Trabalho apresentado no XXXVIII CNMAC, Campinas - SP, 2018.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Um análise comparativa entre quatro algoritmos para reduções de largura de banda de matrizes

Sanderson L. Gonzaga de Oliveira e Guilherme O. Chagas[1]
Departamento de Ciência da Computação, UFLA, Lavras, MG

Resumo. Neste trabalho, é relatado um experimento computacional com o objetivo de comparar quatro algoritmos heurísticos que podem ser considerados, em diferentes situações, os principais métodos em redução de largura de banda de matrizes. Foi publicado, recentemente, o algoritmo *Dual Representation Simulated Annealing* para redução de largura de banda de matrizes. Os resultados desse algoritmo superaram os resultados de outro algoritmo, baseado na metaheurística Variable Neighborhood Search, considerada pelos autores como o método no estado da arte anterior no problema. Neste contexto, as heurísticas devem apresentar custo de execução muito baixo, pois são utilizadas para acelerar resolutores de sistemas de equações lineares. Entretanto, em geral, algoritmos baseados em meta-heurísticas são lentos e não são eficazes nesse problema de otimização combinatória, mesmo quando aplicados em instâncias pequenas. Neste trabalho, comparamos resultados do algoritmo *Dual Representation Simulated Annealing* com resultados do método *Reverse Cuthill–McKee*, uma variação desse método e uma variação da busca em largura, que foram consideradas, recentemente, as três heurísticas de baixo custo de execução mais promissoras no problema de redução de largura de banda, para diferentes áreas de aplicação. Baseados nos resultados de nossos experimentos, consideramos que essas três heurísticas de baixo custo de processamento superam o algoritmo *Dual Representation Simulated Annealing*, quando o tempo de execução é uma variável considerada na comparação.

Palavras-chave. Matrizes esparsas, algoritmos em grafos, redução de largura de banda, teoria dos grafos, renumeração de vértices.

1 Introdução

Um passo fundamental em diversas aplicações científicas e em engenharia é a solução de sistemas de equações lineares de grande porte, $Ax = b$, em que A é uma matriz $n \times n$, x é o vetor de incógnitas e b é o vetor de termos independentes. Geralmente, esse é o passo que demanda o maior tempo de execução da simulação feita em diversas áreas da ciência e engenharia. Políticas de paginação e a arquitetura hierárquica de memória dos projetos de computadores modernos favorecem programas que consideram localidade de referência em memória. Uma numeração de vértices adequada é desejada para se obter soluções de sistemas de equações lineares com baixo custo de execução. Com uma ordenação dos vértices apropriada, a matriz de coeficientes A associada terá largura de banda pequena,

[1]sanderson@dcc.ufla.br,guilherme.chagas@computacao.ufla.br

justificando, assim, a utilização de heurísticas para redução de largura de banda como alternativa para se obter uma numeração de vértices que forneça coerência de *cache*. As heurísticas para redução de largura de banda são empregadas como um passo de pré-processamento de matrizes na resolução de sistemas de equações lineares [13, 11, 12].

Seja $A = [a_{ij}]$ uma matriz simétrica, $n \times n$, associada a um grafo não-direcionado $G = (V, E)$, composto de um conjunto V de vértices e um conjunto E de arestas. A largura de banda da linha i da matriz A é $\beta_i(A) = i - \min_{1 \leq j \leq i} [j : a_{ij} \neq 0]$. Com isso, a largura de banda da matriz A é definida como $\beta(A) = \max[\beta_i(A)]$. De forma equivalente, a largura de banda do grafo G, para uma numeração $S = \{s(v_1), s(v_2), \cdots, s(v_{|V|})\}$, isto é, uma bijeção de V para o conjunto $\{1, 2, \cdots, |V|\}$, é $\beta(G) = \max_{v \in V}[(\{v, u\} \in E)\ |s(v) - s(u)|]$. O problema de minimização de largura de banda pertence à classe NP-Difícil [16].

Os autores do algoritmo *Dual Representation Simulated Annealing* (DRSA) [20] afirmam que esse método é o atual estado da arte em redução de largura de banda de matrizes. Foram realizados testes em um conjunto de instâncias com, no máximo, 1000 vértices [20], e foram mostrados resultados que esse algoritmo supera a heurística baseada na meta-heurística *Variable Neighborhood Search*, considerada pelos autores como o método no estado da arte anterior no problema, mas com custo de execução, pelo menos, cinco vezes mais lenta. Apesar de reduzirem substancialmente a largura de banda da instância, esses algoritmos apresentam custo alto de execução. Nesse problema, todavia, as heurísticas devem apresentar custo de execução muito baixo, pois são utilizadas para acelerar resolutores de sistemas de equações lineares. Em geral, algoritmos baseados em meta-heurísticas são lentos, mesmo quando aplicados em instâncias pequenas. Ainda, os tempos mostrados são aqueles em que o algoritmo encontra a melhor solução [20]. Entretanto, o usuário deve esperar a heurística terminar (sua execução), isto é, o melhor resultado pode ser encontrado pelo programa muito antes do término (da execução) da heurística.

Neste presente trabalho, comparamos resultados do método DRSA com o método *Reverse Cuthill–McKee* (RCM) [6] com vértices iniciais fornecidos pelo algoritmo de George-Liu [7, 8]. Esse método é um dos mais conhecidos para redução de largura de banda de matrizes [1, 3, 10, 11, 12, 18]. O método RCM-GL [8] fornece reduções de largura de banda razoáveis, a custo de execução extremamente baixo.

Nossos experimentos também avaliam as heurísticas de Koohestani e Poli (KP-band) [15] e a busca em largura com ordenação inversa (RBFS) [12] para numeração de vértices. Em publicações anteriores [11, 10, 12], foi verificado que esses três algoritmos fornecem resultados melhores que algoritmos baseados em meta-heurísticas, se o custo computacional (tempo de execução e ocupação de memória) é levado em consideração.

O restante deste texto está estruturado da seguinte forma. Os algoritmos avaliados neste experimento computacional são brevemente explicados na Seção 2. Os resultados e análises são apresentados na Seção 3. As conclusões são apresentadas na Seção 4.

2 Algoritmos avaliados

No algoritmo 1, é mostrado um pseudo-código do método *Reverse Cuthill–McKee* [6]. Esse método é uma variação gulosa da busca em largura, em que cada lista de adjacências de

DOI: 10.5540/03.2018.006.02.0422

vértices é numerada em ordem de grau (veja a linha 6). Ao final, a numeração é invertida (veja a linha 12 nesse pseudo-código).

Algoritmo 1: Método *Reverse Cuthill–McKee* [6].

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

A heurística de Koohestani e Poli [15] é baseada na heurística *Reverse Cuthill–McKee* [6]. A diferença entre elas está na estratégia utilizada para renumerar os vértices de cada lista de adjacências. Como descrito, no método *Reverse Cuthill–McKee* [6], os vértices em cada lista de adjacências são renumerados por ordem de grau (veja a linha 6 no Algoritmo 1). No método KP-band [15], os vértices em cada lista de adjacências são renumerados na ordem (crescente) dada pela fórmula $0,179492928171 \cdot \varsigma(w)^3 + 0,292849834929 \cdot \varsigma(w)^2 - 0,208926175433 \cdot |V| - 0,736485142138 \cdot |V| \cdot \varsigma(w) - 1,77524579882 \cdot \varsigma(w) - 1,75681383404$, em que $\varsigma(w)$ é o somatório dos graus dos vértices adjacentes ao vértice w. Note que ordenar os vértices por essa fórmula substitui "em ordem crescente de grau" na linha 6 do Algoritmo 1.

O algoritmo RBFS [12] é simplesmente uma numeração de vértices na ordem dada pela busca em largura e, ao final, a ordem é invertida. Para os três métodos, *Reverse Cuthill–McKee* [6], KP-band [15] e RBFS, o vértice inicial do processamento é dado pelo algoritmo de George-Liu (GL) [7]. No algoritmo 2, é mostrado o algoritmo de George e Liu [7] para encontrar vértices pseudo-periféricos [14]. Dado um vértice $v \in V$, a estrutura de nível $\mathscr{L}(v)$ enraizada no vértice v, com 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 $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, para $i = 1, 2, 3, \ \ldots, \ell(v)$ e $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u, w\} \in E\}$. Em particular, $\ell(v) = \max_{\forall u \in V}[d(v, u)]$ representa a excentricidade do vértice v e a distância $d(v, u)$ é o tamanho do menor caminho que conecta os vértices v e u. O algoritmo de George-Liu retorna (na linha 11) um vértice v que tenha excentricidade maior ou igual (veja a linha 7) ao vértice de menor grau em $L_{\ell(v)}(v)$ (veja a linha 5 nesse pseudocódigo).

No algoritmo *Dual Representation Simulated Annealing*, há uma representação dual interna do problema. Essa representação é utilizada em conjunto com uma função de vizinhança composta de três operadores de perturbação. A função de avaliação do algo-

DOI: 10.5540/03.2018.006.02.0422

Algoritmo 2: Algoritmo de George e Liu [7] para encontrar vértices pseudo-periféricos.

Entrada: grafo $G = (V, E)$;
Saída: vértice pseudo-periférico $v \in V$;
início
  $v \leftarrow EscolheVertice(V)$;
  `// constrói-se estrutura de nível enraizada`
  $\mathscr{L}(v) \leftarrow BuscaEmLargura(v)$;
  repita
    $u \leftarrow VerticeComGrauMinino(L_{\ell(v)}(v))$;
    `// constrói-se estrutura de nível enraizada`
    $\mathscr{L}(u) \leftarrow BuscaEmLargura(u)$;
    se ($\ell(u) > \ell(v)$) **então**
      $v \leftarrow u$; $\mathscr{L}(v) \leftarrow \mathscr{L}(u)$;
    fim-se;
  até que ($u \neq v$) ;
  retorna v;
fim.

ritmo DRSA diferencia soluções de redução de largura de banda ao considerar todas as diferenças absolutas entre rótulos de vértices adjacentes. Calibramos nosso código nas instâncias mostradas na publicação do algoritmo DRSA. O algoritmo encontra as larguras de bandas mostradas na publicação original da heurística [20], mas converge muito lentamente.

3 Resultados e análises

Nesta seção, são apresentados e analisados resultados de simulações com os algoritmos DRSA, RCM-GL [8], KP-band [15] e RBFS-GL [12] para redução de largura de banda. São utilizadas instâncias pequenas, pela dificuldade de executar o método DRSA em instâncias maiores. Os métodos RCM-GL [8], KP-band [15] e RBFS-GL [12] foram implementados na linguagem de programação C++. O método DRSA foi implementado na linguagem de programação C, que pode gerar códigos mais rápidos, em comparação com a linguagem de programação C++. A estrutura de dados utilizada na implementação dos algoritmos de reordenação foi a Compress Row Storage [17].

A estação de trabalho utilizada nos experimentos contém processador Intel® Core™ i7-4770 (CPU 3.40GHz, 8MB de memória *cache*, 8GB de memória principal DDR3 1.333GHz) (Intel; Santa Clara, CA, United States). O sistema operacional instalado nessa máquina é uma distribuição Ubuntu 16.04 64 bits, com versão *kernel* do Linux 4.2.0-36-generic.

Na Tabela 1, são mostrados resultados de experimentos com cinco matrizes obtidas da base SuiteSparse [4]. Nessa tabela, são mostrados os nomes das matrizes, dimensões,

DOI: 10.5540/03.2018.006.02.0422

largura de banda original (β_0), e as larguras de banda e tempos de execução, em segundos, resultantes das execuções dos algoritmos KP-band [15], RCM-GL [8], RBFS-GL [12] e DRSA [20]. Como os tempos de execução dos algoritmos KP-band [15], RCM-GL [8] e RBFS-GL [12] são bastante baixos, o tempo de execução do algoritmo DRSA foi limitado em 60 segundos, que é 30 mil vezes mais demorado que a execução mais lenta do método RBFS-GL [12] nesse conjunto de matrizes. Na Tabela 1 e na Figura 1, mostra-se que as reduções de largura de banda do algoritmo DRSA são bastante inferiores às reduções de largura de banda alcançadas pelos algoritmos KP-band [15], RCM-GL [8] e RBFS-GL [12].

Tabela 1: Resultados de larguras de banda e tempos de execução, em segundos, de quatro algoritmos para redução de largura de banda [KP-band (KP) [15], RCM-GL [8], RBFS-GL (RB) [12], DRSA (SA)] aplicados em cinco matrizes obtidas da base SuiteSparse [4].

Matriz	$\|V\|$	$\|E\|$	β_0	KP(β)	KP t(s)	RCM(β)	RCM t(s)	RB(β)	RB t(s)	SA(β)	SA t(s)
c-28	4598	30590	4597	1850	0,004	1852	0,003	1880	0,002	4099	60
c-27	4563	30927	4525	2117	0,004	2081	0,004	2150	0,001	4008	60
c-26	4307	34537	4204	2018	0,003	2078	0,006	1984	0,001	3851	60
t2dal_a	4257	37465	86	86	0,004	85	0,068	86	0,001	3997	60
c-29	5033	43731	5024	2451	0,003	2463	0,004	2478	0,002	4643	60

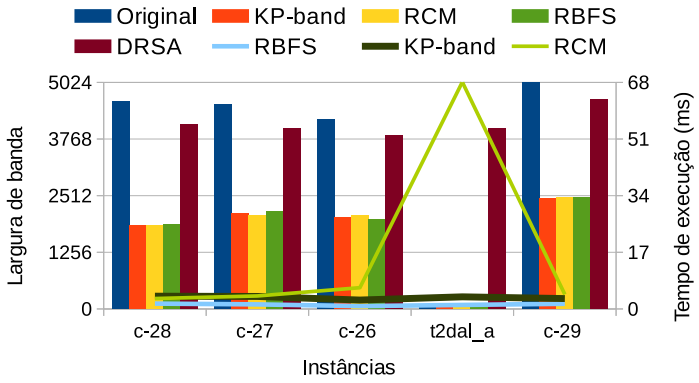

Figura 1: Largura de banda e tempos de execução resultantes de algoritmos para redução de largura de banda de matrizes esparsas aplicados em cinco matrizes obtidas da base SuiteSparse [4]. "Original" refere-se à largura de banda original da matriz. Os tempos de execução do algoritmo DRSA não são mostrados no gráfico porque foi estabelecido 60 segundos como o tempo limite de tempo de execução desse algoritmo.

4 Conclusões

Neste trabalho, foram comparados resultados dos algoritmos de reordenação de vértices RCM-GL [8], KP-band [15], *Dual Representation Simulated Annealing* [20] e RBFS-GL [12]

para redução de largura de banda de matrizes. Baseado nos experimentos, foi mostrado que o algoritmo DRSA requer muito mais tempo de execução que os algoritmos RCM-GL [8], KP-band [15], e RBFS-GL [12]. Ao limitar o tempo de execução do algoritmo DRSA, seus resultados são piores que os resultados em redução de largura de banda obtidos pelos outros três algoritmos. Com isso, consideramos que o algoritmo DRSA pode ser utilizado em estudos teóricos com instâncias bastante pequenas, como no contexto de se encontrar uma largura de banda próxima ao ótimo local sem considerar um limite de tempo de execução. Resultados de algoritmos práticos podem ser comparados com as larguras de banda retornadas por tal algoritmo meta-heurístico. Entretanto, há um sério limitador em sua utilidade prática, por ser extremamente lento. Em particular, no contexto de resolução de sistemas de equações lineares, é melhor que o algoritmo de reordenação de vértices execute muito mais rapidamente que o resolutor, ou seria necessário um número de execuções do resolutor (acelerado) de forma a amortizar o tempo da heurística de redução de largura de banda utilizada na fase de pré-processamento e, este, não é o caso do algoritmo DRSA. Portanto, consideramos que os algoritmos RBFS-GL [12], KP-band [15] e RCM-GL [8] são os métodos mais promissores para redução de largura de banda, de forma que possam ser utilizados como pré-processamento de matrizes esparsas para aceleração de resolutores diretos e iterativos de sistemas de equações lineares. Como forma de continuar este estudo, pretendemos utilizar esses três algoritmos para acelerar resolutores diretos e iterativos de sistemas de equações lineares.

Referências

[1] M. Benzi, D. B. Szyld, and A. Van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM Journal on Scientific Computing*, 20(5):1652–1670, 1999.

[2] Boost. Boost C++ libraries. [online] Disponível na Internet via WWW. URL: http://www.boost.org/, 2018. Acesso em: 2017-06-28.

[3] J. J. Camata, A. L. Rossa, A. M. P. Valli, L. Catabriga, and G. F. Carey and A. L. G. A. Coutinho. Reordering and incomplete preconditioning in serial and parallel adaptive mesh refinement and coarsening flow solutions. *International Journal for Numerical Methods in Fluids*, 69:802–823, 2012.

[4] T. A. Davis and Y. Hu. SuiteSparse Matrix Collection. *ACM Transactions on Mathematical Software*, 38(1):1–25, 2011.

[5] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations, 2015.

[6] A. George. Computer implementation of the finite element method. PhD thesis, Stanford University, Stanford, USA, 1971.

[7] A. George and J. W. H. Liu. An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software*, 5(3):284–295, September 1979.

[8] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall Professional Technical Reference, New York, USA, 1981.

[9] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. SIAM J. Matrix Anal. Appl., 3(1):333–356, 1992.

[10] S. L. Gonzaga de Oliveira, A. A. A. M. Abreu, D. T. Robaina, and M. Kischnhevsky. An evaluation of four reordering algorithms to reduce the computational cost of the Jacobi-preconditioned conjugate gradient method using high-precision arithmetic. *International Journal of Business Intelligence and Data Mining*, 12(2):190–209, 2017.

[11] S. L. Gonzaga de Oliveira, J. A. B. Bernardes, and G. O. Chagas. An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. *Computational & Applied Mathematics*, 2016, DOI: 10.1007/s40314-016-0394-9.

[12] S. L. Gonzaga de Oliveira, J. A. B. Bernardes, and G. O. Chagas. An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. *Computational & Applied Mathematics*. DOI: 10.1007/s40314-017-0490-5, 2017.

[13] S. L. Gonzaga de Oliveira, J. A. B. Bernardes, and G. O. Chagas. An evaluation of several heuristics for bandwidth and profile reductions to reduce the computational cost of the preconditioned conjugate gradient method. In *Proceedings of the Brazilian Symposium on Operations Research (SBPO 2016)*, September 2016.

[14] S. L. Gonzaga de Oliveira e G. O. CHAGAS. *Introdução a heurísticas para redução de largura de banda de matrizes*. Sociedade Brasileira de Matemática Aplicada e Computacional, São Carlos-SP, 2014. Notas em Matemática Aplicada.

[15] B. Koohestani and R. Poli. A hyper-heuristic approach to evolving algorithms for bandwidth reduction based on genetic programming. In Research and Development in Intelligent Systems XXVIII, 93–106. Springer, London, 2011.

[16] C. H. Papadimitriou. The NP-completeness of bandwidth minimization problem. *Computing Journal*, 16:177–192, 1976.

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[18] Science and Technology Facilities Council. HSL. A collection of Fortran codes for large scale scientific computation. [online] Disponível na Internet via WWW. URL: http://www.hsl.rl.ac.uk/. Acesso em: 2017-06-28.

[19] The MathWorks. Inc. MATLAB. [online] Disponível na Internet via WWW. URL: http://www.mathworks.com/products/matlab/, 1994–2018. Acesso em: 2017-06-28.

[20] J. Torres-Jimenez, I. Izquierdo-Marquez, A. Garcia-Robledo, A. Gonzalez-Gomez, J. Bernal, and R. N. Kacker. A dual representation simulated annealing algorithm for the bandwidth minimization problem on graphs. *Inf. Sci.*, 303:33–49, 2015.

