i do pseudo-grafo, atribui a sua numeração a um vértice do grafo real que ainda não foi escolhido por nenhuma outra formiga, ou seja, uma formiga k só poderá numerar o vértice v_j se $v_j \in P_i^k \subseteq V$ para o grafo $G = (V, E)$. Após a formiga k ter percorrido a pseudo-aresta que liga um vértice do pseudo-grafo ao vértice escolhido do grafo real, a formiga $k + 1$ seleciona o seu vértice.

A segunda fase consiste na montagem da lista de atribuição. Se o vértice v_j é selecionado por uma formiga k proveniente do vértice v_i do pseudo-grafo, então, $s(j) \leftarrow s(i)$. Isso significa que a númeração de v_i é atribuída ao vértice v_j do grafo real, possibilitando ao final da iteração o cálculo da largura de banda da matriz. A largura de banda é o custo da solução.

3.3.3 Terceira etapa: busca local

Depois do cálculo do custo da solução corrente, é realizada a busca local para melhorar a qualidade da solução. A busca local requer um custo computacional adicional. Entretanto, em casos em que o tempo computacional não é um fator crucial, a busca local pode ajudar a heurística a convergir para a solução ótima (em relação ao vértice pseudo-periférico v_0). Nesta heurística, a busca local é realizada ao trocar duas formigas de lugar em cada contorno da estrutura SRT. Trocam-se as formigas que estiverem posicionadas nos vértices que possuírem as maiores larguras de banda em cada contorno da estrutura SRT. Considerando-se as formigas k_1 e k_2 posicionadas nos respectivos vértices v_{j_1} e v_{j_2} no contorno $C_l(v_0)$ da estrutura SRT

com larguras de banda $\beta_{j_1}(A)$ e $\beta_{j_2}(A)$ maiores que as demais larguras de banda do vértices pertencentes ao contorno $C_l(v_0)$, atribui-se $s(j_2) \leftarrow s(i_1)$ e $s(j_1) \leftarrow s(i_2)$. Em seguida, calcula-se $\beta(A)$. Se houver diminuição da largura de banda, é aplicada a busca local no próximo contorno da estrutura SRT. Caso contrário, a troca de posição das formigas não é realizada.

3.3.4 Quarta etapa: atualização das informações

Nesta etapa, utilizam-se as informações obtidas nas segunda e terceira etapas. Com isso, identifica-se, em cada contorno da estrutura SRT, o vértice que possui a menor largura de banda juntamente com seus vértices adjacentes e, também, selecionam-se as respectivas formigas. Em seguida, seleciona-se a formiga associada ao vértice de menor largura de banda. Então, todos esses dados são armazenados.

3.3.5 Quinta etapa: atualização do feromônio

Nesta etapa, as formigas depositam os feromônios nas pseudo-arestas de acordo com os dados provenientes da quarta etapa. A etapa de atualização do feromônio consiste em duas fases.

A primeira fase corresponde à evaporação do feromônio. A evaporação é dada pela atribuição (3.3.2). Segundo Kaveh e Sharafi [43], o valor adequado de ρ depende do número total de iterações definido de acordo com o critério de parada e do valor do feromônio inicial.

A segunda fase consiste no depósito de feromônio, que ocorre de acordo com a atribuição (3.3.3) e a equação (3.3.4). Primeiramente, deposita-se o feromônio nas pseudo-arestas associadas aos vértices associados à menor largura de banda, no valor de $1/(m \cdot \beta(A))$, em que $m = |V|$ é o número de formigas. Em seguida, as formigas associadas aos vértices de menor largura de banda, uma a cada contorno da estrutura SRT, depositam os seus feromônios nas respectivas pseudo-arestas. O valor depositado é $1/(m \cdot \beta_j(A))$. Então, inicia-se outra iteração. O critério de parada da heurística depende dos parâmetros especificados *a priori*, como exemplos, um limite para o número de iterações ou uma largura de banda menor que um determinado valor.

3.4 Heurística com busca em vizinhança variável

A meta-heurística *Variable Neighbourhood Search* (VNS), foi proposta por Mladenović e Hansen [65]. Divide-se a meta-heurística VNS em três fases: perturbação, busca local e troca de vizinhança.

Inicialmente, o raio k da vizinhança da solução inicial S é atualizado como k_{min}. Geram-se perturbações para tentar escapar dos vales que os mínimos locais pertencem. Na etapa de perturbação, uma solução S' é encontrada dentro da vizinhança $N_{k_{min}}(S)$.

Há a tentativa de se alcançar um ótimo local por meio de busca local. Realiza-se uma exploração dentro do raio da vizinhança corrente. Uma busca local é aplicada na solução S' para encontrar uma solução S''.

Na terceira etapa, verifica-se se a solução encontrada S'' é melhor que a solução corrente S. Caso seja, S'' passará a ser a solução corrente e k será atualizado novamente para k_{min}. Caso contrário, o raio da vizinhança k é incrementado de um k_{step}, que é ajustado de acordo com o problema.

Repetem-se esses passos até que um critério de parada seja satisfeito. Um número máximo de iterações ou um tempo máximo de processamento são exemplos de critério de parada.

Seja C_S o conjunto de soluções possíveis para um determinado problema de otimização. A vizinhança $N_k(S)$, com $S \in C_S$, é o conjunto de soluções vizinhas a S, dentro do raio k. Inicia-se por uma solução S e a meta-heurística VNS encontra, por meio de busca local, um $S'' \in N_k(S)$. Troca-se a solução corrente S e $N_k(S)$ por S'' e $N_k(S'')$ se e, somente se, o custo de S'' é melhor que o custo de S. Caso a solução S'' não seja melhor que a solução corrente S, então, aumenta-se o raio de abrangência k de $N_k(S)$ e realiza-se uma nova busca.

Exemplifica-se, na figura 3.1, a busca da meta-heurística VNS por soluções dentro das estruturas de vizinhança. Inicia-se pela solução S e realiza-se a exploração da vizinhança até o raio k, encontrando-se a solução S_1''. Como no exemplo a solução S_1'' não apresentou melhoria em relação à solução corrente S, a solução S continua como a solução corrente. Em seguida, aumenta-se o raio de $N_k(S)$ para $N_{k+1}(S)$. Com isso, encontra-se a solução S_2'', que representa melhoria em relação à solução S; portanto, S_2'' passa a ser a solução corrente. A busca, então, passa a ser na vizinhança $N_k(S_2'')$. Esses passos são repetidos até que um critério de parada seja satisfeito.

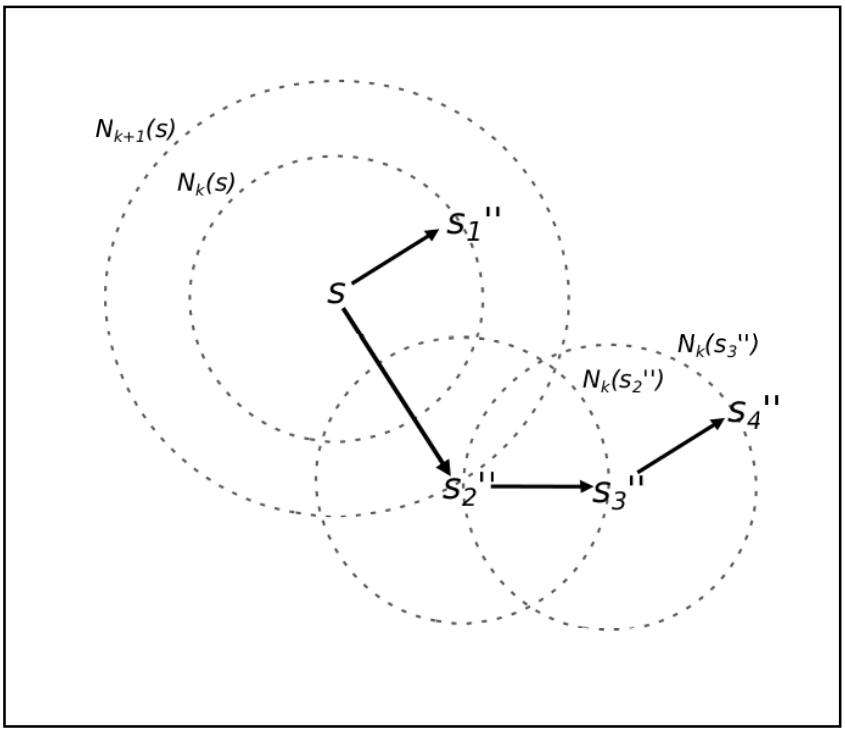

Figura 3.1: Exemplo de busca nas estruturas de vizinhanças pela meta-heurística VNS.

Na heurística *Variable Neighbourhood Search for bandwidth reduction* (VNS-*band*), uma solução S é o conjunto das numerações dos vértices, ou seja, $S = \{s(1), s(2), \cdots, s(|V|)\}$, em que $s(i)$ é a numeração do vértice i da solução S. Na heurística VNS-*band*, uma solução inicial das numerações dos vértices é cons-

truída por uma busca em largura aleatória, iniciando-se por um vértice aleatório. Com a busca em largura aleatória, gera-se uma estrutura de nível enraizada de um vértice aleatório. Com isso, a cada nível da estrutura de nível enraizada o primeiro vértice a ser renumerado é escolhido aleatoriamente. Renumeram-se os vértices de cada nível dessa estrutura de nível enraizada iniciando-se desse vértice aleatório. Os passos da heurística VNS, perturbação, busca local e troca de vizinhança, são seguidos até que o critério de parada seja satisfeito. Segundo Mladenovic et al. [66], a heurística VNS-*band* é, ou pelo menos era até a data de publicação do artigo, a heurística no estado da arte em heurísticas para a redução de largura de banda. Os testes foram realizados na coleção de matrizes esparsas Harwell-Boeing (http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/). As heurísticas, principalmente as baseadas em meta-heurísticas, são bastante dependentes das instâncias em que são aplicadas. Provavelmente, essa heurística é uma das que mais reduzem a largura de banda. Ainda, entre as heurísticas baseadas em meta-heurísticas, é uma das mais rápidas.

As quatro etapas da heurística VNS-*band* são descritas nas subseções seguintes. Na subseção 3.4.1, descreve-se o passo de inicialização. A etapa de perturbação é descrita na subseção 3.4.2. A busca local é explicada na subseção 3.4.3. Descreve-se, na subseção 3.4.4, a etapa da permutação de vizinhança. Finalmente, na subseção 3.4.5, explica-se a heurística em detalhes.

3.4.1 Inicialização

A primeira etapa da heurística VNS-*band* consiste em construir uma solução inicial das numerações dos vértices de $G = (V, E)$. Para isso, constrói-se a estrutura de nível enraizada de um vértice inicial aleatório v. A renumeração é realizada em cada nível da estrutura de nível enraizada $\mathscr{L}(v)$. A renumeração em cada nível $L_i(v)$ é iniciada por um vértice aleatório. Quando todos os vértices do nível $L_i(v)$ estiverem numerados, renumera-se o nível $L_{i+1}(v)$ e assim sucessivamente até $L_{\ell(v)}(v)$.

Considere, para essa etapa, que o grafo $G = (V, E)$ é representado por uma lista de adjacências $E = \{E(1), \cdots, E(|V|)\}$, em que $E(i) = \{e(i,1), \cdots, e(i, |E(i)|)\}$ representa os vértices adjacentes ao vértice i. A heurística inicia por um vértice v aleatório. O vértice v recebe a numeração 1, ou seja, $s(v) = 1$, e constrói-se a estrutura de nível enraizada $\mathscr{L}(v)$. Renumeram-se os vértices por nível da estrutura de nível enraizada, iniciando-se por um vértice aleatório em cada nível $L_i(v)$, para $1 \leq i \leq |L_{\ell(v)}(v)|$. Mladenovic et al. [66] denominaram esse procedimento de busca em largura aleatória. Para um nível $L_i(v)$ com vértices u_j, com $j = 1, \cdots, |L_i(v)|$, escolhe-se aleatoriamente um índice j^* no intervalo $[1, |L_i(v)|]$, para que u_{j^*} seja o primeiro vértice do nível corrente a ser renumerado. Por exemplo, caso o nível da estrutura de nível enraizada seja $L_1(v)$ e o vértice u seja o primeiro vértice a ser renumerado, então, u deve receber o número 2, ou seja, $s(u) \leftarrow 2$ (porque o vértice $v \in L_0(v)$ recebeu a renumeração 1). Renumeram-se os demais vértices de $L_i(v)$ na ordem $u_{j^*+1}, u_{j^*+2}, \cdots, u_{|L_i(v)|}, u_1, \cdots, u_{j^*-1}$ se $j^* \neq 1$ e $u_{j^*}, u_{j^*+1}, \cdots, u_{|L_i(v)|}$ se $j^* = 1$. Repete-se esse passo para cada nível da estrutura de nível enraizada. Como a construção da estrutura de nível enraizada é pela busca em largura, essa sub-rotina é $O(|V| + |E|)$.

Mostra-se a sub-rotina para a construção da solução inicial no algoritmo 14. O algoritmo recebe um grafo $G = (V, E)$ e a numeração S original do grafo. Inicializam-se as variáveis v, o vetor q, a variável *rotulo* e a variável *vniCor*. O vetor q é o conjunto de vértices no nível corrente da estrutura de nível. A variável *vniCor* é o número de vértices no nível corrente da estrutura de nível enraizada. v é o vértice

inicial e a variável *rotulo* é a renumeração que será atribuída aos vértices. Considere que, inicialmente, $(\forall v \in V)\ v.visitou \leftarrow falso$. Na estrutura de repetição *para cada* nas linhas 12 a 18, visitam-se todos os vértices w adjacentes a u. Marcam-se como visitados aqueles que não foram visitados, armazenando-os em r, que é o vetor dos vértices do próximo nível da estrutura de nível. Ainda, incrementa-se a variável $vniProx$ em uma unidade para cada vértice w visitado. Os passos das linhas 12 a 18 são repetidos para todos os vértices pertencentes a q. Desse modo, são visitados todos os vértices adjacentes aos vértices de q, que ainda não foram visitados. Para iniciar a numeração do vetor q, na linha 10, escolhe-se, aleatoriamente, um índice j^* no intervalo [1 e $|L_i(v)|$]. Na estrutura de repetição das linhas 11 a 23, numeram-se os vértices do nível corrente. Especificamente, nas linhas 19 a 22, numeram-se os vértices do nível corrente da estrutura de nível, com a variável *rotulo*. Realiza-se a numeração de $s(q(j^*)), s(q(j^*+1)), \cdots, s(q(vni)), s(q(1)), s(q(2)), \cdots, s(q(j^*-1))$ se $j^* \neq 1$ e $s(q(j^*)), s(q(j^*+1)), \cdots, s(q(vni))$ se $j^* = 1$. A cada vértice numerado, incrementa-se a variável *rotulo* em uma unidade. Na estrutura de repetição da linha 24, copia-se o conteúdo do vetor r para q. Na linha 25, copia-se o conteúdo da variável $vniProx$ para $vniCor$. Dessa forma, o conteúdo do vetor q estará atualizado para que o próximo nível da estrutura de nível seja numerado, na próxima iteração. O algoritmo para quando *rotulo* exceder $|V|$. O algoritmo retorna S renumerado.

3.4.2 Perturbação

Nessa etapa, realiza-se uma perturbação iniciando-se de uma solução (ou numeração) S e gera-se uma solução $S' \in N_k(S)$. Na heurística VNS-*band*, essa perturbação consiste em realizar uma troca das numerações dos vértices. Essa etapa é dividida em três procedimentos. O primeiro, mostrado na subseção 3.4.2.1, é um método que seleciona previamente alguns vértices que possuem as maiores larguras de banda e trocam-se as suas respectivas numerações. No segundo método, mostrado na subseção 3.4.2.2, é realizada a rotação das numerações de alguns vértices. Finalmente, a terceira parte, mostrada na subseção 3.4.2.3, é um procedimento que, dependendo das circunstâncias, seleciona qual dos dois procedimentos anteriores será utilizado para realizar a perturbação.

Para o entendimento desta etapa, é necessário compreender o conceito de diferença (ou distância) entre as soluções. A distância k entre duas soluções S e S' é dada pelo *número de vértices com numeração diferente entre S e S'* e decrementa-se um dessa diferença. Uma solução S' pertence à vizinhança $N_k(S)$ se a solução S' difere de S em $k+1$ numerações, ou seja, $\mathfrak{h}(S, S') = k \iff S' \in N_k(S)$, em que $\mathfrak{h}$ é a distância entre S e S'. A distância $\mathfrak{h}$ pode ser definida como uma distância de Hamming [33]. Essa distância é o número de posições em que duas *strings* de mesmo comprimento diferem entre si. A distância $\mathfrak{h}$ entre as soluções S e S' é definida por

$$\mathfrak{h}(S, S') = \left(\sum_{i=1}^{|V|} f(i) \right) - 1,$$

em que

$$f(i) = \begin{cases} 1, & \text{se } s(i) \neq s'(i) \\ 0, & \text{caso contrário} \end{cases}$$

e $s(i)$ corresponde à numeração do vértice i na solução S.

Algoritmo 14: solução inicial.

Entrada: grafo $G = (V, E)$; numeração S;
Saída: S renumerado;

```
1  início
2     para cada ( v ∈ V ) faça v.visitou ← falso;
3     v ← VerticeAleatorio(V); // retorna um vértice aleatório de V
4     q(1) ← v; // q é o nível corrente da estrutura de nível
5     v.visitou ← verdadeiro;
      // rotulo é a numeração que será atribuída aos vértices
6     rotulo ← 1;
7     vniCor ← 1; // vniCor é o número de vértices no nível corrente
8     enquanto ( rotulo ≤ |V| ) faça
         // vniProx é o número de vértices no próximo nível da
9        vniProx ← 0; // estrutura de nível
10       j* ← InteiroAleatorio(1, vni);
11       para ( i ← 1; i ≤ vniCor; i ← i + 1 ) faça
12          para cada ( vértice w ∈ Adj(G, q(i)) ) faça
13             se ( w.visitou = falso ) então
14                vniProx ← vniProx + 1;
15                w.visitou ← verdadeiro;
                  // armazena-se em r o próximo nível da estrutura
                  // de nível; considera-se que há memória
16                r(vniProx) ← w; // suficiente para r
17             fim-se;
18          fim-para-cada;
19          s(q(j*)) ← rotulo;
20          rotulo ← rotulo + 1;
21          j* ← j* + 1;
22          se ( j* > vniCor ) então j* ← 1;
23       fim-para;
24       para ( j ← 1; j ≤ vniProx; j ← j + 1; ) faça q(j) ← r(j);
25       vniCor ← vniProx;
26    fim-enquanto;
27    retorna S;
28 fim.
```

Como exemplo, considere a solução $S = \{s(1), s(2), s(3), s(4)\} = (1, 2, 3, 4)$ e $S' = \{s'(1), s'(2), s'(3), s'(4)\} = (4, 3, 2, 1)$. Portanto, $\mathfrak{h}(S, S') = 3$, porque a numeração do S e S' diferem em todos os quatro vértices. Note que o número de trocas na numeração de S para resultar em S' deve ser menor ou igual a k. No exemplo, para se obter a numeração S', deve-se realizar duas trocas, $s(1)$ com $s(4)$ e $s(2)$ com $s(3)$.

3.4.2.1 Primeiro processo da perturbação

Considere que A_S é a matriz obtida pela numeração S. No primeiro procedimento, inicialmente, seleciona-se um conjunto de vértices $Y \subseteq V$ com larguras de banda maior que um valor β', em que $Y = \{v \ : \ \beta_v(A_S) \geq \beta'\}$ e $|Y| \geq k$ para que suas numerações sejam trocadas. Desse modo, selecionam-se vértices com as maiores

larguras de banda. Em seguida, seleciona-se um vértice aleatório $u \in Y$ e seu *vértice crítico* v. Um vértice crítico v de u é um vértice adjacente a u com a maior diferença de numeração, ou seja, $|s(u) - s(v)| = \beta_u(A_S)$ (veja as definições 12 e 13 da seção 1.4, na página 9). O próximo passo é encontrar um vértice w, tal que $s_{\min}(u) \leq s(w) \leq s_{\max}(u)$, em que $s_{\max}(u) = \max_{\{u,u'\} \in E}(s(u'))$ e $s_{\min}(u) = \min_{\{u,u'\} \in E}(s(u'))$. Isso significa que a numeração do vértice w não pode aumentar $\beta_u(A_S)$. Além disso, é preciso analisar também as adjacências de w de forma que colocar v no lugar de w não altere $\beta_v(A_S)$ na nova posição de v. Para isso, verifica-se $\max(|s(v) - s_{\min}(w)|, |s_{\max}(w) - s(v)|)$. Com isso, busca-se colocar v com uma nova numeração de forma que $\beta_v(A_S)$ seja a menor possível.

Mostra-se a primeira sub-rotina de perturbação no algoritmo 15. O algoritmo recebe o grafo $G = (V, E)$, o raio k da solução corrente, a numeração S corrente e o limiar β'. O valor de β', passado para o algoritmo 15, é de forma que $|Y| \geq k$. Na estrutura de repetição *para* nas linhas 4 a 29, seleciona-se o vértice $u \in Y$, o vértice crítico v de u e o vértice w para trocar com v. Repete-se esse processo k vezes. A sub-rotina retorna a renumeração $S' \in N_k(S)$.

3.4.2.2 Segundo processo da perturbação

Segundo Mladenovic et al. [66], utilizar apenas o algoritmo 15 para escapar de um mínimo local não é suficiente em algumas situações. Portanto, utiliza-se uma segunda sub-rotina de perturbação, proposta por Rodriguez-Tello, Kao e Torres-Jimenez [74], baseada na rotação de duas vizinhanças.

Na sub-rotina de rotação de vizinhanças na etapa de perturbação da meta-heurística VNS, troca-se a numeração de todos os vértices que pertencem a um intervalo preestabelecido. Inicialmente, para delimitar o intervalo de troca, escolhem-se dois índices b e f com um terceiro índice m entre eles, ou seja, $1 \leq b < m < f \leq |V|$. Trocam-se as numerações de todos os vértices no intervalo $[b, f]$, deslocando-os m posições, ou seja, a numeração $s(b)$ passará a ser a posição $s(b + m)$, se $b + m \leq f$. Para um índice i, em que $i + m > f$, a numeração $s(i)$ será deslocada para a posição $s(b + |f - (i + m)|)$.

A sub-rotina de rotação é mostrada no algoritmo 16. O algoritmo recebe um grafo $G = (V, E)$, o raio k da solução corrente e a numeração S corrente. Dentro da estrutura de repetição *para* nas linhas 3 a 12, primeiramente, escolhem-se, aleatoriamente, os índices b e f do intervalo de troca. Estabelecem-se, aleatoriamente, os índices b e f, em que, na função *min*, utilizam-se os parâmetros 20 e $\beta(A_S)/2$. Esses parâmetros foram sugeridos por Mladenovic et al. [66], pois gerou bons resultados nos testes realizados. Na linha 7, estabelece-se m, também aleatório, no intervalo (b, f). Na estrutura de repetição *para* nas linhas 8 a 11, realiza-se a rotação no intervalo $[b, f]$. A sub-rotina retorna a numeração $S' \in N_k(S)$.

3.4.2.3 Escolha do processo da perturbação

Uma sub-rotina de escolha é utilizada para decidir qual sub-rotina de perturbação utilizar. O algoritmo 17 define quando utilizar o algoritmo de perturbação 15 ou 16. No algoritmo 17, o valor k'_{max} é definido de acordo com a necessidade de perturbação para se escapar de um vale. Se $k \leq k'_{max}$, utiliza-se o algoritmo 15; caso contrário, utiliza-se o algoritmo 16. Segundo Mladenovic et al. [66], verificou-se que o valor de k, quando passado ao algoritmo 16, gerava perturbações desnecessárias, pois necessita-se de $k = \left\lfloor \frac{k - k_{min}}{k_{step}} \right\rfloor$, em que k_{min} é o raio inicial para a busca na

Algoritmo 15: Perturbação1.

Entrada: grafo $G = (V, E)$; raio da vizinhança k; numeração S; limiar β';
Saída: renumeração S';

```
1  início
2      S' ← S;
       // SelecionaVerticesParaTroca retorna um conjunto de vértices
       // com β_v(A_S) ≥ β', em que β' é um limiar para que |Y| ≥ k
       // e β_u(A_s) é a largura de banda da u-ésima linha da matriz A
       // obtida pela numeração S
3      Y ← SelecionaVerticesParaTroca(S, β');
4      para ( i ← 1; i ≤ k; i ← i + 1 ) faça
5          u ← VerticeAleatorio(Y);
6          v ← ∅;
7          β_u ← −∞;
           // seleciona um vértice v, em que |s(u) − s(v)| = β_u(A_S)
8          para cada ( vértice v' ∈ Adj(G, u) ) faça
9              β_uv' ← |s(u) − s(v')|;
10             se ( β_uv' > β_u ) então
11                 β_u ← β_uv';
12                 v ← v';
13             fim-se;
14         fim-para-cada;
15         menor ← +∞;
16         w ← ∅;
17         para cada ( w' ∈ V ) faça
18             se ( s_min(u) ≤ s(w') ≤ s_max(u) ∧ s_min(w) ≤ s(v) ≤ s_max(w) )
               então
19                 maximo ← max(|s_max(w') − s(v)|, |s(v) − s_min(w')|);
20                 se ( maximo < menor ) então
21                     menor ← maximo;
22                     w ← w';
23                 fim-se;
24             fim-se;
25         fim-para-cada;
26         aux ← s(v); // troca a numeração de v pela numeração de w
27         s(v) ← s(w);
28         s(w) ← aux;
29     fim-para;
30     retorna S';
31 fim.
```

vizinhança e k_{step} é o incremento do raio a cada passo. Quando não se consegue escapar de um mínimo local, o raio da vizinhança $N_k(S)$ é incrementado por k_{step}, possibilitando novas soluções.

A sub-rotina de escolha é mostrada no algoritmo 17. O algoritmo recebe o raio k da solução corrente, k_{min} como o raio inicial, k_{step} como o incremento para aumentar o raio de busca, k'_{max}, que é o raio máximo, a numeração S corrente e o limiar β'.

Algoritmo 16: Perturbação2.

Entrada: grafo $G = (V, E)$; raio da vizinhança k; numeração S;
Saída: renumeração S';

```
1  início
2      S' ← S;
3      para ( i ← 1; i ≤ k; i ← i + 1 ) faça
4          b ← InteiroAleatorio(1, |V|);
5          f ← b + InteiroAleatorio(2, min(20, β(A_S)/2));
6          se ( f > |V| ) então f ← |V|;
7          m ← InteiroAleatorio(b + 1, f − 1);
8          para ( j ← b; j ≤ f; j ← j + 1 ) faça
9              se ( j ≥ b + m ) então  s'(j) ← j − m;
10             senão  s'(j) ← j + f − b − m + 1;
11         fim-para;
12     fim-para;
13     retorna S';
14 fim.
```

Algoritmo 17: Perturbação.

Entrada: grafo $G = (V, E)$; raio da vizinhança k; raio inicial k_{min}; raio incremento k_{step}; numeração S; raio máximo k'_{max}; limiar β';
Saída: renumeração S';

```
1  início
2      se ( k ≤ k'_max ) então  S' ← Perturbacao1(G, k, S, β'); //algoritmo 15
3      senão  S' ← Perturbacao2(G, ⌊(k − k_min)/k_step⌋, S); //algoritmo 16
4      retorna S';
5  fim.
```

3.4.3 Busca local

A busca local da heurística VNS-*band* é baseada na busca local proposta por Lim, Rodrigues e Xiao [54]. Essa busca local é basicamente a mesma da heurística de Lim, Rodrigues e Xiao [55] que é descrita na subseção 3.2.2. Segundo Mladenovic et al. [66], a busca local proposta por Lim, Rodrigues e Xiao [54] apresentou resultados melhores na redução de largura de banda do que a busca local original utilizada pela meta-heurística VNS [65].

Nessa etapa, seleciona-se um conjunto de vértices críticos do grafo para que suas numerações sejam trocadas. Um vértice $v \in V$ é um vértice crítico de $G = (V, E)$ se existe um vértice $u \in V$ adjacente a v, em que $\{u, v\} \in E$ é uma *aresta crítica* de $G = (V, E)$. Uma aresta $\{u, v\} \in E$ é crítica se $|s(u) - s(v)| = \beta(A_S)$. O conjunto E_c, conjunto de arestas críticas de $G = (V, E)$, é definido por $E_c = \{\{u, v\} \in E : |s(u) - s(v)| = \beta(A_s)\}$. O conjunto de vértices críticos V_c de $G = (V, E)$ é todo vértice de arestas de E_c.

Para a aplicação da busca local, define-se a vizinhança reduzida para a troca de numeração de v como $\mathrm{N}(v, S)$ do vértice crítico v, em que $\mathrm{N}(v, S) = \{u : (\{u, v\} \in E_c)\ |med(v) - s(u)| < |med(v) - s(v)|\}$ e $med(v) = \left\lfloor \frac{s_{max}(v) + s_{min}(v)}{2} \right\rfloor$. O conjunto de vértices para a aplicação da busca local, ou o conjunto de vizinhanças reduzidas para

a troca de numeração em relação à solução S, é definido por $N(S) = \bigcup_{v \in V_c} \mathrm{N}(v, S)$.

Como a etapa da busca local é depois da etapa de perturbação, S', nesse passo, é a solução corrente. Com $N(S')$, forma-se um conjunto de vértices apropriados para a troca. A busca local é aplicada no conjunto $N(S')$. Troca-se a numeração do vértice crítico v com cada numeração dos vértices $u \in \mathrm{N}(v, S') \subseteq N(S')$ até que uma melhora em relação à solução corrente S' seja constatada. Com isso, gera-se a solução S''.

3.4.4 Permutação de vizinhança

A etapa de permutação de vizinhança consiste em decidir sobre duas questões: quando permutar ou não permutar a solução corrente e qual será a próxima vizinhança. Para isso, muitas estratégias foram propostas e Hansen e Mladenović [35] apresentaram uma revisão.

1. Quando permutar (mover) ou não: para a primeira questão, necessita-se saber qual critério utilizar para permutar a solução corrente e quando não permutar. Na redução de largura de banda, utilizam-se três critérios para permutar a solução corrente S para a solução S''.

   (a) Verifica-se qual solução possui a menor largura de banda. Se $\beta(A_S) > \beta(A_{S''})$, então, S'' passa a ser a solução corrente, ou seja, $S \leftarrow S''$. Em caso contrário, S continua como a solução corrente.

   (b) Se $\beta(A_S) = \beta(A_{S''})$, então, avalia-se o número de vértices críticos de cada solução. Se $|V_c(S)| > |V_c(S'')|$, então, S'' passa a ser a solução corrente. Em caso contrário, S continua como a solução corrente, em que $V_c(S)$ é o conjunto de vértices críticos da solução S.

   (c) Se $\beta(A_S) = \beta(A_{S''})$ e $|V_c(S)| = |V_c(S'')|$, então, avalia-se a diferença (distância) entre as soluções S e S''. Troca-se a solução S por S'' se $\mathfrak{h}(S, S'') > \mathfrak{t}$, em que $\mathfrak{t}$ é um parâmetro constante que, segundo Mladenovic et al. [66], nos testes realizados, obteve-se bons resultados para $\mathfrak{t} = 10$. Esse terceiro critério baseia-se em uma variação da metaheurística VNS proposta por Hansen e Mladenović [34], em que se admite a permutação da solução corrente S por um S' pior, se S' estiver relativamente longe de S. Porém, para o problema de redução de largura de banda, se S e S'' possuírem a mesma largura de banda e o mesmo número de vértices críticos, permuta-se de S para S'' apenas se $\mathfrak{h}(S, S'') > \mathfrak{t}$.

2. Próxima vizinhança: a segunda questão é descobrir qual será a vizinhança no caso de haver permutação da solução corrente e como alterar a vizinhança corrente quando não houver permutação. Utilizam-se parâmetros k_{min}, k_{step} e k_{max} para definir se o raio de busca na vizinhança será incrementado, se voltará ao raio inicial ou se excedeu o limite. k_{min} é o raio inicial para a busca na vizinhança. Se a solução corrente S é trocada para S'', então, atualiza-se o raio da busca de $N_k(S'')$ para k_{min}, ou seja, utiliza-se $N_{k_{min}}(S'')$. Um exemplo pode ser visto na figura 3.1, na página 52. k_{step} é o incremento do raio quando uma troca não é realizada. Se após a perturbação e a busca local não houver melhorias em relação à solução corrente S, então, estabelece-se $k \leftarrow k + k_{step}$. Com isso, se $k < k_{max}$, repetem-se os passos de busca. k_{max} é o raio máximo em que é realizada a busca.

Mostra-se a sub-rotina que verifica se haverá ou não a permutação da solução corrente S pela solução S'' no algoritmo 18. A sub-rotina recebe a solução corrente S, a solução S'' e o parâmetro $\mathfrak{t} = 10$. Verifica-se, no algoritmo 18, se é viável realizar a permutação da solução corrente S pela solução S'' ou não. Se for viável, o retorno do algoritmo é verdadeiro; em caso contrário, o algoritmo 18 retorna falso.

Algoritmo 18: Troca.

Entrada: numeração S; numeração S''; parâmetro $\mathfrak{t}$;
Saída: *verdadeiro* ou $falso$;
1 **início**
    `// permuta-se ou não a solução corrente`
2 **retorna** $(\beta(A_S) > \beta(A''_S)) \vee (\beta(A_S) = \beta(A_{S''}) \wedge |V_c(S)| > |V_c(S'')|) \vee (\beta(A_S) = \beta(A_{S''}) \wedge |V_c(S)| = |V_c(S'')| \wedge \mathfrak{h}(S, S'') > \mathfrak{t})$;
3 **fim.**

3.4.5 VNS-*band*

Nesta subseção, apresenta-se a heurística VNS-*band* completa. A heurística VNS-*band* é mostrada no algoritmo 19. Mladenovic et al. [66] utilizaram o tempo de execução como critério de parada.

O algoritmo recebe o raio k da vizinhança, o raio inicial k_{min}, o incremento k_{step} do raio, o raio máximo k_{max}, o raio pré-definido k'_{max}, o limiar β', a lista de adjacências E, o parâmetro $\mathfrak{t}$, a numeração corrente S e o tempo máximo t_{max}. Mladenovic et al. [66] utilizaram, em todos os testes apresentados no artigo, os seguintes valores nos parâmetros: $k_{min} = k_{step} = 5$, $k'_{max} = 100$, $k_{max} = 200$, $\mathfrak{t} = 10$ e $t_{max} = 500s$.

Nas linhas 2 a 4, as variáveis β_{min}, t e i_{max} são inicializadas, respectivamente. β_{min} é a variável com a menor largura de banda encontrada durante a execução, t é o tempo inicial e i_{max} recebe $\lfloor (k_{max} - k_{min})/k_{step} \rfloor$. Na linha 6, dentro da estrutura de repetição *enquanto* externa, constrói-se a solução inicial.

Após a construção da solução inicial pelo algoritmo 14, aplica-se a busca local nessa solução e inicializam-se as variáveis i e k. Na estrutura de repetição *enquanto* interna nas linhas 11 a 22, aplica-se a perturbação na solução corrente S, gerando-se S'. Em seguida, gera-se S'', aplicando-se a busca local em S'. Ainda dentro do *enquanto* interno, na estrutura condicional, verifica-se se haverá troca da solução corrente S pela solução S''. Caso troque, a solução corrente passa a ser S'' e atualiza-se o raio k para k_{mim}. Em caso contrário, incrementa-se o raio k com k_{step}. Na estrutura condicional nas linhas 23 a 26, verifica-se se a largura de banda da solução corrente é menor do que a menor largura de banda encontrada até o momento. Caso seja, guarda-se a largura de banda em β_{min} e a numeração correspondente em S^*. A sub-rotina $TempoUCP$ na linha 27 retorna o tempo decorrido. O algoritmo para quando o tempo de processamento excede o limite t_{max}. O algoritmo retorna a renumeração S^* do grafo.

3.5 Heurística por busca em sistema carregado

A *busca em sistema carregado*, *Charged System Search* (*CSS*), é uma meta-heurística desenvolvida por Kaveh e Talatahari [46] para a resolução de problemas relaciona-

Algoritmo 19: VNS-*band*.

Entrada: grafo $G = (V, E)$; raio da vizinhança k; raio inicial k_{min}; raio incremento k_{step}; raio máximo k_{max}; raio pré-definido k'_{max}; limiar β'; parâmetro $\mathfrak{t}$; numeração S; tempo máximo t_{max};
Saída: renumeração final S^*;

```
1  início
2      β_min ← +∞; // β_min terá a menor largura de banda encontrada
3      t ← 0;
4      i_max ← ⌊(k_max − k_min)/k_step⌋;
5      enquanto ( t < t_max ) faça
6          S ← SolucaoInicial(S); // algoritmo 14
7          S ← BuscaLocal(S); // etapa descrita na subseção 3.4.3
8          S* ← S;
9          i ← 0;
10         k ← k_min;
11         enquanto ( i ≤ i_max ) faça
12             S' ← Perturbacao(G, k, k_min, k_step, k'_max, S, β'); // algoritmo 17
13             S'' ← BuscaLocal(S');
               // algoritmo 18
14             se ( Troca(S, S'', t) = verdadeiro ) então
15                 S ← S'';
16                 k ← k_min;
17                 i ← 0;
18             senão
19                 k ← k + k_step;
20                 i ← i + 1;
21             fim-se;
22         fim-enquanto;
23         se ( β(A_S) < β_min ) então
24             β_min ← β(A_S);
25             S* ← S; // S* é a numeração final
26         fim-se;
27         t ← TempoUCP();
28     fim-enquanto;
29     retorna S*;
30 fim.
```

dos à análise de estruturas. Os autores utilizaram as leis de Coulomb e de Gauss, da eletrostática, e as leis da mecânica de Newton para desenvolver essa meta-heurística.

O CSS é composto por *partículas carregadas* (PC). Cada PC é considerada uma esfera eletricamente carregada que exerce uma força elétrica sobre as outras partículas, de acordo com as leis de Coulomb e de Gauss. A força resultante e as leis da mecânica determinam as novas posições e as velocidades das PCs. A utilização de tais leis pelo CSS provê um modelo que pode ser utilizado em problemas de otimização discretos e contínuos.

Kaveh e Sharafi [44] utilizaram um modelo baseado nessas leis para criar uma proposta para a renumeração dos vértices de um grafo. Com isso, utilizaram essa meta-heurística para as reduções de banda e de *profile* de matrizes esparsas.

Esta seção é dividida como a seguir. Na subseção 3.5.1, são descritas definições utilizadas pela meta-heurística CSS. Descreve-se, na subseção 3.5.2, o pseudo-código da heurística de Kaveh e Sharafi [44].

3.5.1 Definições

Nesta subseção, são apresentadas as leis da eletrostática e da mecânica newtoniana utilizadas pelo CSS. Na subseção 3.5.1.1, são descritas algumas definições das leis da eletrostática necessárias para o entendimento do CSS. Algumas leis da mecânica newtoniana são descritas na subseção 3.5.1.2. Descreve-se, na subseção 3.5.1.3, algumas regras utilizadas pelo CSS.

3.5.1.1 Leis da eletrostática

Um campo elétrico que envolve uma carga pontual é definido pela lei de Coulomb. Essa lei provê a magnitude da força elétrica entre duas cargas pontuais. A lei de Coulomb é expressa por

$$F_{ij} = k_e \frac{q_i q_j}{{r_{ij}}^2} \frac{r_i - r_j}{\|r_i - r_j\|}, \tag{3.5.5}$$

em que k_e é a constante de Coulomb e r_{ij} é a distância entre duas cargas q_i e q_j, que estão nas respectivas posições r_i e r_j. Para o caso em que a distância r_{ij}, de r_i para r_j, é menor do que o raio a de uma esfera, utiliza-se a lei de Gauss. Com isso, tem-se a força exercida em um ponto dentro da esfera expressa por

$$F_{ij} = k_e \frac{q_i q_j}{a^3} \frac{r_i - r_j}{\|r_i - r_j\|}. \tag{3.5.6}$$

Segundo Kaveh e Sharafi [44], para calcular a força elétrica sobre uma carga q_j, na posição r_j, devido a um grupo de pontos carregados, o princípio da superposição é aplicado a forças elétricas como

$$F_j = \sum_{i=1, i \neq j}^{N_p} F_{ij}, \tag{3.5.7}$$

em que N_p é o total de partículas carregadas. A força F_{ij} é obtida pela equação (3.5.5), caso $r_{ij} \geq a$, ou pela equação (3.5.6), caso $r_{ij} < a$.

3.5.1.2 Leis da mecânica

Na mecânica newtoniana, estuda-se o movimento dos objetos sem considerar os seus tamanhos. O deslocamento de uma partícula pontual, de uma posição r_0 até a posição r_1, é representado por $\Delta r = r_1 - r_0$.

A velocidade $v = \frac{\Delta r}{\Delta t}$ é a razão entre o deslocamento Δr e o intervalo de tempo Δt. A aceleração é definida por $\alpha = \frac{\Delta v}{\Delta t}$, em que Δv é a variação da velocidade. A posição final de um objeto, que estava inicialmente com velocidade v_0 na posição r_0, é obtida por $r_1 = r_0 + v_0 \Delta t + \frac{\alpha (\Delta t)^2}{2}$. A segunda lei de Newton define $F = m\alpha$, em que a força F é obtida pelo produto da massa m pela aceleração α. Dessa forma, a posição final de um objeto pode ser reescrita como $r_1 = r_0 + v_0 \Delta t + \frac{F (\Delta t)^2}{2m}$.

3.5.1.3 Regras da meta-heurística busca em sistema carregado

Nesta subseção, são apresentadas as regras do CSS, de acordo com as leis físicas apresentadas. Cada PC é afetada pelos campos elétricos das outras partículas. A quantidade da força resultante é determinada pelas lei da eletrostática e a qualidade do movimento é determinada pelas leis da mecânica de Newton [44]. Antes de se apresentar o CSS, é necessário definir algumas regras, dadas a seguir.

1. Cada PC possui uma carga elétrica q_i que gera um campo elétrico em torno de si. A magnitude da carga é definida por $q_i = \frac{apt(i)-apt_{melhor}}{apt_{melhor}-apt_{pior}}$, com $i = 1, 2, ..., M$, em que M é o total de PCs. O melhor e pior valores de *aptidão* (*fitness*), de todas as partículas são, respectivamente, apt_{melhor} e apt_{pior}. A função *aptidão*, ou função objetivo, é representada por $apt(i)$, para a partícula i. A distância entre duas cargas, presentes nas posições X_i e X_j, é definida por $r_{ij} = \frac{\|X_i - X_j\|}{\|(X_i+X_j)/2 - X_{melhor}\|+\varepsilon}$, em que X_{melhor} é a melhor partícula corrente e ε é uma constante com valor próximo a zero para se evitar singularidades. A posição de uma PC, ou possível solução X_i, é composta por n vetores $x_{i,j}$, para $j = 1, 2, ..., n$. Os significados de X_j e $x_{i,j}$ são diferentes e explicados adiante.

2. A posição inicial de cada PC é definida aleatoriamente e as velocidades são, inicialmente, zero.

3. As PCs podem atrair-se ou repelir-se.

4. As PCs são consideradas boas ou ruins de acordo com o seu coeficiente c_j. As partículas acima da média são consideradas boas, atraem as outras e possuem coeficiente $0 < c_j \leq 1$. Partículas abaixo da média são consideradas ruins, repelem as outras e seu coeficiente é $-1 \leq c_j < 0$.

5. O valor da força resultante em cada PC é determinado por

$$F_j = q_j \sum_{i=1, i\neq j}^{M} \left(\frac{q_i}{a^3} r_{ij} i_1 + \frac{q_i}{(r_{ij})^2} i_2 \right) c_{ij} (X_i - X_j). \tag{3.5.8}$$

   A equação (3.5.8) é uma adaptação da equação (3.5.7). Utilizam-se i_1 e i_2 para realizar essa adaptação. Para a heurística CSS-*band*, Kaveh e Sharafi [44] utilizaram $a = 1$. Dessa forma, se $r_{ij} < a$, tem-se $i_1 = 1$ e $i_2 = 0$. Se $r_{ij} \geq a$, tem-se $i_1 = 0$ e $i_2 = 1$.

6. A nova posição de cada PC é determinada por

$$x_{j,atual} = fix(rand_1 . k_a . \frac{F_j}{m_j} . \Delta t^2 + rand_2 . k_v . v_{j,anterior} . \Delta t + x_{j,anterior}), \tag{3.5.9}$$

   em que $k_v = 0,5 \cdot (1 - iter/iter_{max})$, $k_a = 0,5 \cdot (1 + iter/iter_{max})$, $iter$ é o valor da iteração atual e $iter_{max}$ é o valor máximo de iterações do algoritmo. O valor do coeficiente de velocidade k_v controla a influência da velocidade anterior. O coeficiente de aceleração k_a aumenta a velocidade de convergência do algoritmo. O valor de k_v diminui e o valor de k_a aumenta à medida que o valor de $iter$ aumenta. Os valores de $rand_1$ e $rand_2$ são gerados aleatória e uniformemente distribuídos no intervalo $(0, 1)$. Nestas abordagens, tem-se $\Delta t = 1$ e $m_j = q_j$. A função fix arredonda o valor para o inteiro mais

próximo ainda não escolhido, que será o novo rótulo do vértice. Por exemplo, se a entrada para fix for $4,9$ e o rótulo 5 já estiver associado a algum vértice, será retornado 4, desde que esse rótulo esteja disponível. A nova velocidade de uma partícula é determinada por

$$v_{j,atual} = \frac{X_{j,atual} - X_{j,anterior}}{\Delta t}, \tag{3.5.10}$$

em que a velocidade da j-ésima PC depende da posição atual $X_{j,atual}$ e da posição anterior $X_{j,anterior}$.

7. As $M/4$ melhores partículas são mantidas na memória *cache* (MC) e também podem influenciar as outras PCs. As $M/4$ piores partículas são impedidas de influenciar as outras PCs. O valor de M representa a quantidade de PCs.

8. As partículas que violam o limite são regeneradas por meio da *busca harmônica* (BH), descrita por Kaveh e Talatahari [45, 46]. O algoritmo BH é baseado nas probabilidades: *taxa considerada da memória harmônica* (TCMH) e *taxa de ajuste potencial* (TAP), que têm seus valores atribuídos aleatoriamente no intervalo $(0,1)$. O valor $TCMH$ define a probabilidade de se escolher aleatoriamente um valor presente na MC. O valor $1-TCMH$ define a probabilidade de se definir aleatoriamente um valor. Se um valor de MC é escolhido, então, é avaliada a probabilidade $TAP \cdot TCMH$ de se trocar esse valor por um vizinho no vetor MC. A probabilidade $1 - TAP$ definirá se não será escolhido um valor vizinho.

9. O critério de parada é o número máximo de iterações.

3.5.2 Pseudocódigo para a heurística CSS-*band*

A heurística CSS-*band* tenta encontrar uma renumeração para os vértices de um grafo $G = (V, E)$ para reduzir a largura de banda ou o *profile*. A função objetivo, ou *aptidão*, utilizada para a redução de largura de banda é diferente da função utilizada para a redução de *profile*. A função objetivo avalia o resultado obtido pela partícula, conforme a regra 1.

Para a redução de *profile* de uma matriz A, de dimensão $|V| \times |V|$, que representa o grafo $G = (V, E)$, a função objetivo é

$$P = \sum_{i=1}^{|V|} \beta_i, \tag{3.5.11}$$

em que $\beta_i = i - \min_{1 \leq j \leq |V|}(j \mid a_{ij} \neq 0)$ é a largura de banda da i-ésima linha da matriz.

Para a redução de largura de banda, a função objetivo, que representa a largura de banda do grafo, é $\beta(A) = \max_{1 \leq i \leq |V|}(\beta_i)$. Para a matriz A, cada permutação de linha e coluna é considerada uma solução potencial e é representada por uma PC. Cada PC, ou possível solução X_i, é composta por n vetores $x_{i,j}$, para $j = 1, 2, ..., n$, contendo o número associado ao vértice $j \in V$. O algoritmo 20 contém as nove regras descritas.

Na linha 3 do algoritmo 20, é calculada a quantidade $M = fix(|V|/100) + 5$ de PCs. Existirá uma PC para cada 100 vértices e pelo menos cinco partículas são necessárias. Uma quantidade grande de PCs aumenta o tempo de execução do

Algoritmo 20: CSS-*band*.

Entrada: grafo $G = (V, E)$ e o número máximo de iterações $iter_{max}$;
Saída: grafo $G = (V, E)$ com os vértices reordenados;

```
1  início
2      iter ← 1;
3      M ← fix(|V|/100) + 5;
       // calcula quantidade de PCs
       // inicializam-se as M PCs com posição aleatória e
       // velocidade v_j = 0
4      para ( i ← 1; i ≤ M; i ← i + 1 ) faça Inicializa(PC_i);
       // avaliam-se as PCs (largura de banda ou profile),
       // ordenando-as em ordem crescente de coeficiente c_i
5      para ( i ← 1; i ≤ M; i ← i + 1 ) faça  Avalia(PC_i);
6      InicializaMC(); // vetor com as M/4 melhores PCs
7      enquanto ( iter ≤ iter_max ) faça
           // cálculo da carga de cada PC, pela regra 1
8          DeterminaCargaPCs();
           // cálculo da força resultante de cada PC pela regra 5
9          DeterminaForcaResultantePCs();
           // cálculo da posição de cada PC pela equação (3.5.9)
10         DeterminaPosicaoPCs();
           // cálculo da velocidade de cada PC pela equação (3.5.10)
11         DeterminaVelocidadePCs();
12         para ( i ← 1; i ≤ M; i ← i + 1 ) faça
13             se ( EstouraLimites(PC_i) ) então  BH(PC_i);
14         fim-para;
15         para ( i ← 1; i ≤ M; i ← i + 1 ) faça  Avalia(PC_i);
16         se ( NovasPCsMelhoresMC() ) então
               // removem-se as piores PCs e inserem-se as melhores em
17             AtualizaMC(); // MC
18             para ( i ← 1; i ≤ M; i ← i + 1 ) faça Avalia(PC_i);
19             para ( i ← 1; i ≤ M; i ← i + 1 ) faça
20                 se ( EstouraLimites(PC_i) ) então  BH(PC_i);
21             fim-para;
22         fim-se;
23         iter ← iter + 1;
24     fim-enquanto;
25     para ( i ← 1; i ≤ M; i ← i + 1 ) faça  Avalia(PC_i);
26     retorna MelhorPC(); // retorna a melhor solução encontrada
27 fim.
```

algoritmo. Por outro lado, uma quantidade pequena de PCs levará a um resultado indesejável [44]. Na linha 4, cada PC é inicializada com velocidade $v_{i,j} = 0$, vetor de permutação e posição aleatórios, de acordo com a regra 2.

Na linha 5, a sub-rotina *Avalia* avalia cada partícula e coloca-as em ordem crescente do coeficiente c_j. A sub-rotina *Avalia* executa a função objetivo, que depende do tipo de redução, de largura de banda ou de *profile*, que se pretende realizar. Para isso, é associado o valor 1 para a melhor PC e -1 para a pior. Será associado um valor no intervalo $(-1, 0)$ ou no intervalo $(0, 1)$ para as outras

partículas, pelas regras 3 e 4. Na linha 6, inicializa-se o vetor MC pela regra 7. MC conterá as $M/4$ melhores PCs.

A regra 9 é considerada na estrutura de repetição das linhas 7 a 24. Na linha 8, a magnitude da carga de cada PC é calculada pela regra 1. Essa etapa é diferente para as reduções de banda ou de *profile*. Para a redução de largura de banda, a função objetivo é dada por $\beta(A)$. Para a redução de *profile*, a função objetivo é dada pela equação (3.5.11).

Na linha 9, o valor da força elétrica resultante é calculado para cada PC, conforme a equação (3.5.8) da regra 5. As novas posição e velocidade de cada PC são determinadas pelas equações (3.5.9) e (3.5.10), respectivamente, nas linhas 10 e 11, pela regra 6. As partículas que violarem os limites permitidos são regeneradas pela BH, nas linhas 12 a 14, pela regra 8.

As PCs são avaliadas novamente na linha 15. Na linha 16, se os valores obtidos forem melhores do que os armazenados no vetor MC, as piores PCs são excluídas e as melhores PCs são inseridas em MC, na linha 17. Na linha 18, as partículas são avaliadas novamente. Nas linhas 19 a 21, as PCs que violarem os limites serão novamente restauradas pela BH. Finalmente, as partículas são avaliadas, na linha 25. Na linha 26, a melhor PC, que representa a melhor solução encontrada para a reordenação do grafo $G = (V, E)$, é retornada.

3.6 Exercícios

1. Descreva, sucintamente, a busca local realizada pelas heurísticas NC-HC e FNC-HC.

2. Realize a etapa do *Node Centroid* (NC), mostrada na subseção 3.2.1, na página 44, da heurística NC-HC no grafo da figura 3.2. Considere $\lambda = 0.5$. Verifique se houve redução na largura de banda da matriz correspondente

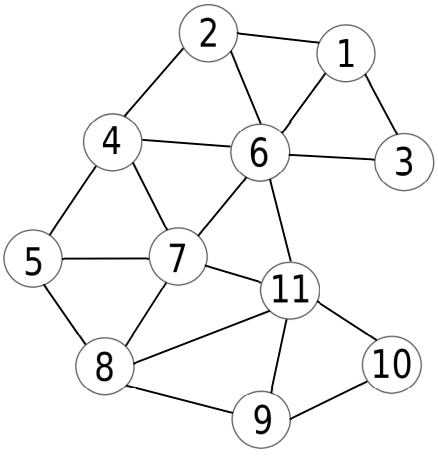

Figura 3.2: Grafo com vértices renumerados pela heurística CMr.

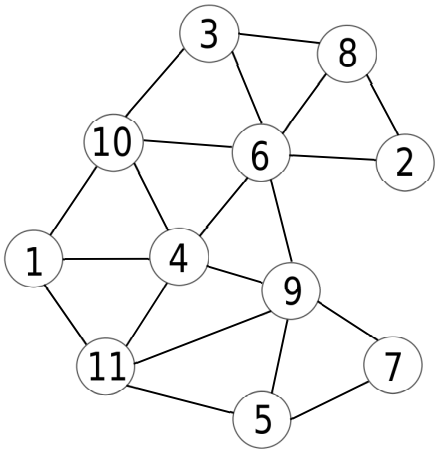

Figura 3.3: Grafo com vértices numerados aleatoriamente.

3. Descreva, sucintamente, a heurística de Kaveh e Sharafi [43] baseada em Otimização por Colônia de Formigas.

4. Descreva, sucintamente, a heurística VNS-*band* de Mladenovic et al. [66].

5. Realize a etapa da construção da solução inicial da heurística VNS-*band*, mostrada na subseção 3.4.1, na página 53. Para isso, considere o grafo mostrado na figura 3.3. Considere, também, que o vértice inicial é o vértice 3. Note que, para a execução dessa etapa, os vértices precisam estar organizados por algum critério. Isso porque a renumeração é realizada nível a nível da estrutura de nível iniciando-se de um vértice aleatório. Os vértices de cada nível são numerados a partir desse vértice.

6. Descreva, sucintamente, a heurística CSS-*band* de Kaveh e Sharafi [44].

7. Cite e explique os passos estocásticos utilizados nas heurísticas ACO-KS [43] e CSS-*band* [44].

Capítulo 4

Algoritmos que encontram um vértice pseudo-periférico

4.1 Introdução

Em algumas heurísticas para a redução de largura de banda, o vértice inicial da renumeração influencia na qualidade da solução. Isso ocorre, como exemplos, nas heurísticas Cuthill-McKee, Cuthill-McKee reverso e quatro-etapas. Neste capítulo, são descritos algoritmos que encontram um vértice inicial para métodos de redução de largura de banda de matrizes.

Cuthill e McKee [9] iniciaram a ordenação com o vértice de grau mínimo do grafo e que apresentasse estrutura de nível enraizada com a menor largura de nível, mas nem sempre essa era uma boa escolha. Após a proposta de Cuthill e McKee [9], houve publicações com algoritmos que encontram o vértice inicial, como em Cheng [6].

Gibbs, Poole e Stockmeyer [26] propuseram que o melhor vértice inicial seria um vértice cuja distância para outro vértice qualquer fosse igual ao diâmetro do grafo, caracterizando-se, assim, vértices periféricos. Isso porque, quanto maior a excentricidade de um vértice v, menos vértices haverá em um mesmo nível de uma estrutura de nível de v [9]. Diâmetro, vértice periférico, excentricidade e estrutura de nível estão definidos na seção 1.4, na página 6.

Um vértice periférico é ótimo vértice inicial para algumas heurísticas de redução de largura de banda. Para se encontrar o vértice periférico, é necessário, por exemplo, aplicar a busca em largura em todos os vértices do grafo. O custo dessa tarefa, no pior caso, é $O(|V| \cdot (|V| + |E|))$, em que $|V|$ é o número de vértices e $|E|$ é o número de arestas do grafo.

Gibbs, Poole e Stockmeyer [26] propuseram, então, como descrito na subseção 2.3.1, na página 19, um algoritmo que encontra vértice cuja excentricidade é próxima ao diâmetro do grafo, caracterizando-se, assim, um vértice pseudo-periférico. A seguir, são listados alguns algoritmos que encontram vértice pseudo-periférico:

- algoritmo de Gibbs, Poole e Stockmeyer [26];
- algoritmo de George e Liu [21];
- algoritmo de Arany [1];
- algoritmo de Pachl [69];

- algoritmo de Smyth [79];
- algoritmos de Kaveh [40];
- algoritmo de Grimes, Pierce e Simon [31];
- algoritmo de Kaveh [41];
- algoritmo de Souza e Murray [81];
- algoritmo de Feng [19];
- razão-largura-profundidade [89, 90].

A seguir, são mostrados alguns algoritmos que encontram um vértice pseudo-periférico. O algoritmo apresentado na seção 4.2 foi desenvolvido por George e Liu [21], que é uma modificação do algoritmo proposto por Gibbs, Poole e Stockmeyer [26]. Na seção 4.3, mostram-se os algoritmos propostos por Kaveh [40, 42]. Na seção 4.4, apresenta-se o algoritmo de Pachl [69]. Na seção 4.5, descreve-se o algoritmo de Wang et al.[90].

4.2 Algoritmo de George-Liu

Esse algoritmo foi proposto como melhoria do algoritmo de Gibbs, Poole e Stockmeyer [26], mostrado na subseção 2.3.1, na página 19. George e Liu [21] propuseram quatro modificações nesse algoritmo. A primeira modificação, citada no texto como *short circuiting*, consiste em terminar a montagem das estruturas de nível enraizadas dos vértices folhas assim que encontrar um nível com largura de nível maior do que a já encontrada. A segunda modificação é iniciar o algoritmo com um vértice qualquer em vez de um com grau mínimo. A terceira e quarta modificações são denominadas *shrinking*, pois ambas são estratégias para escolher um vértice do último nível da estrutura de nível enraizada. A terceira modificação é montar a estrutura de nível enraizada apenas para o vértice folha que possuir o grau mínimo. A última proposta foi escolher um vértice folha qualquer para a montagem da estrutura de nível. Após experimentos, o algoritmo final proposto por George e Liu [21] é a combinação da segunda e da terceira propostas de melhoria. Esse algoritmo é descrito nesta seção.

Como mostrado por George e Liu [21], o algoritmo pode ser dividido em alguns passos simples. Na parte da inicialização, um vértice qualquer $v \in V$ é escolhido para ser o vértice inicial. Em seguida, a estrutura de nível $\mathscr{L}(v)$ é gerada. Em seguida, escolhe-se um vértice de grau mínimo $u \in L_{\ell(v)}(v)$. Gera-se $\mathscr{L}(u)$ e verifica-se se $\ell(v) < \ell(u)$. Se essa condição for satisfeita, então, o vértice u passa a ser o vértice v e o processo é repetido. Caso contrário, o vértice pseudo-periférico será v.

Mostra-se o pseudocódigo no algoritmo 21. Inicializa-se a variável v com um vértice arbitrário. Em seguida, é montada a estrutura de nível enraizada $\mathscr{L}(v)$, utilizando a busca em largura. Na estrutura de repetição *repita* das linhas 4 a 11, constrói-se a estrutura de nível do nó folha u de grau mínimo de $\mathscr{L}(v)$. Se a excentricidade de u for maior que a excentricidade de v, então, atribui-se u a v e repete-se o processo. O algoritmo retorna o vértice v quando $\ell(v)$ for maior ou igual que a excentricidade do seu vértice folha de grau mínimo.

Como um exemplo de uma execução do algoritmo de George e Liu [21], considere o grafo representado na figura 4.1. O vértice $v = 1$ foi escolhido arbitrariamente

Algoritmo 21: George-Liu.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $v \in V$;

```
1  início
2      v ← EscolheVertice(V);
       // constrói-se estrutura de nível enraizada
3      L(v) ← BuscaEmLargura(v);
4      repita
5          u ← VerticeComGrauMinino(L_{ℓ(v)}(v));
           // constrói-se estrutura de nível enraizada
6          L(u) ← BuscaEmLargura(u);
7          se ( ℓ(u) > ℓ(v) ) então
8              v ← u;
9              L(v) ← L(u);
10         fim-se;
11     até que (u ≠ v) ;
12     retorna v;
13 fim.
```

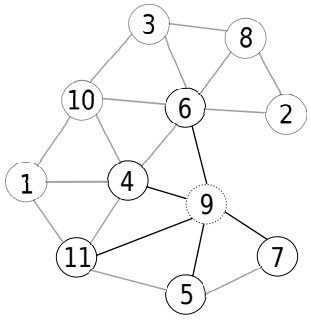

Figura 4.1: Um grafo para a escolha do vértice pseudo-periférico e sua representação matricial M.

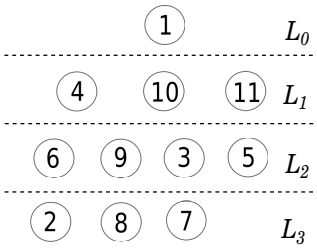

Figura 4.2: Estrutura de nível enraizada do vértice 1 do grafo da figura 4.1.

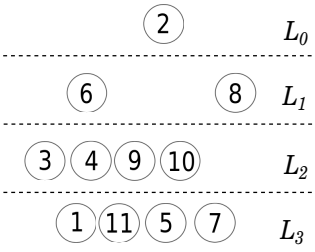

Figura 4.3: Estrutura de nível enraizada para o vértice 2 do grafo da figura 4.1.

para a inicialização do algoritmo. Na figura 4.2, mostra-se a estrutura de nível enraizada do vértice 1. Note que o número de níveis de $\mathscr{L}(1)$ é 4 e $\ell(1) = 3$.

Atribui-se à variável u o vértice de $L_{\ell(v)}(v)$ com grau mínimo. Como os vértices 2 e 7 apresentam grau 2, não há diferença na escolha de um ou outro. Nesse exemplo, optou-se pelo vértice 2. Em seguida, a estrutura de nível do vértice 2 é construída e mostrada na figura 4.3. Como $\ell(v) = \ell(u)$, o algoritmo para, determinando-se o vértice $v = 1$ como o vértice pseudo-periférico. Neste exemplo, o algoritmo de fato encontrou um vértice periférico do grafo.

4.3 Algoritmos de Kaveh

Kaveh [40] propôs algoritmos que encontram um vértice inicial pseudo-periférico adequado para a heurística quatro etapas, mostrado na seção 2.5, na página 30. Os

algoritmos de Kaveh procuram vértices pseudo-periféricos analisando-se somente a largura de nível da estrutura *Shortest Route Tree* (SRT), mostrada na subseção 2.5.1, na página 32. Para o algoritmo quatro-etapas, há a necessidade da criação da estrutura SRT. Entretanto, por simplicidade, apresentam-se os algoritmos aqui ao se utilizar a estrutura de nível enraizada e o grafo $G = (V, E)$.

Kaveh [40] comparou os resultados dos seis algoritmos A a F. Os algoritmos foram aplicados em 13 exemplos e os dados referentes ao tempo da execução e à largura de nível foram mostrados. As simulações com os algoritmos A, D e E sempre encontraram as menores larguras de nível. Entre esses três, o mais rápido foi o algoritmo D. Os algoritmos que obtiveram os menores custos computacionais foram B e C. Nas comparações entre os algoritmos B e C, foram encontradas mais vezes a menor largura de nível com o algoritmo B. O algoritmo G é descrito por Kaveh [42, p. 230].

Algoritmo A

1. Construa a estrutura de nível enraizada para cada vértice $v \in V$.
2. Selecione aquele que possuir a menor largura de nível $b(\mathscr{L}(v))$.

Um pseudo-código para o algoritmo A é mostrado no algoritmo 22. O algoritmo recebe um grafo $G = (V, E)$. Os passos 1 e 2 são realizados em conjunto, na estrutura de repetição do algoritmo 22. O algoritmo retorna um vértice pseudo-periférico $u \in V$, na linha 11. Claramente, essa abordagem tem custo computacional alto para grafos com grande número de vértices e a sua aplicação pode ser inviável [42, p. 229].

Algoritmo 22: pseudo-periférico A.

```
   Entrada: grafo G = (V, E);
   Saída: vértice pseudo-periférico u ∈ V;
 1 início
 2     u ← ∅;
 3     largura ← +∞;
 4     para cada ( vértice v ∈ V ) faça
           // passo 1:  constrói-se a estrutura de nível enraizada
 5         ℒ(v) ← BuscaEmLargura(v); // de cada vértice
           // passo 2
 6         se ( b(ℒ(v)) < largura ) então
 7             u ← v;
 8             largura ← b(ℒ(v));
 9         fim-se;
10     fim-para-cada;
11     retorna u;
12 fim.
```

Algoritmo B

1. Selecione um vértice v de grau mínimo de V.
2. Construa a estrutura de nível enraizada de v.
3. Construa as estruturas de nível enraizadas de todos os vértices do último nível $L_{\ell(v)}(v)$ da estrutura de nível $\mathscr{L}(v)$.
4. Selecione aquela que apresentar a menor largura de nível. O processo de montagem da estrutura de nível enraizada para quando encontrar uma largura de nível maior que a largura de nível da estrutura de nível enraizada já selecionada.

Um pseudo-código para o algoritmo B é mostrado no algoritmo 23. O algoritmo recebe um grafo $G = (V, E)$. Os passos 1 e 2 são realizados, respectivamente, nas linhas 2 e 3. Os passos 3 e 4 são realizados em conjunto, no laço de repetição do algoritmo 23. O algoritmo retorna um vértice pseudo-periférico $u \in V$, na linha 14.

Algoritmo 23: pseudo-periférico B.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $v \in V$;

```
1  início
2      v ← VerticeComGrauMinimo(V); // passo 1
       // passo 2:  constrói-se a estrutura de nível 𝓛(v)
3      𝓛(v) ← BuscaEmLargura(v);
4      largura ← +∞;
5      para cada ( vértice w ∈ L_ℓ(v)(v) ) faça
6          𝓛(w) ← BuscaEmLargura(w); // passo 3
           // passo 4
7          se ( b(𝓛(w)) < largura ) então
8              v ← w;
9              largura ← b(𝓛(w));
10         senão se( (b(𝓛(v)) > largura ) então
11             break;
12         fim-se;
13     fim-para-cada;
14     retorna v;
15 fim.
```

Algoritmo C

1. Obtenha um vértice arbitrário $v \in V$.
2. Construa a estrutura de nível enraizada de v e selecione um vértice w de grau mínimo de $L_{\ell(v)}(v)$.
3. Construa a estrutura de nível enraizada de w.
4. Repita os passos 3 e 4 do algoritmo B, considerando $L_{\ell(w)}(w)$.

Um pseudo-código para o algoritmo C é mostrado no algoritmo 24. O algoritmo recebe um grafo $G = (V, E)$. O passo 1 é realizado na linha 2. O passo 2 é realizado nas linhas 3 e 4. O passo 3 é realizado na linha 5. O passo 4 é realizado nas linhas 7 a 15 do algoritmo 24. O algoritmo retorna um vértice pseudo-periférico $w \in V$, na linha 16.

Algoritmo 24: pseudo-periférico C.

```
   Entrada: grafo G = (V, E);
   Saída: vértice pseudo-periférico w ∈ V;
 1 início
 2     v ← VerticeArbitrario(V); // passo 1
       // passo 2
       // constrói-se estrutura de nível L(v)
 3     L(v) ← BuscaEmLargura(v);
 4     w ← VerticeComGrauMinimo(L_ℓ(v)(v));
       // passo 3:  constrói-se estrutura de nível L(w)
 5     L(w) ← BuscaEmLargura(w);
 6     largura ← +∞;
       // passo 3 do algoritmo B
 7     para cada ( vértice u ∈ L_ℓ(w)(w) ) faça
 8         L(u) ← BuscaEmLargura(u);
           // passo 4 do algoritmo B
 9         se ( b(L(u)) < largura ) então
10             w ← u;
11             largura ← b(L(u));
12         senão se( (b(L(w)) ≥ largura ) então
13             break;
14         fim-se;
15     fim-para-cada;
16     retorna w;
17 fim.
```

Algoritmo D

1. Construa a estrutura de nível enraizada de cada vértice $v \in V$.

2. Selecione a estrutura que possuir a menor largura de nível $b(\mathscr{L}(v))$. Diferentemente do algoritmo A, a construção da estrutura de nível enraizada para ao encontrar uma largura de nível maior ou igual que o já determinado.

Um pseudo-código para o algoritmo D é mostrado no algoritmo 25. O algoritmo recebe um grafo $G = (V, E)$. Os passos 1 e 2 são realizados em conjunto, no laço de repetição do algoritmo 25. O algoritmo retorna um vértice pseudo-periférico $u \in V$, na linha 17.

Algoritmo E

1. Obtenha um vértice arbitrário $v_1 \in V$ e omita todos os vértices adjacentes ao vértice v_1.

Algoritmo 25: pseudo-periférico D.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $u \in V$;

```
1  início
2      u ← v_1;
3      ℒ(v_1) ← BuscaEmLargura(v_1);
4      largura ← b(ℒ(v_1));
5      i ← 2;
6      trocou ← verdadeiro;
7      enquanto ( (i ≤ |V|) ∧ (trocou = verdadeiro) ) faça
           // passo 1:  constrói-se estrutura de nível ℒ(v_i)
8          ℒ(v_i) ← BuscaEmLargura(v_i);
           // passo 2
9          se ( b(ℒ(v_i)) < largura ) então
10             u ← v_i;
11             largura ← b(ℒ(v_i));
12         senão
13             trocou ← falso;
14         fim-se;
15         i ← i + 1;
16     fim-enquanto;
17     retorna u;
18 fim.
```

2. Selecione outro vértice arbitrário $v_2 \in V$ e omita todos os vértices adjacentes ao vértice v_1. Repita esse processo até que não haja mais vértices de V para serem obtidos.

3. Construa as estruturas de nível enraizadas para todos os vértices selecionados nos passos anteriores. Selecione aquele que possuir a menor largura de nível. Considere que o vértice escolhido é v.

4. Construa as estruturas de nível enraizadas para cada vértice adjacente ao vértice v e selecione aquele que possuir a menor largura de nível.

Um pseudo-código para o algoritmo E é mostrado no algoritmo 26. O algoritmo recebe um grafo $G = (V, E)$. O passos 1 e 2 são realizados nas linhas 2 a 11. Os passos 3 e 4 são realizados nas linhas 12 a 21 e 22 a 30, respectivamente. O algoritmo retorna um vértice pseudo-periférico $u \in V$, na linha 31.

Algoritmo F

1. Construa a estrutura de nível enraizada de um vértice arbitrário $v \in V$. Selecione o vértice $w_1 \in L_{\ell(v)}(v)$ com grau mínimo.

2. Construa a estrutura de nível enraizada do vértice w_1. Selecione o vértice $w_2 \in L_{\ell(w_1)}(w_1)$ com grau mínimo.

3. Construa as estruturas de nível de w_1 e de w_2. O par w_1 e w_2 é denominado par gerador de w_3. Encontre um vértice w_3 que se encontra em $L_{\ell(w_1)}(w_1)$ e em $L_{\ell(w_2)}(w_2)$. Construa a estrutura de nível enraizada de w_3.

Algoritmo 26: pseudo-periférico E.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $u \in V$;

início
  // início dos passos 1 e 2
  $G' \leftarrow G$; // $G' = (V', E')$
  $k \leftarrow 1$;
  $U \leftarrow \varnothing$;
  enquanto ($\exists v \in V'$ **) faça**
    // obtém um vértice arbitrário u_k de V' e insere-o em U
    $u_k \leftarrow VerticeArbitrario(V')$;
    $U \leftarrow U + \{u_k\}$;
    // remove-se o vértice u_k de V' e removem-se de V'
    // vértices adjacentes ao vértice u_k
    $V' \leftarrow V' - \{u_k\} - Adj(G', u_k)$, em que
    $Adj(G, u_k) = \{w \in V' : \{u_k, w\} \in E'\}$;
    // supõe-se que a remoção anterior dos vértices de V' não
    // removeu as arestas; por isso, removem-se de E' as
    // arestas incidentes a u_k
    $E' \leftarrow E' - \{\{u, w\} : \{u_k, w\} \in E'\}$;
    $k \leftarrow k + 1$;
  fim-enquanto;
  // há $k - 1$ vértices em U no final dos passos 1 e 2
  $largura \leftarrow +\infty$; // início do passo 3
  $v \leftarrow \emptyset$; // v recebe nulo
  $i \leftarrow 1$;
  enquanto ($i < k$ **) faça**
    // constrói-se a estrutura de nível $\mathscr{L}(u_i)$
    $\mathscr{L}(u_i) \leftarrow BuscaEmLargura(u_i)$;
    se ($b(\mathscr{L}(u_i)) < largura$ **) então**
      $v \leftarrow u_i$;
      $largura \leftarrow b(\mathscr{L}(u_i))$;
    fim-se;
  fim-enquanto;
  // v é o vértice escolhido no final do passo 3
  $largura \leftarrow +\infty$; // início do passo 4
  $u \leftarrow \emptyset$; // u recebe nulo
  // vértice com menor largura de banda e adjacente a v
  para cada (*vértice* $w \in Adj(G, v)$ **) faça**
    $\mathscr{L}(w) \leftarrow BuscaEmLargura(w)$;
    se ($b(\mathscr{L}(w)) < largura$ **) então**
      $u \leftarrow w$;
      $largura \leftarrow b(\mathscr{L}(w))$;
    fim-se;
  fim-para-cada;
  retorna u;
fim.

4. Repita o passo 3, utilizando-se os pares geradores (w_1, w_3) e (w_2, w_3) para encontrar w_4 e w_5, respectivamente. Construa as estruturas de nível enraizadas de w_4 e w_5.

5. Repita o passo 3 para w_i e w_{i+1}, podendo-se utilizar $i = 3, 4, \cdots, |V|-3$, para gerar $w_{i+3}, w_{i+4}, \cdots, w_{|V|}$.

6. Compare as estruturas de nível enraizadas montadas de w_i $(i = 1, 2, \cdots)$ e selecione o vértice que possuir a menor largura de nível.

Um pseudo-código para o algoritmo F é mostrado no algoritmo 27. O algoritmo recebe um grafo $G = (V, E)$ e o número k: haverá $w_1, w_2, w_3, w_4, w_5, \cdots, w_k$, em que $5 \leq k \leq |V|$. Os passos 1 e 2 são realizados nas linhas 3 a 5 e 6 e 7, respectivamente. O passo 3 é realizado nas linhas 8 a 10, ao se utilizar a sub-rotina $GeraNovoPseudoPeriferico$, mostrada no algoritmo 28. O passo 4 é realizado nas linhas 11 a 18. O passo 5 é realizado nas linhas 14 a 18. O passo 6 é realizado nas linhas 19 a 27. O algoritmo retorna um vértice pseudo-periférico $u \in V$, na linha 28.

Algoritmo G

1. Inicie por um vértice arbitrário v.

2. Construa a estrutura de nível enraizada $\mathscr{L}(v)$.

3. Obtenha o vértice w de grau mínimo em $L_{\ell(v)}(v)$.

4. Construa a estrutura de nível enraizada $\mathscr{L}(w)$.

5. Obtenha todos os vértices dos níveis zero, pares e último.

6. Construa a estrutura de nível enraizada de cada um desses vértices para encontrar a estrutura de nível com menor largura de nível. O processo de formação de estruturas de nível enraizadas é encerrado quando uma largura de nível de um vértice excede a largura de nível da estrutura de nível do vértice anterior. O vértice selecionado é u.

7. Verifique os vértices adjacentes ao vértice u por possíveis reduções na largura de nível para decidir o vértice final.

Um pseudo-código para o algoritmo G é mostrado no algoritmo 29. O algoritmo recebe um grafo $G = (V, E)$. Os passos 1, 2, 3 e 4 são realizados nas linhas 2, 3, 4 e 5, respectivamente. Os passos 5 e 6 são realizados nas linhas 6 a 17, ao se utilizar a sub-rotina $MenorLarguraDeNivel$, mostrada no algoritmo 30. O passo 7 é realizado nas linhas 18 a 24. O algoritmo retorna um vértice pseudo-periférico de V, na linha 25.

4.4 Algoritmo de Pachl

O processo inicia com um vértice $u \in V$, busca-se o vértice mais distante desse vértice. Isso é realizado ao se construir a estrutura de nível enraizada de u e obtém-se um vértice $v \in L_{\ell(u)}(u)$. Em seguida, gera-se a estrutura de nível enraizada de v e obtém-se um vértice $w \in L_{\ell(v)}(v)$. Depois, é analisada se a distância entre o vértice u e o seu vértice mais distante v é igual à distância de v ao seu vértice mais

Algoritmo 27: pseudo-periférico F.

Entrada: grafo $G = (V, E)$; valor k, em que $5 \leq k < |V|$;
Saída: vértice pseudo-periférico $v \in V$;

```
1  início
2      se ( k < 5 ∨ k > |V| − 1 ∨ |V| < 5 ) então vá para fim;
       // serão gerados k vértices geradores w_i, para 1 ≤ i ≤ k
3      v ← VerticeArbitrario(V); // início do passo 1
       // constrói-se a estrutura de nível ℒ(v)
4      ℒ(v) ← BuscaEmLargura(v);
5      w_1 ← VerticeComGrauMinimo(L_ℓ(v)(v));
       // início do passo 2
       // constrói-se a estrutura de nível
6      ℒ(w_1) ← BuscaEmLargura(w_1);
7      w_2 ← VerticeComGrauMinimo(L_ℓ(w_1)(w_1));
       // constrói-se a estrutura de nível
8      ℒ(w_2) ← BuscaEmLargura(w_2);
9      w_3 ← GeraNovoPseudoPeriferico(w_1, w_2); // passo 3
       // constrói-se a estrutura de nível
10     ℒ(w_3) ← BuscaEmLargura(w_3);
       // primeira parte do passo 4
11     w_4 ← GeraNovoPseudoPeriferico(w_1, w_3);
       // constrói-se a estrutura de nível
12     ℒ(w_4) ← BuscaEmLargura(w_4);
       // segunda parte do passo 4:  o par gerador (w_2, w_3) gera w_5
       // passo 5:  o par gerador (w_{j−2}, w_{j−1}) gera w_{j+1}
13     j ← 4;
14     enquanto ( j < k ) faça
15         w_{j+1} ← GeraNovoPseudoPeriferico(w_{j−2}, w_{j−1}); // algoritmo 28
           // constrói-se a estrutura de nível
16         ℒ(w_{j+1}) ← BuscaEmLargura(w_{j+1});
17         j ← j + 1;
18     fim-enquanto;
19     v ← w_1; // início do passo 6
20     largura ← b(ℒ(w_1));
21     i ← 2;
22     enquanto ( i ≤ k ) faça
23         se ( b(ℒ(w_i)) < largura ) então
24             v ← w_i;
25             largura ← b(ℒ(w_i));
26         fim-se;
27     fim-enquanto;
28     retorna v;
29 fim.
```

distante w, ou seja, o algoritmo verifica se $d(u, v) = d(v, w)$, para $u, v, w \in V$. Caso essa condição seja satisfeita, então, o vértice pseudo-periférico será v.

Esse algoritmo é $O(|E| \cdot \sqrt{|V|})$, conforme demonstrado por Pachl [69], em um grafo com V vértices e E arestas. O custo para se gerar a estrutura de nível, pela busca em largura, é $O(|E|)$ e isso é realizado $\sqrt{|V|}$ vezes.

Algoritmo 28: GeraNovoPseudoPeriferico.

Entrada: vértices geradores $w_a, w_b \in V$;
Saída: vértice gerador $w_c \in V$;

```
1  início
       // um par gerador (w_a, w_b) gera um novo vértice gerador w_c
       // constroem-se as estruturas de nível de w_a e de w_b
2      L(w_a) ← BuscaEmLargura(w_a);
3      L(w_b) ← BuscaEmLargura(w_b);
4      w_c ← ∅; // w_c recebe nulo
5      i ← 1; // percorre-se L_{ℓ(w_a)}(w_a)
6      j ← 1; // percorre-se L_{ℓ(w_b)}(w_b)
7      encontrou ← falso;
       // determina-se o vértice w_c ∈ L_{ℓ(w_a)}(w_a) ∧ L_{ℓ(w_b)}(w_b)
8      enquanto ( i ≤ |L_{ℓ(w_a)}(w_a)| ∧ não encontrou ) faça
9          enquanto ( j ≤ |L_{ℓ(w_b)}(w_b)| ∧ não encontrou ) faça
               // considere que L_{ℓ(w_a)}(w_a) = {w_{a_1}, w_{a_2}, ..., w_{a_n}} e
               // n = |L_{ℓ(w_a)}(w_a)|
               // considere que L_{ℓ(w_b)}(w_b) = {w_{b_1}, w_{b_2}, ..., w_{b_m}} e
               // m = |L_{ℓ(w_b)}(w_b)|
10             se ( w_{a_i} = w_{b_j} ) então
11                 w_c ← w_{a_i};
12                 encontrou ← verdadeiro;
13             fim-se;
14             j ← j + 1;
15         fim-enquanto;
16         i ← i + 1;
17     fim-enquanto;
18     retorna w_c;
19 fim.
```

O pseudo-código é mostrado no algoritmo 31. Inicializa-se a variável v_0 com um vértice arbitrário na linha 2 e gera-se $\mathscr{L}(v_0)$ enraizada de v_0 na linha 3. Obtém-se um vértice do último nível de $\mathscr{L}(v_0)$ na linha 4 e armazena-se esse vértice em v_1. Em seguida, gera-se a estrutura de nível enraizada de v_1 e de um vértice $v_2 \in L_{\ell(v_1)}(v_1)$. Repete-se esse processo, na estrutura de repetição das linhas 6 a 10, até que $d(v_j, v_{j-1}) = d(v_{j-1}, v_{j-2})$. O algoritmo retorna v_{j-1}.

Considere o grafo da figura 4.4. A escolha do vértice 4 destacado na figura foi arbitrária. Como pode ser visto na figura 4.4, os vértices no último nível da estrutura de nível enraizada do vértice 4 são os vértices 2, 3, 5, 7 e 8, todos com distância 2, ou seja, $\ell(4) = 2$. A escolha entre vértices com a mesma distância pelo algoritmo é indiferente e escolheu-se o vértice 8.

No exemplo, a procura pelo vértice mais distante é realizada, conforme observado no algoritmo 31, no vértice 8. O vértice mais distante escolhido entre as opções é o vértice 5, possuindo uma distância 3, como pode ser observado na figura 4.5. O algoritmo verifica se $d(2, 8) = d(8, 5)$. Como $d(2, 8) = 2$ e $d(8, 5) = 3$, o algoritmo repetirá o processo.

É realizada a busca pelo vértice mais distante do vértice 5. Conforme apresentado na figura 4.6, o vértice 10, com distância 3, é escolhido como o mais distante. Como $d(8, 5) = d(5, 10)$, o vértice 5 é escolhido como o vértice pseudo-periférico.

Algoritmo 29: pseudo-periférico G.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $u \in V$;

```
1  início
2      v ← VerticeArbitrario(V); // passo 1
       // passo 2:  constrói-se estrutura de nível ℒ(v)
3      ℒ(v) ← BuscaEmLargura(v);
4      w ← VerticeComGrauMinimo(L_ℓ(v)(v)); // passo 3
5      ℒ(w) ← BuscaEmLargura(w); // passo 4
6      largura ← b(ℒ(w)); // início dos passos 5 e 6
7      i ← 2; // percorrem-se os níveis pares de ℒ(w)
8      ret[1] ← w;
9      ret[2] ← b(ℒ(w));
10     ret[3] ← verdadeiro;
11     enquanto ( i < ℓ(w) ∧ ret[3] = verdadeiro ) faça
12         ret ← MenorLarguraDeNivel(L_i(w), ret[1], ret[2]);
13         i ← i + 2;
14     fim-enquanto;
       // trata L_ℓ(w)(w)
15     se ( ret[3] = verdadeiro ) então
16         ret ← MenorLarguraDeNivel(L_ℓw(w), ret[1], ret[2]);
17     fim-se;
       // ret[1] contém o vértice u do passo 6 e ret[2] contém a
       // largura de nível de ℒ(u)
       // passo 7
18     para cada ( vértice w ∈ Adj(G, ret[1]) ) faça
19         ℒ(w) ← BuscaEmLargura(w);
20         se ( b(ℒ(w) < ret[2] ) então
21             ret[1] ← w;
22             ret[2] ← b(ℒ(w));
23         fim-se;
24     fim-para-cada;
25     retorna ret[1];
26 fim.
```

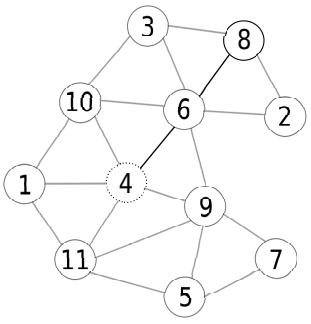

Figura 4.4: Escolha do vértice u.

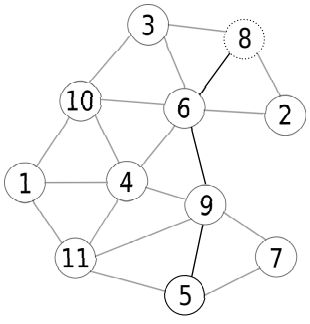

Figura 4.5: Escolha do vértice v.

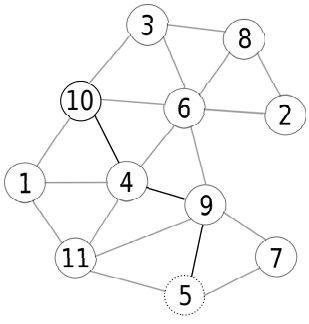

Figura 4.6: Escolha do vértice w.

Algoritmo 30: *MenorLarguraDeNivel*.

Entrada: estrutura de nível enraizada $\mathscr{L}(w)$; vértice u; largura de nível $b(\mathscr{L}(u))$;
Saída: vetor ret com três entradas, $ret[1]$ com vértice pseudo-periférico u, $ret[2]$ com a largura de nível de $\mathscr{L}(u)$ e $ret[3]$ com falso se ocorre que a largura de nível de algum vértice em $\mathscr{L}(w)$ é maior que a largura de nível encontrada na iteração anterior ou verdadeiro em caso contrário;

```
// determina o vértice w' com a menor largura de nível b em
// L(w) e se b < b(L(u))
```

```
1  início
2      ret[1] ← u;
3      ret[2] ← b(L(u));
4      ret[3] ← verdadeiro;
5      para cada ( vértice w' ∈ L(w) ) faça
6          L(w') ← BuscaEmLargura(w');
7          se ( b(L(w')) < b(L(u)) ) então
8              ret[1] ← w';
9              ret[2] ← b(L(w');
10         senão se( b(L(w')) > b(L(u)) ) então
11             ret[3] ← falso;
12             break;
13         fim-se;
14     fim-para-cada;
15     retorna ret;
16 fim.
```

Algoritmo 31: Pachl.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $v \in V$;

```
1  início
2      v0 ← VerticeArbitrario(V);
       // constrói-se estrutura de nível enraizada
3      L(v0) ← BuscaEmLargura(v0);
       // obtém-se uma vértice de L_{ℓ(v0)}(v0)
4      v1 ← ObtemVertice(L_{ℓ(v0)}(v0));
5      j ← 1;
6      repita
           // constrói-se estrutura de nível enraizada
7          L(vj) ← BuscaEmLargura(vj);
           // obtém-se uma vértice de L_{ℓ(vj)}(vj)
8          v_{j+1} ← ObtemVertice(L_{ℓ(vj)}(vj));
9          j ← j + 1;
10     até que d(vj, v_{j-1}) = d(v_{j-1}, v_{j-2}) ;
11     retorna (v_{j-1});
12 fim.
```

Nesse exemplo, o vértice pseudo-periférico escolhido é um vértice periférico. Isso

ocorreu porque o grafo é pequeno. Para finalizar, mostra-se, na figura 4.7, a numeração do grafo da figura 4.1 juntamente com sua matriz correspondente após a ordenação realizada pela heurística Cuthill-McKee, iniciando-se por um vértice pseudo-periférico, o vértice 5.

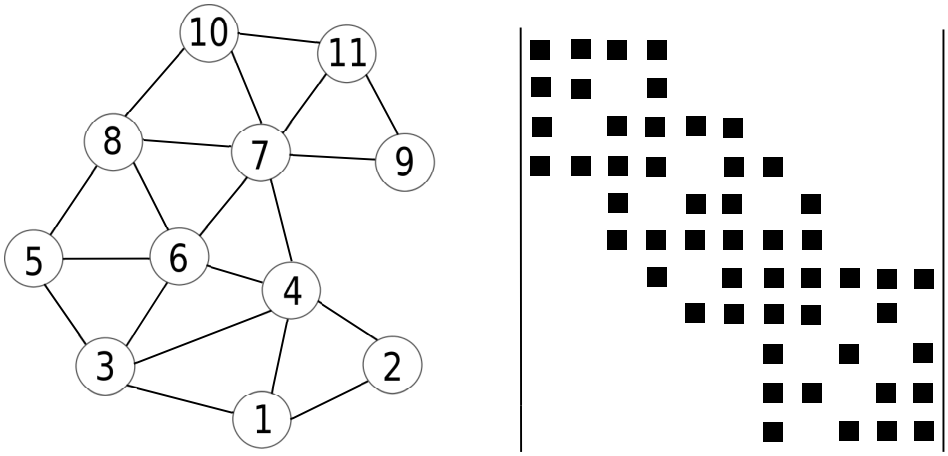

Figura 4.7: Grafo ordenado pela heurística CM com a escolha do vértice pseudo-periférico.

4.5 Algoritmo razão-largura-profundidade

Proposto por Wang et al. [90, 89], esse algoritmo for criado para encontrar vértices iniciais para a heurística GPS. Nos testes realizados por Wang et al. [90], os resultados em relação às excentricidades dos vértices pseudo-periféricos encontrados pelo algoritmo razão-largura-profundidade foram comparados com as excentricidades dos vértices pseudo-periféricos encontrados na primeira etapa da heurística GPS [26]. Em 12 instâncias testadas, o algoritmo razão-largura-profundidade encontrou, nas 12 instâncias, vértices com excentricidades maiores que as excentricidades dos vértices encontrados pela primeira etapa da heurística GPS.

O algoritmo de Wang et al. [90] utiliza um parâmetro chamado de *width-depth-ratio*. Para se definir esse parâmetro, utiliza-se a observação de que, quanto maior o comprimento da estrutura de nível de um vértice, e quanto mais estreita a estrutura de nível é, melhores soluções são geradas na redução de largura de banda. Esse algoritmo busca dois vértices que possuam estrutura de nível com mais níveis e a largura de nível mais estreita possível.

No algoritmo de Wang et al. [90], primeiramente, são selecionados vértices com grau mínimo. Esses vértices compõem o conjunto de vértices pré-candidatos. Em seguida, geram-se as estruturas de nível enraizadas de cada um desses vértices e calculam-se as razões entre as larguras de nível e as excentricidades para cada estrutura de nível enraizada dos vértices do conjunto de pré-candidatos. Escolhem-se os vértices que possuírem a menor razão do conjunto de vértices pré-candidatos. Esses vértices são colocados em um *novo* conjunto de vértices candidatos.

Se houver apenas um vértice u no conjunto de vértices candidatos, então, seleciona-se um vértice $v \in L_{\ell(u)}(u)$ com a menor razão entre a largura de nível e a excentricidade da estrutura de nível enraizada. Com isso, os vértices u e v são selecionados

como vértices pseudo-periféricos. Em ocorrendo dois vértices u e v no conjunto de vértices candidatos e se $u \in L_{\ell(v)}(v)$ e $v \in L_{\ell(u)}(u)$, u e v são escolhidos como vértices pseudo-periféricos.

Em caso contrário, seleciona-se, no conjunto de vértices candidatos, o vértice v com a menor numeração e seleciona-se um vértice $u \in L_{\ell(v)}(v)$ com a menor razão entre a largura de nível e a excentricidade da estrutura de nível enraizada. Os vértices u e v são selecionados como vértices pseudo-periféricos.

Mostra-se o pseudocódigo do algoritmo de Wang et al. [90] no algoritmo 32. Define-se o conjunto *Candidados* de vértices de grau mínimo e menor razão largura-profundidade na estrutura de repetição das linhas 5 a 17. Se o conjunto *Candidados* contém um só elemento, isso é tratado nas linhas 18 a 21: retorna-se um conjunto composto pelo vértice $v \in Candidatos$ e pelo vértice com a menor razão largura-profundidade de $\mathscr{L}_{\ell(v)}(v)$. Esse vértice é encontrado pelo algoritmo 33. Se o conjunto *Candidados* possui apenas dois vértices u e v e $u \in L_{\ell(v)}(v)$ e $v \in L_{\ell(u)}(u)$, então, retorna-se $\{u, v\}$ na linha 27. Em correndo que o conjunto *Candidados* possua mais de dois vértices, então, escolhe-se, na linha 29, um vértive v que possuir a menor numeração. Em seguida, seleciona-se um vértice $u \in L_{\ell(v)}(v)$ que possuir a menor razão largura-profundidade. Esses dois vértices são retornados na linha 31.

4.6 Exercícios

1. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo A de Kaveh, mostrado na seção 4.3.

2. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo B de Kaveh, mostrado na seção 4.3.

3. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo C de Kaveh, mostrado na seção 4.3.

4. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo D de Kaveh, mostrado na seção 4.3.

5. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo E de Kaveh, mostrado na seção 4.3.

6. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo F de Kaveh, mostrado na seção 4.3.

7. Encontre o vértice pseudo-periférico do grafo da figura 4.1 pelo algoritmo G de Kaveh, mostrado na seção 4.3.

8. Encontre os dois vértices pseudo-periféricos do grafo da figura 4.1 pelo algoritmo de razão-largura-profundidade, mostrado na seção 4.5.

Algoritmo 32: Razão-largura-profundidade.

Entrada: grafo $G = (V, E)$;
Saída: conjunto U de dois vértices pseudo-periféricos $v, u \in V$;

início
  // grau do vértice com grau mínimo
  $grauMinimo \leftarrow +\infty$;
  $Candidatos \leftarrow \varnothing$;
  $razao_{menor} \leftarrow +\infty$;
  // monta o conjunto *Candidados*
  para cada (*vértice* $w \in V$) **faça**
    se ($Grau(G, w) \leq grauMinimo$) **então**
      $\mathscr{L}(w) \leftarrow BuscaEmLargura(w)$;
      $w.razao \leftarrow b(\mathscr{L}(w))$ / $\ell(w)$;
      se ($(w.razao < razao_{menor}) \vee (Grau(G, w) < grauMinimo)$) **então**
        $razao_{menor} \leftarrow w.razao$;
        $Candidatos \leftarrow \{w\}$;
      senão se($w.razao = razao_{menor}$ **) então**
        $Candidatos \leftarrow Candidatos \cup \{w\}$;
      fim-se;
      $grauMinimo \leftarrow Grau(G, w)$;
    fim-se;
  fim-para-cada;
  se ($|Candidatos| = 1$) **então**
    // obtém-se o vértice do conjunto
    $v \leftarrow EscolheVertice(Candidatos, 1)$;
    $\mathscr{L}(v) \leftarrow BuscaEmLargura(v)$;
    // retorna v e o vértice com a menor razão entre
    // largura de nível e excentricidade de $L_{\ell(v)}(v)$
    retorna $\{v\} \cup VerticeMenorRazao(L_{\ell(v)}(v))$;
  senão se($|Candidatos| = 2$ **) então**
    // obtém-se o primeiro vértice do conjunto
    $v \leftarrow EscolheVertice(Candidatos, 1)$;
    // obtém-se o segundo vértice do conjunto
    $u \leftarrow EscolheVertice(Candidatos, 2)$;
    $\mathscr{L}(v) \leftarrow BuscaEmLargura(v)$;
    $\mathscr{L}(u) \leftarrow BuscaEmLargura(u)$;
    se ($u \in L_{\ell(v)}(v) \wedge v \in L_{\ell(u)}(u)$) **então retorna** $\{u, v\}$;
  fim-se;
  // obtém-se o vértice com a menor numeração
  // no conjunto $Candidatos$
  $v \leftarrow EscolheVerticeMenorRotulo(Candidatos)$;
  $\mathscr{L}(v) \leftarrow BuscaEmLargura(v)$;
  // retorna v e o vértice com a menor razão entre
  // largura de nível e excentricidade de $L_{\ell(v)}(v)$
  retorna $\{v\} \cup VerticeMenorRazao(L_{\ell(v)}(v))$;
fim.

Algoritmo 33: $VerticeMenorRazao$.

Entrada: nível $L_i(v) \in \mathscr{L}(v)$;
Saída: vértice pertencente a $L_i(v)$ com menor razão entre largura de nível e excentricidade;

```
1  início
2      razao_menor ← +∞;
3      para cada ( vértice w ∈ L_i(v) ) faça
           // contrói-se a estrutura de nível
4          ℒ(w) ← BuscaEmLargura(w);
5          razao ← b(ℒ(w)) / ℓ(w);
6          se ( razao < razao_menor ) então
7              razao_menor ← razao;
8              verticeMenorRazao ← w;
9          fim-se;
10     fim-para-cada;
11     retorna verticeMenorRazao;
12 fim.
```

Capítulo 5

Implementações da heurística Cuthill-McKee reverso

5.1 Introdução

As reduções de largura de banda e de *profile* de matrizes podem proporcionar redução no custo computacional na resolução de sistemas de equações lineares por métodos baseados no subespaço de Krylov [48]. Como exemplo de aplicação de uma heurística para as reduções de largura de banda e de *profile*, implementou-se a heurística Cuthill-McKee reverso para se reduzirem a largura de banda e o *profile* de matrizes de sistemas de equações lineares. Com isso, verificou-se o desempenho computacional do método dos gradientes conjugados [37, 49] para resolver sistemas de equações lineares. Neste capítulo, são apresentados detalhes e os resultados dessa implementação.

Neste capítulo, apresentam-se implementações da heurística Cuthill-McKee reverso (CMr) e uma variação dessa heurística, chamada aqui de CMr-pseudo. Nessa variação da heurística CMr, o vértice inicial é um vértice pseudo-periférico obtido pelo algoritmo de George e Liu [21], mostrado na seção 4.2, na página 70. As heurísticas CMr e CMr-pseudo foram utilizadas para reduzirem a largura de banda e o *profile* de matrizes de sistemas de equações lineares oriundas de uma resolução da equação de Laplace pelo método dos volumes finitos. Com isso, verificou-se o desempenho computacional do método dos gradientes conjugados para resolver esses sistemas de equações lineares.

Para exemplificar a utilização de uma heurística para as reduções de largura de banda e de *profile*, implementou-se a heurística CMr porque é uma das heurísticas mais tradicionais para os problemas das reduções de largura de banda e de *profile*. Por causa da sua simplicidade e por ser eficiente, a heurística CMr iniciada por um vértice pseudo-periférico tornou-se um padrão em muitas simulações computacionais [73, p. 107]. Mesmo sendo uma das primeiras heurísticas a ser proposta, a heurística CMr é utilizada em pesquisas atuais, por exemplo, Kazoela, Nikolos e Synolakis [47] utilizaram a redução de largura de banda obtida pela heurística CMr para se reduzir o custo computacional do pré-condicionador *incomplete LU algorithm with threshold* (ILUT) [77, p. 286-297]. Como ênfase para a importância da heurística CMr, essa heurística é implementada nos resolutores (combinados com pré-condicionadores) mais populares de sistemas com matrizes simétricas e positivas-definidas [23].

Nas implementações, utilizou-se o algoritmo de George e Liu [21] para encontrar

um vértice pseudo-periférico inicial para a heurística CMr. O algoritmo de George e Liu [21] foi utilizado por ser um dos algoritmos clássicos para se encontrar um vértice pseudo-periférico e por apresentar melhorias em relação ao algoritmo original de Gibbs, Poole e Stockmeyer [26].

Na seção 5.2, apresentam-se detalhes das implementações e as ferramentas utilizadas. Mostram-se as simulações na seção 5.3. Na subseção 5.4, apresentam-se considerações sobre as simulações.

5.2 Detalhes da implementação e ferramentas utilizadas

O objetivo do projeto computacional criado para as simulações publicadas em Oliveira e Gonzaga de Oliveira [68] é solucionar a equação de Laplace no domínio computacional. O projeto computacional utilizado para acoplar a heurística CMr foi desenvolvido em C++. Nesse projeto computacional, geram-se triangulações de Delaunay em um domínio quadrado, para a solução da equação de Laplace com as condições de contorno: $T(a,0) = 0$ para $0 < a < 1$ e $T(a,1) = T(0,a) = T(1,a) = 10$ para $0 \leq a \leq 1$.

Para a discretização da equação diferencial parcial, utilizou-se o método dos volumes finitos. Polígonos de Voronoi foram utilizados como volumes de controle.

Cada vértice da triangulação, exceto os vértices da borda, recebem um número. Por meio da ordem dessa numeração, gera-se o sistema de equações lineares. Soluciona-se o sistema de equações lineares pelo método dos gradientes conjugados. Refina-se a malha pelo refinamento de Ruppert [76]. Malhas de qualidade são geradas por *off-centers* de Üngör [85, 86]. A triangulação de Delaunay e o diagrama de Voronoi são detalhados em Nogueira e Gonzaga de Oliveira [67], por exemplo.

No projeto computacional, os vértices da malha (ou grafo) são ligados por uma lista encadeada. Gera-se o sistema de equações lineares conforme a ordem dessa lista encadeada. No projeto computacional sem as reduções de largura de banda e de *profile* pela heurística CMr ou pela heurística CMr-pseudo, os novos vértices gerados pelo refinamento são inseridos no final dessa lista encadeada. Isso gera um sistema de equações lineares com matriz de coeficientes com largura de banda e *profile* grandes. No projeto computacional, para se reduzirem a largura de banda e o *profile* do sistema de equações lineares, permuta-se a ordem dos vértices nessa lista encadeada pela heurística CMr.

Um exemplo de triangulação de Delaunay, com 19 vértices, é mostrado na figura 5.1. Nessa malha, não se utilizou a heurística CMr para as reduções de largura de banda e de *profile*. Há vértices com numeração baixa adjacentes a vértices com numeração alta. Um exemplo disso é o vértice 1, que é adjacente ao vértice 18. Isso faz com que a largura de banda e o *profile* da matriz de adjacências sejam grandes. Na malha da figura 5.1, os vértices da borda recebem rótulo -1. Claramente, esses vértices não fazem parte do sistema de equações lineares.

Mostra-se, na figura 5.2, o diagrama de atividades do projeto computacional com a heurística CMr. O programa começa com uma malha inicial e com configurações iniciais. Em seguida, gera-se o sistema de equações lineares referente à malha inicial pelo método dos volumes finitos. Claramente, a heurística CMr foi inserida entre a montagem do sistema de equações lineares e a resolução pelo método dos gradientes conjugados, conforme é mostrado na figura 5.2. Depois da resolução do sistema de equações lineares pelo método dos gradientes conjugados, se os critérios para o refinamento adaptativo não forem satisfeitos, o programa para. Se os critérios para

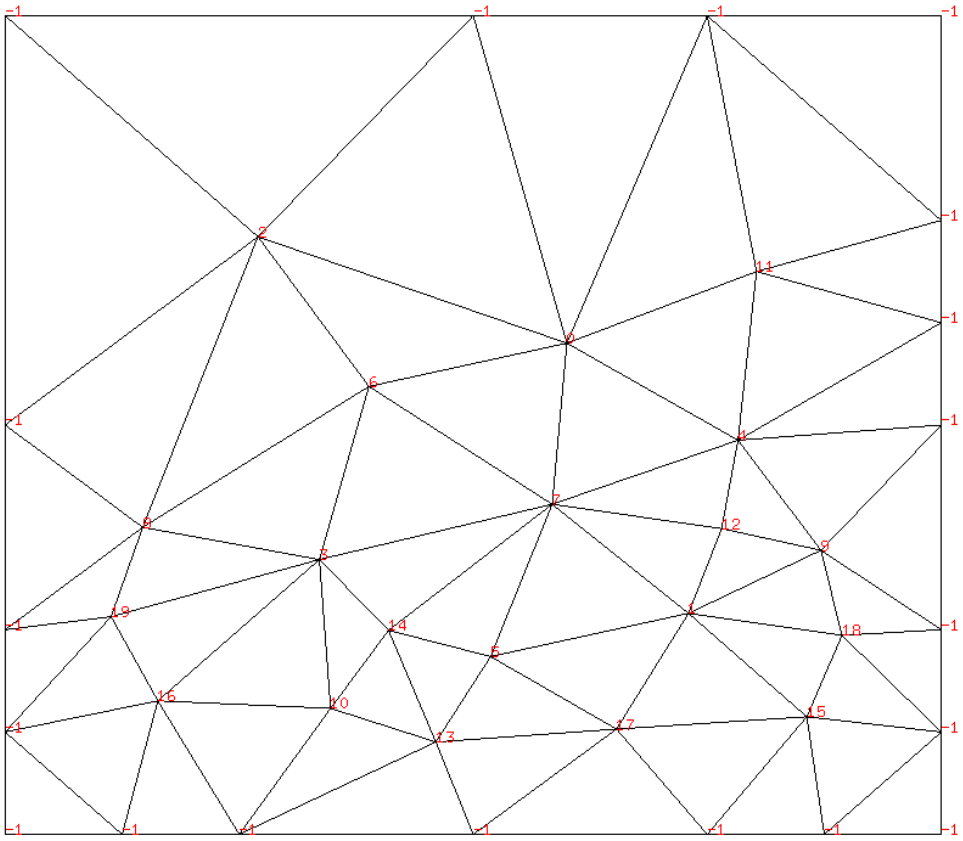

Figura 5.1: Triangulação de Delaunay, em que os 19 vértices internos estão numerados.

o refinamento adaptativo forem satisfeitos, a malha é refinada pelo refinamento de Ruppert [76]. Em seguida, refina-se a malha por *off-centers*, gerando-se uma malha com a qualidade definida pelo usuário. No projeto computacional, utiliza-se a medida de qualidade de triângulos *circunradius-to-shortest-edge-ratio*. Isso significa que uma malha é dita de qualidade se todos os ângulos de todos os triângulos são menores que $arcsin\frac{1}{2B}$. O limite B deve ser maior que a razão do raio do circuncírculo do triângulo pelo comprimento da menor aresta do triângulo. Finalmente, retorna-se ao passo de montagem do sistema de equações lineares.

Nas simulações, utilizou-se um computador IBM® x3400 M3 com processador Intel® Xeon® E5620 2.40GHz com 12Mb de memória *cache*, 16Gb de memória RAM DDR3 1333MHz. O sistema operacional utilizado foi o Ubuntu 12.04 LTS 64bits com *kernel* versão 3.2.0-38-*generic*. Utilizou-se a IDE NetBeans 7.2.1 com *plugin* para C/C++ para a implementação dos projetos computacionais.

5.3 Testes experimentais

Nesta subseção, são apresentados as simulações com o método dos gradientes conjugados. Em particular, utiliza-se a aproximação anterior para se obter as aproximação seguinte. Desse modo, o método dos gradientes conjugados converge com

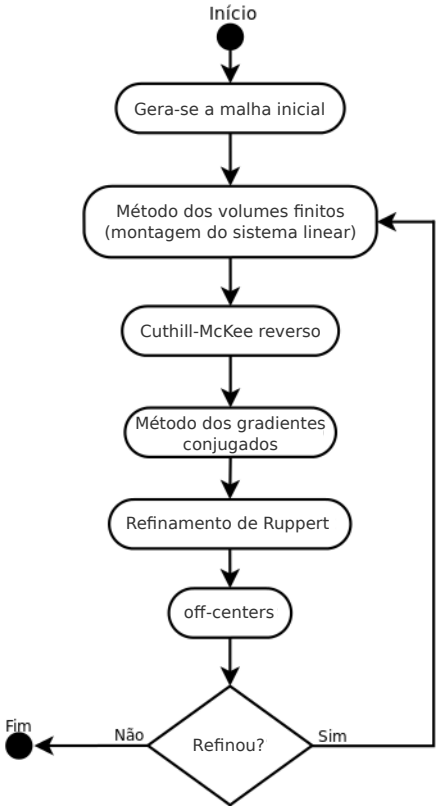

Figura 5.2: Diagrama de atividades do projeto computacional.

um número razoável de iterações.

Aplicou-se o método dos gradientes conjugados 10 vezes na mesma iteração e verificou-se o tempo em cada execução. Fez-se a média, em segundos, dessas 10 execuções. Como o tempo de execução não variou muito (desvio padrão, em média, 0,1), considerou-se que dez execuções foram suficientes em cada teste. Os tempos de execuções da heurística CMr e da heurística CMr-pseudo não foram sumarizados, pois não foram considerados significativos. Por exemplo, nas simulações com 277948 vértices, as heurísticas CMr e CMr-pseudo apresentaram tempo médio de execução menor que um segundo.

Os detalhes das simulações são mostrados nas subseções seguintes. Na subseção 5.3.1, mostram-se dados das simulações sem a heurística CMr. Apresentam-se os dados da execução do projeto computacional com a heurística CMr e sem o algoritmo que encontra um vértice pseudo-periférico na subseção 5.3.2. Os dados da execução do projeto computacional com a heurística CMr iniciando por um vértice pseudo-periférico são apresentados na subseção 5.3.3. Na subseção 5.3.4, apresentam-se exemplos de redução de largura de banda em matrizes de um projeto computacional para a solução da equação de Laplace.

5.3.1 Simulações sem a heurística Cuthill-McKee reverso

Mostram-se os dados das execuções do método dos gradientes conjugados no projeto computacional sem a utilização da heurística CMr na tabela 5.1. Na primeira

coluna, mostram-se os números de vértices das malhas. Mostram-se, na segunda coluna, em segundos, os tempos médios do método dos gradientes conjugados para solucionar os sistemas de equações lineares com tamanhos referentes aos números de vértices. São mostrados, na terceira coluna, os números de iterações que o método dos gradientes conjugados utilizou para solucionar os sistemas de equações lineares. Mostram-se as larguras de banda e os *profiles* referentes aos sistemas de equações lineares na quarta e quinta coluna, respectivamente.

Resoluções de sistemas de equações lineares sem as reduções de largura de banda e de *profile*				
Vértices	GC: tempo (s)	GC: iterações	β	*profile*
4885	0,548	401	4460	7741207
12203	3,114	649	11726	52697023
32508	21,998	1069	31722	393718964
54165	53,096	1367	53158	1111334245
89320	128,911	1771	88189	3078204659
130824	242,944	2151	129413	6611779836
178852	395,145	2487	177130	12166024388
223993	561,592	2765	221797	18573739773
256125	694,341	2957	253922	23535198622
277982	765,054	2963	275813	26692399231

Tabela 5.1: Dados referentes às execuções do método dos gradientes conjugados e aos sistemas de equações lineares sem as reduções de largura de banda e de *profile*.

Utilizou-se o mesmo sistema de equações lineares nas dez execuções do método dos gradientes conjugados. Logo, não houve variação no número de iterações desse método, em uma mesma instância. O mesmo ocorreu com a largura de banda e com o *profile*.

5.3.2 Simulações com a heurística Cuthill-McKee reverso

Mostram-se, na tabela 5.2, os dados das execuções do método dos gradientes conjugados no projeto computacional com a utilização da heurística CMr para as reduções de largura de banda e de *profile* sem a utilização do algoritmo que encontra um vértice pseudo-periférico. Ao se observar os números de iterações mostrados nas tabelas 5.1, 5.2 e 5.3, nota-se que quase não houve variação no número de iterações do método dos gradientes conjugados nos diferentes testes.

Também nota-se que os tempos médios de execução do método dos gradientes conjugados com a utilização da heurística CMr, mostrados na tabela 5.2, foram menores que os tempos médios das simulações do projeto computacional sem a utilização da heurística CMr, mostrados na tabela 5.1. Na tabela 5.2, a terceira, a sexta e a oitava colunas são as porcentagens das reduções dos tempos médios das execuções do método dos gradientes conjugados, largura de banda e de *profile*, respectivamente, em relação à tabela 5.1.

5.3.3 Simulações com a heurística CMr-pseudo

Na tabela 5.3, mostram-se os dados das execuções do método dos gradientes conjugados no projeto computacional com a utilização da heurística CMr para as reduções

Resoluções de sistemas de equações lineares com as reduções de largura de banda e de *profile* pela heurística CMr							
Vértices	GC: tempo (s)	Redução	GC: iterações	β	Redução	*profile*	Redução
4885	0,512	6,56%	402	165	96,30%	467294	93,96%
12203	3,038	2,44%	650	281	97,60%	1900641	96,39%
32511	20,256	7,91%	1075	457	98,56%	8399981	97,87%
54174	47,563	10,42%	1370	590	98,89%	17898481	98,38%
89339	116,485	9,64%	1770	747	99,15%	37754045	98,77%
130854	219,812	9,52%	2165	937	99,27%	72310418	98,91%
178884	354,85	10,19%	2478	1139	99,35%	122995495	98,99%
224003	504,656	10,13%	2753	1288	99,42%	179790220	99,03%
256106	630,729	9,16%	2955	1387	99,45%	222104335	99,05%
277948	692,685	9,46%	2938	1457	99,47%	248862315	99,06%

Tabela 5.2: Dados referentes às execuções do método dos gradientes conjugados e ao sistema de equações lineares com as reduções de largura de banda e de *profile* pela heurística CMr, comparados com os resultados mostrados na tabela 5.1.

de largura de banda e de *profile*, iniciando-se por um vértice pseudo-periférico. Nas simulações com a heurística CMr-pseudo, os tempos médios em todas as execuções do método dos gradientes conjugados foram menores que os tempos médios das execuções do método dos gradientes conjugados, mostradas na tabela 5.1.

Resoluções de sistemas lineares com as reduções de largura de banda e de *profile* pela heurística CMr-pseudo							
Vértices	GC: tempo (s)	Redução	GC: iterações	β	Redução	*profile*	Redução
4885	0,499	2,53%	406	104	36,96%	279997	40,08%
12203	2,875	5,36%	644	141	49,82%	1028302	45,89%
32511	19,994	1,29%	1070	249	45,51%	4619380	45,01%
54174	47,501	0,13%	1368	337	42,88%	10129287	43,41%
89339	115,97	0,44%	1769	409	45,25%	21656656	42,63%
130853	218,165	0,75%	2147	532	43,22%	41899637	42,05%
178884	354,058	0,22%	2475	642	43,63%	65188256	46,99%
224002	506,245	-0,31%	2770	696	45,96%	91301582	49,22%
256103	635,599	-0,77%	2963	1379	0,57%	221039305	0,47%
277948	697,497	-0,69%	2944	1440	1,18%	245801797	1,22%

Tabela 5.3: Dados referentes às execuções do método dos gradientes conjugados e ao sistema de equações lineares com as reduções de largura de banda e de *profile* pela heurística CMr iniciando-se por um vértice pseudo-periférico, comparados com os resultados mostrados na tabela 5.2.

Com exceção dos testes com 224002, 256103 e 277948 vértices, mostrados na tabela 5.3, os tempos médios do método dos gradientes conjugados, nas simulações com a heurística CMr-pseudo, foram poucos menores que os tempos médios do método dos gradientes conjugados com a heurística CMr, mostrados nas linhas correspondentes da tabela 5.2. Note que ocorreram algumas diferenças nos números de vértices nas simulações correspondentes mostradas nas tabelas 5.1, 5.2 e 5.3, provavelmente, por causa dos efeitos de arredondamentos numéricos no refinamento

da malha.

Nessas simulações, a heurística CMr-pseudo reduziu mais a largura de banda e o *profile* do que a heurística CMr. Mostram-se, nas colunas 3, 6 e 8 da tabela 5.3, as porcentagens das reduções do tempo de execução do método dos gradientes conjugados, largura de banda e de *profile* em relação aos dados mostrados na tabela 5.2. Mesmo assim, como pode ser observado na tabela 5.3, com 224002, 256103 e 277948 vértices, o método dos gradientes conjugados, após a execução da heurística CMr-pseudo, apresentou tempos médios maiores que os tempos médios após a execução da heurística CMr.

No teste com 224003 na tabela 5.2 e 224002 na tabela 5.3, houve 45,96% e 49,22% de reduções de largura de banda e de *profile*, respectivamente, e aumento do custo computacional do método do gradientes conjugados na utilização da heurística CMr-pseudo em relação à utilização da heurística CMr. Nos testes com 256106 vértices na tabela 5.2 e 256103 vértices na tabela 5.3 e no teste com 277948 vértices em ambas as tabelas, as larguras de banda e os *profiles* não foram muito menores em relação aos resultados apresentados na tabela 5.2, como nas demais instâncias. Isso ocorreu porque, nas instâncias com 256106 vértices na tabela 5.2 e 256103 vértices na tabela 5.3 e a instância com 277948 vértices em ambas as tabelas, as excentricidades (ℓ) dos vértices encontrados pelo algoritmo de George e Liu [21] não foram muito maiores que as excentricidades dos vértices iniciais utilizados pela heurística CMr. Isso pode ser visto na tabela 5.4. Para a definição de excentricidade de vértice, veja a definição 6, na página 8. Na primeira coluna da tabela 5.4 há o tamanho das instâncias mostradas na tabela 5.3. Na segunda coluna, há as excentricidades dos vértices iniciais nas simulações com a heurística CMr. Finalmente, na terceira coluna, há as excentricidades dos vértices iniciais utilizados nas simulações com a heurística CMr-Pseudo. Como foram utilizados vértices arbitrários como vértices iniciais nas simulações com a heurística CMr, esses vértices foram vértices pseudo-periféricos nessas simulações.

Vértices (aprox.)	ℓ (CMr)	ℓ (CMr-Pseudo)
4885	60	78
12203	95	151
32511	153	235
54174	192	297
89339	252	375
130853	287	417
178884	314	516
224002	339	580
256103	350	362
277948	351	378

Tabela 5.4: Excentricidades dos vértices inciais nas simulações com a heurística CMr com e sem o algoritmo para se encontrar um vértice pseudo-periférico. Nas linhas correspondentes da tabela 5.2, houve um vértice a mais nas linhas com 130853 e 224002 vértices e três vértices a mais na linha com 256103 vértices.

5.3.4 Exemplos de largura de banda em matrizes das simulações

Mostram-se alguns exemplos de matrizes de coeficientes dos sistemas de equações lineares do projeto computacional. Na figura 5.3, mostra-se a matriz de coeficientes do sistema de equações lineares, com cerca de 1200 vértices, sem as reduções de largura de banda e de *profile*.

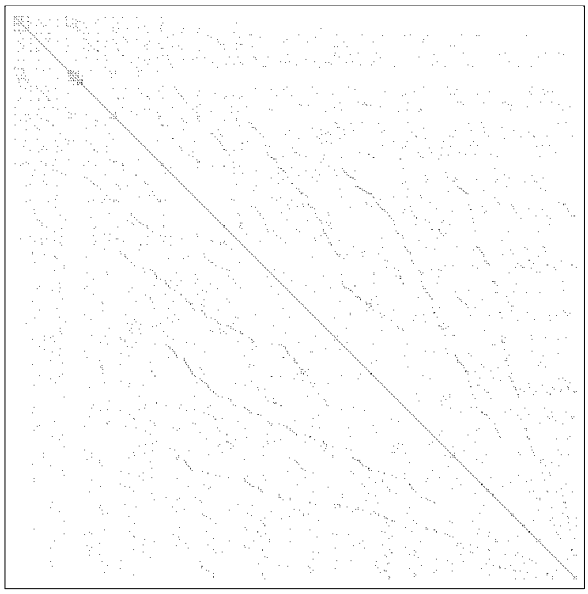

Figura 5.3: Representação de uma matriz de coeficientes com, aproximadamente, 1200 vértices sem a redução de largura de banda.

Na figura 5.4, mostra-se a matriz de coeficientes do sistema de equações lineares, com, aproximadamente, 1200 vértices, com as reduções de largura de banda e de *profile* pela heurística CMr. Na figura 5.5, mostra-se a matriz de coeficientes do sistema de equações lineares, com cerca de 1200 vértices, com as reduções de largura de banda e de *profile* pela heurística CMr, iniciando-se por um vértice pseudo-periférico.

5.4 Considerações sobre as simulações

Implementou-se a heurística Cuthill-McKee reverso (CMr) de George [22] e o algoritmo de George e Liu [21], que encontra um vértice pseudo-periférico. Verificou-se o desempenho do método dos gradientes conjugados com as reduções de largura de banda e de *profile*, após a utilização dessas heurísticas, em matrizes de coeficientes de sistemas de equações lineares.

Com os resultados apresentados na seção 5.3, verificou-se que, ao se utilizar a heurística CMr para se reduzirem a largura de banda e o *profile* da matriz de coeficientes de um sistema de equações lineares, houve redução no tempo de execução do método dos gradientes conjugados nas simulações deste trabalho. Verificou-se que a redução de tempo de execução do método dos gradientes conjugados, após a execução da heurística CMr, não é proporcional às reduções de largura de banda

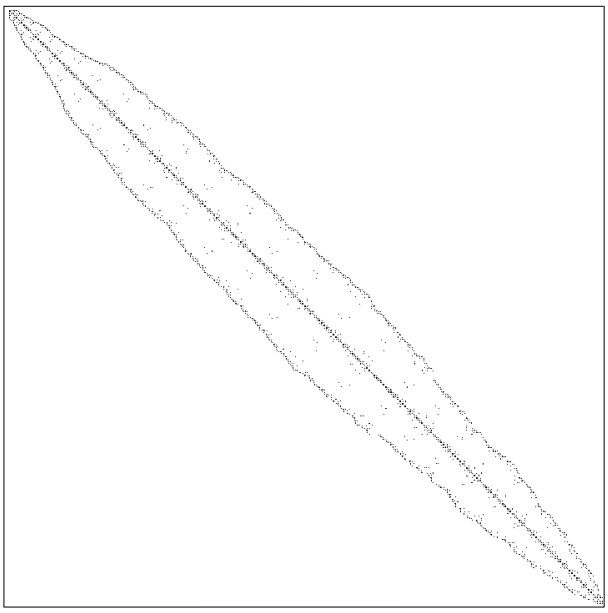

Figura 5.4: Representação de uma matriz de coeficientes com cerca de 1200 vértices com as reduções de largura de banda e de *profile* realizadas pela heurística CMr.

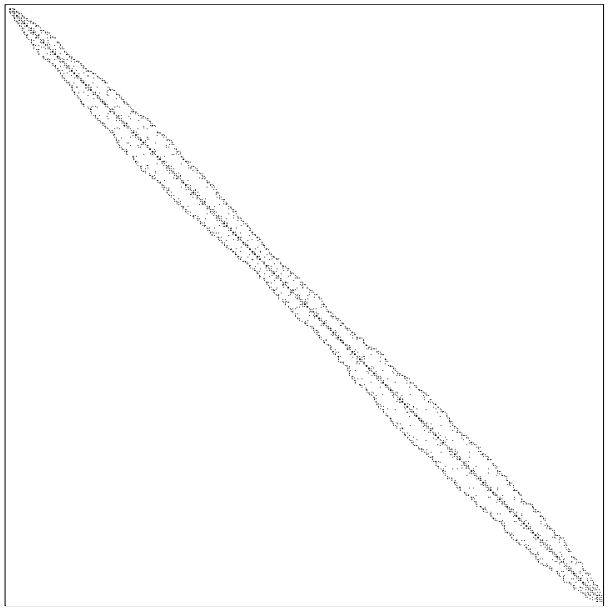

Figura 5.5: Representação de uma matriz de coeficientes com, aproximadamente, 1200 vértices com as reduções de largura de banda e de *profile* realizadas pela heurística CMr iniciando-se por um vértice pseudo-periférico.

e de *profile* da matriz do sistema de equações lineares. Ainda, com as reduções de largura de banda e de *profile* obtidas pela heurística CMr-pseudo, não se obtiveram reduções significativas na redução do custo computacional do método dos gradientes

conjugados em relação aos resultados desse método com as reduções de largura de banda e de *profile* pela heurística CMr. Nos testes com a heurística CMr-pseudo, com cerca de 35,6% de reduções médias em largura de banda e de *profile*, houve redução média no custo computacional do método dos gradientes conjugados de 0,9%, aproximadamente, em relação à utilização da heurística CMr.

Como descrito, os testes foram realizados na resolução da equação de Laplace, com sistemas de equações lineares oriundos de discretizações de tamanho médio por volumes finitos, em que os volumes de controle são polígonos de Voronoi. A resolução dos sistemas de equações lineares foi realizada pelo método dos gradientes conjugados. Esses testes formam um conjunto específico e pequeno de simulações. Apesar disso, podem indicar que, após um certo limite na redução do *profile* ou da largura de banda, pode não haver vantagem significativa na redução do custo computacional na resolução de sistemas de equações lineares pelo método dos gradientes conjugados. Isso pode ser um indício de que a utilização de uma heurística que reduza muito a largura de banda, como as baseadas em meta-heurísticas, não surtam o efeito desejado de redução do custo computacional na resolução do sistemas de equações lineares nesses casos. Com isso, a utilização da heurística CMr para as reduções da largura de banda e de *profile*, iniciada por um vértice "qualquer", pode ser vantajosa em relação à utilização de outras heurísticas; pois a heurística CMr é muito simples de ser implementada. Consequentemente, a manutenção do código é também simples. Em particular, pode-se utilizar, de forma fácil, um vértice da malha computacional (ou vértice do grafo que represente um politopo da malha em discretizações de volumes finitos) que represente uma incógnita no sistema na fronteira do domínio. Esse vértice na fronteira da malha computacional é um vértice pseudo-periférico no grafo e isso foi realizado nas simulações mostradas na subseção 5.3.2.

Claramente, testes precisam ser realizados com simulações em ambiente real. Ainda, testes similares precisariam ser realizados com matrizes de discretizações de elementos finitos em vários modelos tridimensionais. Inclusive, testes precisam ser realizados com a aplicação de métodos iterativos para a resolução de sistemas de equações lineares com matrizes assimétricas, como o GMRES [77], bem como na resolução do sistemas de equações lineares por métodos diretos, isto é, por métodos baseados na eliminação gaussiana.

5.5 Exercícios

1. Observe as reduções de largura de banda e de *profile* mostrados na tabela 5.2 em relação aos dados mostrados na tabela 5.1. Qual é a magnitude dessas reduções?

2. Observe os números de iterações do método dos gradientes conjugados mostrados nas tabelas 5.1 e 5.2.

   (a) Houve redução do número de iterações do método dos gradientes conjugados com a utilização da heurística CMr?

   (b) Houve redução do tempo de processamento com a utilização da heurística CMr?

   (c) Explique as respostas das questões *a* e *b*.

3. Analise os resultados mostrados nas tabelas 5.1 e 5.2 e explique se compensa a utilização da heurística CMr antes da resolução de um sistema de equações lineares pelo método dos gradientes conjugados.

4. No primeiro conjunto de simulações, a diferença do tempo de execução na maior instância, com 277948 vértices, entre as heurísticas CMr e CMr-pseudo é menor que um segundo. Analise os resultados mostrados nas tabelas 5.2 e 5.3 e explique se compensa a implementação do método de George e Liu [21] antes da execução da heurística CMr.

Bibliografia

[1] ARANY, I. Another method finding pseudo-peripheral nodes. In: *Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos nominatae*. Budapesti: Budapesti ELTE, 1983. p. 39–49. Tom. 4.

[2] BARNARD, S.; PHOTEN, A.; SIMON, H. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Aplications*, v. 3, p. 317–334, 1995.

[3] BENZI, M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, v. 182, p. 418–477, 2002.

[4] BOUTORA, Y.; TAKORABET, N.; IBTIOUEN, R.; MEZANI, S. A new method for minimizing the bandwidth and profile of square matrices for triangular finite elements mesh. *IEEE Transactions on Magnetics*, v. 43, n. 4, p. 1513–1516, april 2007.

[5] CAPRARA, A.; SALAZAR-GONZÁLEZ, J.-J. Laying out sparse graphs with provably minimum bandwidth. *INFORMS Journal on Computing*, v. 17, n. 3, p. 356–373, 2005.

[6] CHENG, K. Minimizing the bandwidth of sparse symmetric matrices. *Computing*, Springer Wien, v. 11, p. 107–110, 1973.

[7] CORSO, G. M. D.; MANZINI, G. Finding exact solutions to the bandwidth minimization problem. *Computing*, v. 62, n. 3, p. 189–203, 1999.

[8] CRANE, H. L. J.; GIBBS, N. E.; POOLE, W. G. J.; STOCKMEYER, P. K. Algorithm 508: Matrix bandwidth and profile reduction [f1]. *ACM Transactions on Mathematical Software*, v. 2, n. 4, p. 375–377, 1976.

[9] CUTHILL, E.; MCKEE, J. Reducing the bandwidth of sparse symmetric matrices. In: *ACM Proceedings of the 1969 24th national conference*. New York, USA: ACM, 1969. p. 157–172.

[10] DAVIS, T. A. *Direct methods for sparse linear systems*. Illustrated edition. Gainesville, Florida, USA: Society for Industrial and Applied Mathematic (SIAM), 2006.

[11] DÍAZ, J.; GIBBONS, A. M.; PATERSON, M. S.; TORÁN, J. The minsumcut problem. In: DEHEN, F.; SACK, R. J.; SANTORO, N. (Ed.). *Algorithms and Data Structures*. Ottawa, Canada: Springer-Verlag, 1991, (Lecture Notes in Computer Science, v. 519). p. 65–79.

[12] DORIGO, M.; MANIEZZO, V.; COLORNI, A. *Ant system: an autocatalytic optimizing process*. Milan, 1991. Relatório técnico. Disponível em: <ftp://iridia.ulb.ac.be/pub/mdorigo/tec.reps/TR.02-ANTS-91-016REV.ps.gz>.

[13] DUECK, G. W.; JEFFS, J. A heuristic bandwidth reduction algorithm. *Journal of Combinatorial Mathematics and Combinatorial Computing*, p. 97–108, 1995.

[14] DUFF, I. S.; ERISMAN, A. M.; REID, J. K. On George's nested dissection method. *SIAM Journal on Numerical Analysis*, v. 3, p. 689–695, 1976.

[15] DUFF, I. S.; MEURANT, G. A. The effect of ordering on preconditioned conjugate gradients. *BIT Numerical Mathematics*, v. 29, n. 4, p. 635–657, 1989.

[16] DUTTO, L. C. The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier-Stokes equations. *International Journal for Numerical Methods in Engineering*, v. 36, n. 3, p. 457–497, 1993.

[17] EVERSTINE, G. C. A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront. *International Journal for Numerical Methods in Engineering*, v. 14, p. 837–853, 1979.

[18] FELIPPA, C. A. Solution of linear equations with skyline-stored symmetric matrix. *Computers and Structures*, v. 5, p. 13–29, 1975.

[19] FENG, G. An improvement of the gibbs-poole-stockmeyer algorithm. *Journal of Algorithms and Computational Technology*, v. 4, n. 3, p. 325–333, 2009.

[20] GAREY, M. R.; GRAHAM, R. L.; JOHNSON, D. S.; KNUTH, D. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, v. 34, p. 477–495, 1978.

[21] GEORGE, A.; LIU, J. W. H. An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software*, n. 5, p. 284–295, 1979.

[22] GEORGE, J. A. *Computer implementation of the finite element method*. 228 p. Tese (Doutorado) — Computer Science Department, Stanford University, CA, USA, 1971.

[23] GEORGE, T.; GUPTA, A.; SARIN, V. An empirical analysis of the performance of preconditioners for SPD systems. *ACM Transactions on Mathematical Software*, v. 38, n. 4, 2012.

[24] GIBBS, N. E. Algorithm 509: A hybrid profile reduction algorithm. *ACM Transactions on Mathematical Software*, ACM, New York, NY, USA, v. 2, n. 4, p. 378–387, dez. 1976.

[25] GIBBS, N. E. *A survey of bandwidth and profile reduction algorithms*. Williamsburg, USA, 1980. Relatório técnico.

[26] GIBBS, N. E.; POOLE, W. G.; STOCKMEYER, P. K. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, v. 2, n. 13, p. 236–250, 1976.

[27] GOLOVACH, P. A. The total vertex separation number of a graph. *Diskretnaya Matematika*, v. 9, n. 4, p. 86–91, 1997. In Russian.

[28] GONZAGA DE OLIVEIRA, S.; KISCHINHEVSKY, M. Sierpiński curve for total ordering of a graph-based adaptive simplicial-mesh refinement for Finite Volume discretizations. In: *Congresso Nacional de Matemática Aplicada e Computacional (CNMAC)*. Belém: Anais do Congresso Nacional de Matemática Aplicada e Computacional, 2008. p. 581–585.

[29] GONZAGA DE OLIVEIRA, S. L. *Algoritmos e seus fundamentos*. Lavras: Editora UFLA, 2011. 215–226 p.

[30] GONZAGA DE OLIVEIRA, S. L.; KISCHINHEVSKY, M.; TAVARES, J. M. R. S. Novel graph-based adaptive triangular mesh refinement for finite-volume discretizations. *CMES: Computer Modeling in Engineering & Sciences*, v. 95, n. 2, p. 119–141, 2013.

[31] GRIMES, R. G.; PIERCE, D. J.; SIMON, H. D. A new algorithm for finding a pseudoperipheral node in a graph. *SIAM Journal of Analysis and Applications*, v. 11, p. 323–334, 1990.

[32] GURARI, E. M.; SUDBOROUGH, I. H. Improved dynamic programming algorithms for bandwidth minimization and the min-cut linear arrangement problem. *Journal of Algorithms*, v. 5, p. 531–546, 1984.

[33] HAMMING, R. W. Error detecting and error correcting codes. *Bell System Technical Journal*, v. 29, n. 2, p. 147–160, 1950.

[34] HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. 1st edition. ed. New York: Springer, 2003. p. 145–184.

[35] HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search methods. In: FLOUDAS, C. A.; PARDALOS, P. M. (Ed.). *Encyclopedia of Optimization*. 2nd edition. ed. New York: Springer, 2009. p. 3975–3989.

[36] HARARY, F. *Graph Theory*. Reading, MA: Addison-Wesley, 1969.

[37] HESTENES, M. R.; STIEFEL, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, v. 49, n. 36, p. 409–436, 1952.

[38] IRONS, B. M. A frontal solution scheme for finite element analysis. *International Journal of Numerical Methods in Engeneering*, v. 2, p. 5–32, 1970.

[39] JENNINGS, A. A compact storage scheme for the solution of symmetric linear simultaneous equations. *The Computer Journal*, v. 9, p. 281–285, 1966.

[40] KAVEH, A. Ordering for bandwidth reduction. *Computer Structure*, v. 24, p. 413–420, 1986.

[41] KAVEH, A. Algebraic graph theory for ordering. *Computers and Structures*, v. 37, n. 1, p. 51–54, 1990.

[42] KAVEH, A. *Structural Mechanics: Graph and Matrix Methods*. Baldock, England: Research Studies Press LTD, 2004.

[43] KAVEH, A.; SHARAFI, P. Nodal ordering for bandwidth reduction using ant system algorithm. *Engineering Computations*, v. 26, p. 313–323, 2009.

[44] KAVEH, A.; SHARAFI, P. Ordering for bandwidth and profile minimization problems via charged system search algorithm. *IJST Transactions of Civil Engineering*, v. 36, p. 39–52, 2012.

[45] KAVEH, A.; TALATAHARI, S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers and Structures*, v. 87, p. 267 – 283, 2009. ISSN 0045-7949. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045794909000091>.

[46] KAVEH, A.; TALATAHARI, S. A novel heuristic optimization method: charged system search. *Acta Mechanica*, p. 267–286, January 2010.

[47] KAZOELA, M.; NIKOLOS, A. L.; SYNOLAKIS, C. E. An unstructured finite volume numerical scheme for extended 2D Boussinesq-type equations. *Coastal Engineering*, v. 69, p. 42–66, 2012.

[48] KRYLOV, A. N. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined (in Russian). *Izvestija AN SSSR (News of Academy of Sciences of the USSR), Otdel. mat. i estest. nauk, 1931, VII*, v. 491-539, n. 4, 1931.

[49] LANCZOS, C. Solutions of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, v. 49, n. 3, p. 33–53, 1952.

[50] LEWIS, J. G. Implementations of the Gibbs-Poole-Stockmeyer algorithms and Gibbs-King algorithms. *ACM Transactions on Mathematical Software*, v. 8, p. 180–189, 1982.

[51] LIM, A.; LIN, J.; RODRIGUES, B.; XIAO, F. Ant colony optimization with hill climbing for the bandwidth minimization problem. *Applied Soft Computing*, v. 6, n. 2, p. 180–188, 2006.

[52] LIM, A.; LIN, J.; XIAO, F. Particle Swarm optimization and hill climbing for the bandwidth minimization problem. *Applied Intelligence*, n. 26, p. 175–182, 2007.

[53] LIM, A.; RODRIGUES, B.; XIAO, F. Integrated genetic algorithm with hill climbing for bandwidth minimization problem. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE (GECCO). Chicago, Illinois, USA, 2003. p. 1594–1595.

[54] LIM, A.; RODRIGUES, B.; XIAO, F. Heuristics for matrix bandwidth reduction. *European Journal of Operational Research*, p. 69–91, 2006.

[55] LIM, A.; RODRIGUES, B.; XIAO, F. A fast algorithm for bandwidth minimization. *International Journal on Artificial Intelligence Tools*, v. 3, p. 537–544, 2007.

[56] LIN, Y. X.; YUAN, J. J. *Minimum profile of grid networks in structure analysis.* Zhengzhou, Henan 450052, People's Republic of China, 1993. Relatório técnico.

[57] LIN, Y. X.; YUAN, J. J. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica*, v. 10, n. 1, p. 107–122, 1994.

[58] LIU, J. W. *On reducing the profile of sparse symmetric matrices.* 208 p. Tese (Doutorado) — University of Waterloo, fev. 1976.

[59] LIU, W.; SHERMAN, A. H. *Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices*. New Haven, USA, 1975. Technical report #28.

[60] LIU, W.; SHERMAN, A. H. Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, v. 13, n. 2, p. 198–213, april 1976.

[61] LIVESLEY, R. K. The analysis of large structural systems. *The Computer Journal*, v. 3, n. 1, p. 34–39, 1960.

[62] MAFTEIU-SCAI, L. O. Bandwidth reduction in sparse matrices. In: *Analele Universitãţii de Vest, Seria Matematicã - Informaticã*. Timişoara, Romania: Versita, 2010. XLVIII, n. 2, p. 163–173.

[63] MARTÍ, R.; CAMPOS, V.; PIÑANA, E. A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, v. 186, p. 513–528, 2008.

[64] MARTÍ, R.; LAGUNA, M.; GLOVER, F.; CAMPOS, V. Reducing the bandwidth of a sparse matrix with tabu search. *European Journal of Operational Research*, v. 135, n. 2, p. 450–459, 2001.

[65] MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers and Operations Research*, v. 24, n. 11, p. 1097–1100, 1997.

[66] MLADENOVIC, N.; UROSEVIC, D.; PÉREZ-BRITO, D.; GARCÍA-GONZÁLEZ, C. G. Variable neighbourhood search for bandwidth reduction. *European Journal of Operational Research*, n. 200, p. 14–27, 2010.

[67] NOGUEIRA, J. R.; GONZAGA DE OLIVEIRA, S. L. Introduction to triangular mesh generation and the delaunay triangulation. In: *Proceedings of the XXXII Iberian Latin-American Congress on Computational Methods in Engineering - XXXII CILAMCE*. Ouro Preto: ABMEC, 2011.

[68] OLIVEIRA, T. H.; GONZAGA DE OLIVEIRA, S. L. Uma resolução da equação de Laplace por Volumes Finitos e diagrama de Voronoi com refinamento adaptativo e de Ruppert. In: *Proceedings of the SIBGRAPI 2012 - International Conference on Graphics, Patterns and Images - Workshop of Undergraduate Works (WUW)*. Ouro Preto: SBC, 2012.

[69] PACHL, J. K. Finding pseudoperipheral nodes in graphs. *Journal of Computer and System Sciences*, v. 29, n. 1, p. 48–53, 1984. Disponível em: <http://dx.doi.org/10.1016/0022-0000(84)90012-6>.

[70] PAPADIMITRIOU, C. The NP-completeness of bandwidth minimization problem. *Computing Journal*, v. 16, p. 177–192, 1976.

[71] PETIT, J. Addenda to the survey of layout problems. *Bulletin of the EATCS*, European Association for Theoretical Computer Science, v. 105, p. 177–201, 2011.

[72] PIÑANA, E.; PLANA, I.; CAMPOS, V.; MARTÍ, R. GRASP and path relinking for the matrix bandwidth minimization. *European Journal of Operational Research*, v. 153, n. 1, p. 200–210, 2004.

[73] PISSANETZKY, S. *Sparse Matrix Technology*. Eletronic edition. Michigan, USA: Academic Press, 1984.

[74] RODRIGUEZ-TELLO, E.; KAO, H. J.; TORRES-JIMENEZ, J. An improved simulated annealing algorithm for bandwidth minimization. *European Journal of Operational Research*, n. 185, p. 1319–1335, 2008.

[75] RODRIGUEZ-TELLO, E.; TORRES-JIMENEZ, J. An improved evaluation function for the bandwidth minimization problem. In: *Parallel Problem Solving from Nature - PPSN VIII*. Birmingham, UK: Springer Berlin Heidelberg, 2004. v. 3242, p. 652–661.

[76] RUPPERT, J. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, v. 18, n. 3, p. 548–585, 1995.

[77] SAAD, Y. *Iterative Methods for Sparse Linear Systems*. Segunda edição. Philadelphia: Society for Industrial and Applied Mathematics, 2003.

[78] SKIENA, S. S. *The Algorithm Design Manual*. Second edition. London, UK: Springer, 2008.

[79] SMYTH, W. F. Algorithms for the reduction of matrix bandwidth and profile. *Journal of Computational and Applied Mathematics*, v. 12–13, p. 551–561, 1985.

[80] SNAY, R. A. Reducing the profile of sparse symmetric matrices. *Bulletin Géodésique*, v. 50, p. 341–352, 1976.

[81] SOUZA, L. T.; MURRAY, D. W. An alternative pseudoperipheral node finder for resequencing schemes. *International Journal for Numerical Methods in Engineering*, v. 36, n. 19, p. 3351–3379, 1993.

[82] TARJAN, R. E. *Graph theory and gaussian elimination*. Computer Science Department, Stanford University, november 1975. Relatório técnico.

[83] The MathWorks Inc. *MATLAB*. 1994–2013. Acesso em: 03/11/2013. Disponível em: <http://www.mathworks.com/products/matlab/index.html>.

[84] TORRES-JIMENEZ, J.; RODRIGUEZ-TELLO, E. A new measure for the bandwidth minimization problem. In: *International Joint Conference 7th Ibero-American Conference on AI 15th Brazilian Symposium on AI IBERAMIA-SBIA 2000*. Mexico: Springer Berlin Heidelberg, 2000. p. 477–486.

[85] ÜNGÖR, A. Off-Centers: a new type of steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. In: *6th Latin American Symposium*. Buenos Aires, Argentina: Springer, 2004. v. 2976, p. 152–161.

[86] ÜNGÖR, A. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Computational Geometry*, v. 42, n. 2, p. 109–118, 2009.

[87] WANG, Q.; GUO, Y. C.; SHI, X. W. An improved matrix bandwidth and profile reduction algorithm in FEM problems. In: *Progress In Electromagnetics Research Symposium*. Hangzhou, China: The Electromagnetics Academy, 2008. p. 853–857.

[88] WANG, Q.; GUO, Y. C.; SHI, X. W. A generalized GPS algorithm for reducing the bandwidth and profile of a sparse matrix. In: *Progress In Electromagnetics Research*. Cambridge, Massachusetts, USA: EMW Publishing, 2009. v. 90, p. 121–136.

[89] WANG, Q.; GUO, Y. C.; SHI, X. W. An improved algorithm for matrix bandwidth and profile reduction in finite element analysis. *Progress In Electromagnetics Research Letters*, v. 9, p. 29–38, 2009.

[90] WANG, Q.; SHI, X.; GUO, C.; GUO, Y. An improved GPS method with a new pseudo-peripheral nodes finder in finite element analysis. *Finite Elements in Analysis and Design*, v. 48, n. 1, p. 1409–1415, 2012.

Índice