UM ESTUDO COMPARATIVO DE MÉTODOS HEURÍSTICOS PARA REORDENAMENTO DE MATRIZES ESPARSAS

Kamila Ghidetti,
Maria Claudia Boeres,
Lucia Catabriga
Universidade Federal do Espírito Santo - UFES
Departamento de Informática
Av. Fernando Ferrari, 514 - Goiabeiras - 29075-910 - Vitória - ES - Brasil
rg.kamila@gmail.com, boeres@inf.ufes.br, luciac@inf.ufes.br

Resumo

Dentre os algoritmos heurísticos para reordenamento de matrizes existentes na literatura destacam-se: Reverse Cuthill-McKee (RCM), Gibbs-Poole-Stockmeyer (GPS), Nested Dissection (ND) e Espectral (ES). Tais algoritmos são comparados quanto à qualidade de solução (minimização de largura de banda e minimização de envelope) e tempo de execução. Este trabalho tem como objetivo realizar um estudo comparativo deste conjunto de algoritmos, bem como propor algumas modificações para melhorar a qualidade de solução reduzindo o tempo de processamento. Para os testes computacionais, é considerado um conjunto de matrizes estruturalmente simétricas oriundas das mais diversas áreas do conhecimento. Nossos estudos concluem que as modificações implementadas melhoram ou mantem a qualidade da solução e em alguns casos são capazes de reduzir o tempo de processamnto.

PALAVRAS CHAVES. Minimização de Largura de Banda e Redução de Envelope. Ordenação de matrizes. Algoritmos em Grafos. Otimização Combinatória.

Abstract

Among the matrix ordering heuristic algorithms found in the literature, we can cite: Reverse Cuthill-Mackee (RCM), Gibbs-Poole-Stockmeyer (GPS), Nested Dissection (ND) and Espectral Algorithm (ES). These algorithms are compared with respect to quality of solution (minimizing bandwith and envelope reduction) and CPU time. In this work, we present a comparative study of this set of algorithms and propose modifications to some of them in order to reduce CPU time. For the computational experiments, we consider a set of structural symmetric sparse matrices from several knowledge areas. We conclude that the modifications implemented improve or keep the solution quality and, for some matrices, the CPU time is reduced.

KEYWORDS. Matrix Bandwith Minimization and Envelope Reduction. Matrices Ordering. Graph Algorithms. Combinatorial Optimization.

1 Introdução

Uma parcela significativa dos problemas científicos envolvem a resolução de sistemas lineares, sendo que na maioria das aplicações as matrizes são esparsas e de grande porte. Nestes casos, exemplos de pré-processamento que podem ser realizados, de forma a reduzir a complexidade da resolução de sistemas, são a minimização da largura de banda (Carvalho et al. (2009)) e a redução do envelope (Barnad et al. (1995)). Esses pré-processamentos consistem em dispor os elementos não nulos da matriz o mais próximo possível da diagonal principal. Para isso, permutações entre linhas e colunas devem ser realizadas. Essa minimização permite que as operações realizadas na resolução do sistema sejam feitas sobre uma matriz mais compacta. No contexto da solução de sistemas via métodos diretos, a minimização da largura de banda pode proporcionar uma redução no preenchimento que ocorre na decomposição LU, ficando esse preenchimento restrito aos zeros contidos entre os elementos não nulos e a diagonal principal. Contudo, atualmente sistemas lineares de grande porte são usualmente solucionados por métodos iterativos não estacionários (Saad (2003)) que por um lado não alteram a esparsidade da matriz, mas necessitam de critérios de convergência. Em geral, um processo para acelerar a convergência, denominado precondicionamento, se faz necessário. Os precondicionadores baseados na decomposição incompleta LU (Benzi et al. (1999); Camata et al. (2010)) são amplamente utilizados por apresentar uma taxa acentuada na aceleração da convergência. Porém, tais operações alteram a esparsidade da matriz e portanto um processo de reordenamento da matriz é fundamental para a eficiência de tais precondicionadores.

Muitas pesquisas na área de minimização de largura de banda foram realizadas entre as décadas 60 e 80, com destaque para o os métodos RCM e GPS, algoritmos baseados em estratégias de busca em grafos e que proporcionam boa qualidade de solução, Cuthill & McKee (1969); Gibbs et al. (1976). Após esse período, observou-se que os trabalhos existentes na literatura dividem-se majoritariamente em propostas de melhoria ou aplicações de meta-heurísticas ao problema. No entanto, aplicações de métodos exatos para o problema também foram propostos (Martí et al. (2008); Caprara & Salazar (2005)). Porém, a dimensão das instâncias do problema é relativamente pequena: são apresentados experimentos somente para matrizes de dimensão menor que 1000.

Marti et al. (2001) propõem a utilização da Busca Tabu com movimentos baseados no conjunto de elementos críticos (ou seja, os elementos mais distantes da diagonal). Neste trabalho foi realizada a comparação entre este algoritmo, o GPS e uma implementação do Simulated Annealing (SA) proposta por Dueck & Jeffs (1995). Os resultados mostraram que a Busca Tabu apresentou a melhor qualidade de solução, porém o seu tempo computacional é bastante superior (no melhor caso chega a ser 50 vezes maior) ao tempo da heurística GPS. Já com relação a SA, a Busca Tabu também apresentou melhor qualidade de solução, mas um tempo de execução em torno de 20 vezes menor. Piñana et al. (2004) propuseram a aplicação do GRASP com reconexão de caminhos ao problema, utilizando o mesmo conjunto de matrizes no seus experimentos. Os resultados computacionais mostram que o GRASP obteve melhor qualidade de solução e menor tempo computacional do que a Busca Tabu apresentada por Marti et al. (2001), porém o tempo chega a ser milhares de vezes superior ao GPS. Lim et al. (2006) apresentaram abordagens para o problema utilizando algoritmos genéticos e o método *node shift*. A comparação destes métodos com o GPS e o GRASP apresentados em Piñana et al. (2004) foi realizada através de testes computacionais usando o mesmo conjunto de matrizes daqueles trabalhos. Os resultados mostram que ambas as abordagens propostas melhoram a qualidade de solução em relação às heurísticas comparadas. No entanto, com relação ao tempo de execução, o método *node shift* apresenta tempo de execução em torno de 3 vezes maior que o algoritmo genético, que por sua vez apresenta tempo maior do que a heurística GPS.

Uma generalização da heurística GPS foi proposta por Wang et al. (2009). Nesta proposta, novas estruturas de nível associadas ao grafo são criadas a partir de cada um dos vértices pseudo-periféricos. Em seguida os vértices de cada nível são renumerados com base na soma dos rótulos

dos vértices adjacentes. Os testes computacionais mostram que a proposta melhora discretamente a qualidade de solução, com um pequeno aumento no tempo de execução. Em uma abordagem diferente dos trabalhos citados anteriormente, a heurística proposta por Carvalho et al. (2009) não considera o problema representado por grafos e pode ser aplicada tanto para matrizes simétricas quanto para matrizes não simétricas. Os testes computacionais comparam a heurística proposta com o algoritmo Cuthill-Mckee (CM) e mostram que a heurística apresenta melhora na qualidade de solução, porém com um tempo de execução centenas de vezes maior do que o CM.

Em resumo, observa-se que todos os trabalhos citados melhoram a qualidade de solução, se comparados com os algoritmos mais clássicos e mais utilizados na literatura (RCM e GPS). Entretanto, nenhum desses trabalhos conseguiu simultaneamente melhorar a qualidade de solução e obter um tempo computacional equiparável a esses algoritmos. Isso reforça a questão de que muito ainda deve ser feito para que se consiga obter um melhor compromisso entre qualidade de solução e tempo de execução. Neste trabalho são implementados quatro algoritmos existentes na literatura para reordenamento de matrizes esparsas, com o intuito de comparar qualidade de solução e tempo de execução. Além disso, algumas modificações para melhoria de solução e redução de tempo de processamento destes algoritmos são propostas.

O trabalho está organizado como a seguir. Na próxima seção é apresentado um resumo das principais definições matemáticas necessárias para a compreensão dos algoritmos implementados. A seção 3 apresenta os métodos implementados nesse trabalho e as melhorias propostas nos algoritmos. Na sequência, a seção 4 mostra os detalhes de implementação e os testes computacionais são apresentados na seção 5. As principais conclusões e propostas de trabalhos futuros são apresentadas na última seção.

2 Definições Importantes

A seguir é apresentada uma definição matemática para largura de banda e envelope de uma matriz. Seja A uma matriz estruturalmente simétrica, isto é, se $a_{ij} \neq 0$ então $a_{ji} \neq 0$, mas não necessariamente $a_{ij} = a_{ji}$. A largura de banda de A ($lb(A)$) é definida como a maior distância do primeiro elemento não nulo até a diagonal, considerando todas as linhas da matriz e o envelope de A ($env(A)$) é a soma das distâncias do primeiro elemento não nulo até a diagonal, considerando também todas as linhas da matriz (Coleman (1984)). Ou seja:

$$\begin{aligned} b_i &= (i-j) \ \forall \ a_{ij} \neq 0, \ \ i = 2, \ldots, n \\ lb(A) &= \max_{i=2,\ldots,n} \{b_i\} \qquad (1) \\ env(A) &= \sum_{i=2}^{n} b_i \qquad (2) \end{aligned}$$

onde j é o índice da coluna do primeiro elemento não nulo da linha i. O problema de minimização de largura de banda é NP-completo. Devido a essa característica é difícil encontrar a solução exata para o problema em tempo computacional razoável, justificando a utilização de métodos heurísticos. Para propor uma solução para esse problema segundo a ótica da Combinatória, é necessário reformulá-lo por meio da representação por grafos: a matriz associada ao sistema linear pode ser diretamente representada pela matriz de adjacência de um grafo e o seu reordenamento, representado pelo problema de reordenação dos rótulos dos vértices do grafo. Para o melhor entendimento do problema representado por grafos, um resumo dos principais conceitos e propriedades desta estrutura matemática, relacionados ao problema, será apresentado a seguir.

Seja $G = (V, E)$, um grafo não-orientado formado por um conjunto de vértices V e um conjunto de arestas E. Dois vértices são adjacentes se existe uma aresta entre eles em G. Uma aresta incide em um vértice se ele for um dos extremos da aresta. O grau de um vértice é o número de arestas

incidentes a ele. Um *caminho* em G é definido como uma sequência de arestas de E que liga um vértice (vértice inicial) a outro (vértice final). Um *ciclo* é um caminho no qual o vértice inicial coincide com o vértice final. Diz-se que G é conexo se existe um caminho entre cada par de vértices de G. Se G é desconexo, é formado por subgrafos conexos, denominados *componentes conexas*. Um grafo T é uma *árvore* se é conexo e não tem ciclos. Os vértices de uma árvore com grau unitário são chamados *folhas*. Define-se ainda *distância*(u, v) como o custo do menor caminho existente entre os vértices v e w de V, $excentricidade(v)$ como a maior distância de v a todos os outros vértices de G. A $excentricidade(v)$ para $v \in T$, onde T é uma árvore, corresponde ao comprimento do caminho de v até uma folha da árvore. *Diâmetro*(G) é definido como o valor da maior excentricidade do grafo. *Pseudo-diâmetro* corresponde a uma alta excentricidade, porém não necessariamente a maior de todas. *Vértices periféricos* de G são vértices cuja excentricidade é igual ao diâmetro do grafo e *vértices pseudo-periféricos* são aqueles que apresentam altas excentricidades, mas não necessariamente a maior. *Estrutura de níveis com raiz associada* a G consiste em uma árvore enraizada com níveis. Uma árvore é enraizada se possui um vértice que se destaca dos outros (raíz). Um vértice v_i está no nível i de uma árvore enraizada se v_i está a uma distância i da raiz. Neste tipo de árvore, um vértice não pode fazer parte de dois níveis distintos (Menezes (1995)). A banda de uma matriz A é limitada pelo dobro da quantidade de vértices do nível com maior número de vértices (w) menos um, ou seja, $lb(A) \leq 2w - 1$. Denomina-se então, $w(x)$ o valor de w para uma estrutura de níveis definida a partir do vértice x.

3 Algoritmos Heurísticos para Minimização de Largura de Banda e Redução de Envelope

O método heurístico mais referenciado na literatura para resolução do problema de minimização de largura de banda e redução de envelope, e que também é conhecido por proporcionar boa qualidade de solução e baixo tempo de execução é o Cuthill Mckee (CM), Cuthill & McKee (1969). Este método tem como objetivo reordenar as linhas e colunas de uma matriz representada por um grafo. Nele, uma heurística de busca em largura para o caminhamento dos vértices do grafo é aplicada: partindo de um vértice inicial e caminhando pelas vizinhanças dos vértices em ordem crescente de graus, todos os vértices do grafo são visitados. A ordem de visita corresponde à permutação que deverá ser realizada na matriz ao final do procedimento. Pouco tempo depois, George (1971) observou que, com a inversão da ordem de numeração obtida pelo CM, o envelope da matriz podia ser reduzido, sem afetar a largura de banda. Assim foi definido o algoritmo Reverse Cuthill Mckee (RCM). Anos depois, Liu & Sherman (1976) provaram matematicamente a constatação de George (1971). Os bons resultados apresentados pelo RCM motivaram a partir de então a representação do problema de reordenamento de matrizes por grafos, Menezes (1995). Assim, podemos dizer que os algoritmos de reordenação de matrizes tem, em geral, como entrada um grafo simples $G(V, E)$ e o conjunto de vértices adjacentes de um vértice x em G ($Adj(x)$), e como saída uma numeração (rotulação) dos vértices de G. Neste contexto a heurística RCM é apresentada no Algoritmo 1.

Algoritmo 1: **RCM**

1 Encontrar um vértice inicial (x).
2 Renumerar os vértices iniciando com o vértice x.
3 **para** $i = 1, \ldots, |V|$ **faça**
4 Renumerar todos os vértices pertencentes ao conjunto $Adj(x)$, em ordem crescente de seus graus.
5 **fim para**
6 Inverter a numeração final.

O passo 1 do algoritmo consiste na procura por um vértice inicial. A escolha desse vértice está intimamente ligada a qualidade da solução. Alguns estudos nesta direção indicam que resultados com boa qualidade de solução estão ligados a vértices com altas excentricidades. Assim, a primeira abordagem realizada por Cuthill & McKee (1969), propôs iniciar a busca do vértice inicial por cada vértice com graus variando entre $D_{min} \leq D < D_{min} + 1/2D_{max}$, onde D_{min} é o menor grau do grafo e D_{max}, o maior. Essa abordagem torna a heurística RCM lenta, visto que muitos vértices podem ter graus variando em D. George & Liu (1979) propuseram uma heurística baseada na estrutura de nível de um vértice de grau mínimo, que realiza comparações entre valores de excentricidade para cada nova estrutura de nível dos vértices do último nível. O Algoritmo 2 descreve o cálculo de vértices pseudo-periféricos. Com o objetivo de melhorar a qualidade da solução e possivelmente

Algoritmo 2: **Escolha de vértice pseudo-periférico**

1 Escolha um vértice (x) de grau mínimo.
2 Construa a estrutura de nível para o vértice x.
3 **enquanto** *houver vertices* **faça**
4 Escolha um vértice y no último nível da estrutura de nível.
5 Construa a estrutura de nível de y.
6 **se** *excentricidade*(y) < *excentricidade*(x) *e* $w(y) < w(x)$ **então**
7 $x = y$
8 **fim se**
9 **fim enqto**
10 x é o vértice pseudo-periférico.

diminuir o tempo de processamento propomos duas modificações para o Algoritmo 1: (a) considerar o algoritmo de Dijkstra para a obtenção do vértice inicial ao invés de considerar o Algoritmo 2. Este novo algoritmo será denominado Algoritmo RCM-P1 e não será mostrado aqui, pois a menos do passo 1, é idêntico ao Algoritmo 1; (b) utilizar a estrutura de nível do vértice periférico obtida pela heurística proposta por George & Liu (1979) efetuando a renumeração dos vértices a partir de suas posições nos níveis, em ordem crescente de graus (Algoritmo 3, RCM-P2).

Algoritmo 3: **RCM-P2**

1 Encontrar um vértice pseudo-periférico (x) e sua respectiva estrutura de nível.
2 Renumerar os vértices iniciando com o vértice x (raiz da estrutura de nível).
3 **para** *cada nível i* **faça**
4 Renumerar todos os vértices pertencentes ao nível i, em ordem crescente de seus graus.
5 **fim para**
6 Inverter a numeração final.

Outro algoritmo bastante referenciado na literatura é o GPS, Gibbs et al. (1976). Sua idéia é similar ao RCM, porém utiliza como vértices iniciais, dois vértices, com a distância entre eles igual ao pseudo-diâmetro. O algoritmo encontra os vértices iniciais e em seguida, é criada uma estrutura de nível para cada um deles. As duas estruturas de nível são unidas por um processo definido como minimização de largura de nível. Por fim é efetuada a renumeração dos vértices de forma similar ao RCM. Os principais passos estão descritos no Algoritmo 4. Pode-se verificar que o mesmo algoritmo utilizado para encontrar o vértice pseudo-periférico, descrito anteriormente, também pode ser utilizado para encontrar o pseudo-diâmetro. A diferença principal entre os algoritmo GPS e RCM está no passo 2 do Algoritmo 4, pois a minimização da largura de nível requer um processamento adicional, detalhes podem ser vistos em Gibbs et al. (1976). Já no passo 3 desse mesmo algoritmo, a diferença está no fato da renumeração ser realizada sobre a nova estrutura obtida no passo 2. No

Algoritmo 4: **GPS**

1 Encontrar o pseudo-diâmetro, ou seja, os vértices iniciais x e y.
2 Minimizar a largura de nível através da união das estruturas de nível de x e y.
3 Renumerar o grafo de forma similar ao RCM.

algoritmo proposto em Gibbs et al. (1976), foi realizado uma comparação entre a heurística GPS e a heurística RCM. O algoritmo GPS obteve qualidade de solução equiparável ao RCM, entretanto o tempo de execução era em torno de dez a vinte vezes menor. Com a heurística proposta por George & Liu (1979) para encontrar o vértice pseudo-periférico (Algoritmo 2), observa-se que o algoritmo RCM passa a ser mais rápido que o GPS.

Entre a publicação do RCM e do GPS um outro trabalho apresentado por George (1973) introduziu o algoritmo ND, que no processo de reordenamento encontra um conjunto de colunas que divide a matriz em $k \geq 2$ conjuntos disjuntos (Algoritmo 5, ND). Observa-se que o fato de se

Algoritmo 5: **ND**

1 Encontrar um conjunto de colunas S cuja remoção divide a matriz A em p matrizes disjuntas X_i para $i = 1, \ldots, p$.
2 Renomear as colunas das matrizes X_i de forma seus índices sejam menores do que os índices da matriz X_{i+1} que por sua vez devem ser menores do que os índices do conjunto S.
3 Aplicar os passos 1 e 2 recursivamente em cada submatriz X_i até que um número máximo de k partições seja alcançado.

encontrar um conjunto de colunas que divide a matriz k matrizes, pode ser interpretado como um problema de particionamento de grafos. Neste problema, deseja-se obter o menor subconjunto de arestas (corte) que particione o grafo em k conjuntos disjuntos (k componentes conexas do grafo original). O conjunto S do Algoritmo 5 corresponde ao conjunto de vértices formado por apenas um dos extremos de cada aresta desse corte (vértices separadores). Devido a essa característica, várias versões do algoritmo ND vem sendo propostas e heurísticas de particionamento de grafos são muito utilizadas. O algoritmo ND implementado nesse trabalho utiliza uma heurística baseada na adjacência do grafo, realizando uma busca em largura que introduz cada vértice visitado em uma partição. O processo de busca termina quando o valor $|V|/2$ é atingido. Além disso, utilizamos para cada partição obtida nesse processo a heurística Fiduccia Mattheyses (Fiduccia & Mattheyses (1982)), baseada em cortes por arestas, para realizar um refinamento das partições. Para o algoritmo ND propomos duas modificações: (a) se rotularmos os vértices de forma que os rótulos de X_i sejam menores que os rótulos de S que por sua vez sejam menores que os rótulos de X_{i+1}, definimos um novo algoritmo denominado ND-P3; (b) se renumerarmos as partições que consideram o conjunto S formado por todos os vértices extremos das arestas pertencentes ao corte de arestas, definimos um outro algoritmo, que será denominado neste trabalho por ND-P4.

Por fim, o último algoritmo implementado foi o Espectral (ES), proposto por Barnad et al. (1995) e baseado na utilização de autovalores e autovetores da matriz laplaciana do grafo $G(V, E)$ (Algoritmo 6, ES). O passo 1 deste algoritmo possuem uma complexidade elevada, no entanto, pode ser minimizada pela forma de armazenamento da matriz laplaciana. Detalhes serão descritos na próxima seção.

4 Implementação computacional

Todos os algoritmos implementados nesse trabalho utilizam a matriz armazenada por uma estrutura denominada *Compress Sparse Row* (CSR), Saad (2003). Os algoritmos RCM e GPS foram

Algoritmo 6: **ES**

Calcular o segundo menor autovalor da matriz laplaciana (L) e seu autovetor, v, associado.
Ordenar v em ordem crescente e aplicar a permutação em L, gerando L_1.
Calcular $x = env(L_1)$.
Ordenar v em ordem decrescente e aplicar a permutação em L, gerando L_2.
Calcular $y = env(L_2)$.
se $x < y$ **então**
    Reordenar a matriz com base no autovetor em ordem crescente.
senão
    Reordenar a matriz com base no autovetor em ordem decrescente.
fim se

implementados utilizando a heurística proposta por George & Liu (1979), para a escolha do vértice pseudo-periférico e do pseudo-diâmetro. Vale ressaltar que o RCM, implementado neste trabalho considera o algoritmo proposto por Cuthill & McKee (1969), com o reverso descrito por George (1971). O algoritmo GPS foi implementado conforme descrito em Gibbs et al. (1976). Todos os processos de ordenamento necessários aos algoritmos RCM, GPS, ES e ND foram realizados utilizando o algoritmo *QuickSort*. O cálculo da esparsidade é feito em percentual, isto é, $esp(A) = ((n * n) - nnz)/(n * n) * 100$, onde n é o total de vértices e nnz é o total de arestas do grafo, ou seja, n é o número de linhas, e nnz é o número de elementos não nulos da matriz A. O tamanho do envelope e a largura de banda são apresentadas em forma de percentual de redução com relação ao tamanho inicial. O cálculo da redução é feito por:

$$red(env(A)) = 100 - ((env(A)_{\textit{após reordenamento}} * 100)/env(A)_{\textit{inicial}}) \tag{3}$$

onde $env(A)_{\textit{inicial}}$ é o tamanho do envelope inicial e $env(A)_{\textit{após reordenamento}}$ é o tamanho do envelope após o reordenamento da matriz A. O mesmo cálculo é feito para a redução da largura de banda ($red(lb(A))$ trocando $env(A)$ por $lb(A)$.

As matrizes utilizadas nos testes computacionais estão no formato Matrix Market conforme descrito por Boisvert et al. (1996) e disponíveis nos sites: http://math.nist.gov/MatrixMarket/ e http://www.cise.ufl.edu/research/sparse/matrices/. São oriundas das mais diversas áreas de aplicação como por exemplo dinâmica dos fluidos computacional, eletromagnetismo, química, etc. Para o cálculo do autovalor e autovetor associado, necessários no algoritmo ES, foram testadas algumas bibliotecas: Clapack, Blopex, Trilinos e Chaco 2.0. A biblioteca escolhida foi o Chaco, devido ao armazenamento otimizado da matriz e ao baixo tempo de execução. Ressaltamos que o Chaco implementa dois algoritmos para esse fim, Lanczos indicado para matrizes de tamanho pequeno e médio e o Multilevel Symmlq/RQI, utilizado nesse trabalho, indicado para matrizes de grande porte (Hendrickson & Leland (1995)). Este algoritmo considera o método iterativo RQI (Rayleigh Quociente Iteration) que resolve um sistema linear pela decomposição LQ em cada iteração, Barnad & Simon (1993). Cabe ressaltar que a qualidade da solução obtida depende fortemente da tolerância considerada para o processo iterativo. Destacamos ainda, que alteramos o código fonte do Chaco para que somente o autovetor associado ao segundo menor autovalor seja calculado com o objetivo de diminuir o tempo de processamento.

5 Testes Computacionais

Os testes computacionais realizados nesse trabalho comparam os algoritmos RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4, GPS e ES. As métricas analisadas foram o tamanho do envelope ($env(.)$), a largura de banda ($lb(.)$) e o tempo de processamento (CPU). Os testes foram realizados

em um computador com processador INTEL Core 2 Duo 2.2GHz, com 3GB de memória RAM e sistema operacional Ubuntu 9.04.

A Tab. 1 apresenta as principais características do conjunto de matrizes escolhidas para analisar os métodos implementados. Observamos que todas as matrizes escolhidas possuem esparsidade superior a 99%. A Fig. 1(a) apresenta o tempo de CPU (em segundos) dos algoritmos RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4, GPS e ES para o conjunto de matrizes apresentados na Tab. 1. A Fig. 1(b) apresenta a redução do tamanho da envelope $red(env(.))$ e a Fig. 1(c), a redução do tamanho da banda $red(lb(.))$ de todas as matrizes analisadas.

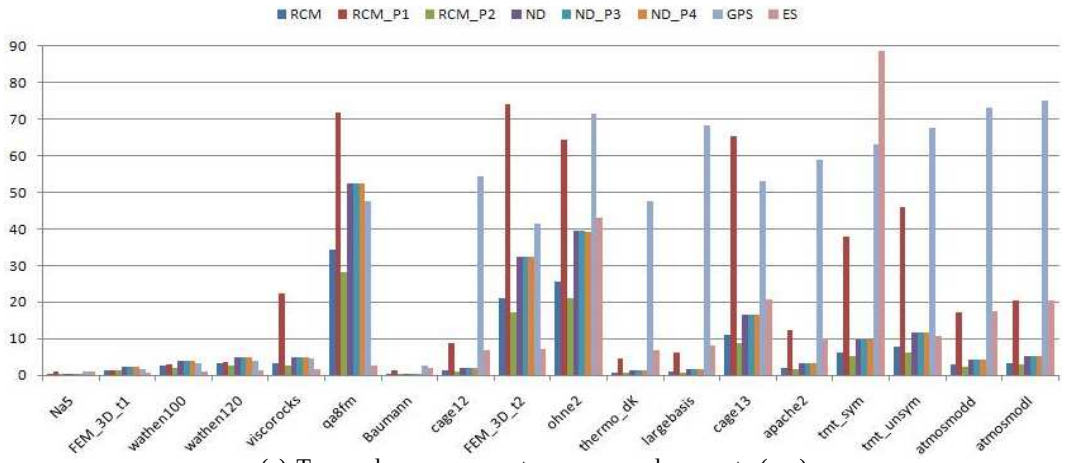
Tabela 1: Características do conjunto de matrizes escolhidas

Matriz (A)	Aréa de aplicação	nnz	n	$esp(.)$
Na5	Problema da química quântica	305630	5832	99,10
FEM_3D_t1	Problema Térmico	430740	17880	99,87
wathen100	Problema randômico 2D/3D	471601	30401	99,95
wathen120	Problema randômico 2D/3D	565761	36441	99,96
viscorocks	Problema Materiais	1133641	37762	99,92
qa8fm	Problema Acústico	1660579	66127	99,96
Baumann	Problema Químico	748331	112211	99,99
cage12	Eletroforese de DNA	2032536	130228	99,99
FEM_3D_t2	Problema Térmico	3489300	147900	99,98
ohne2	Problema de dispositivos semicondutores	11063545	181343	99,97
thermomech_dK	Problema Térmico	2846228	204316	99,99
largebasis	Problema de Otimização	5240084	440020	99,99
cage13	Eletroforese de DNA	7479343	445315	99,99
apache2	Problema Estrutural	4817870	715176	99,99
tmt_sym	Problema Eletromagnético	5080961	726713	99,99
tmt_unsym	Problema Eletromagnético	4584801	917825	99,99
atmosmodd	Problema da dinâmica dos fluidos computacional	8814880	1270432	99,99
atmosmodl	Problema da dinâmica dos fluidos computacional	10319760	1489752	99,99

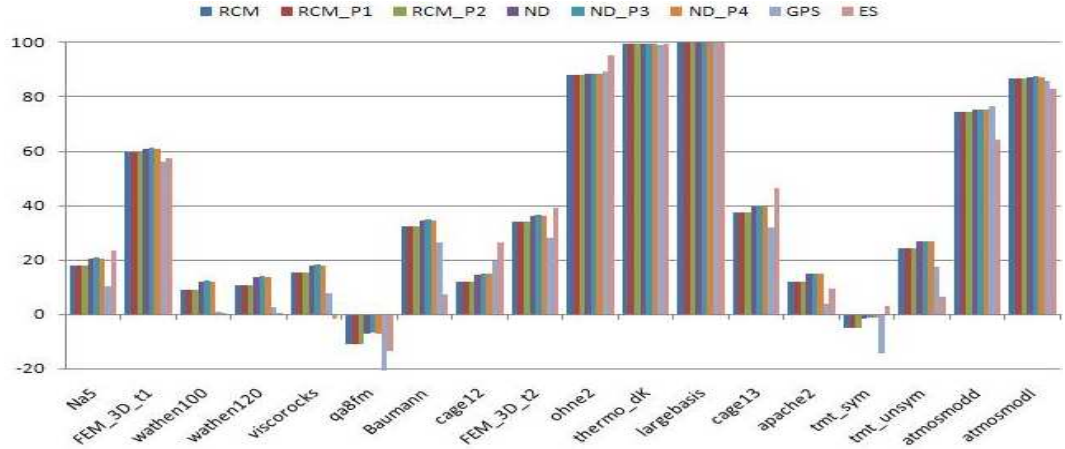
Podemos observar, a partir da Fig. 1(a), que os tempos de execução do algoritmo RCM-P1 para todas as matrizes testadas foi maior do que o tempo obtido pelo RCM. Isso pode ser explicado pela necessidade, no RCM-P1, de executar o algoritmo Dijkstra para todos os vértices do grafo associado à matriz, aumentando sua complexidade computacional de $O(nnz * log(n))$ para $O(n * (nnz * log(n)))$. Os casos com tempos equiparáveis ocorrem quando a heurística para encontrar o vértice inicial (passo 1 do algoritmo RCM) é executada no seu pior caso. Observamos ainda que em 65% dos testes realizados, a redução alcançada pelo algoritmo RCM-P1 foi a mesma do algoritmo RCM, tanto para $env(.)$ (Fig. 1(b)) quanto para $lb(.)$ (Fig. 1(c)). Este fato ocorre pois, na maioria das matrizes testadas, o Algoritmo 2 encontra o vértice de maior excentricidade (periférico). O algoritmo proposto RCM-P2 possui tempo de execução menor que o RCM para todas as matrizes testadas apresentando qualidade da solução similar ao RCM. A redução do tempo é explicada por não ser mais necessária a execução do passo 2 do algoritmo RCM, visto que o Algoritmo 2 é aplicado sobre o grafo ordenado em ordem crescente de graus, fazendo com que a estrutura de nível obtida seja equivalente ao passo 2 do RCM.

O algoritmo ND apresenta um tempo de execução, na média, em torno de 50% maior que o tempo obtido pelo RCM apresentando uma discreta melhora na qualidade de solução. Podemos observar que os tempos de execução das adaptações propostas (algoritmos ND-P3 e ND-P4) equivalem aos tempos do ND, sendo que o ND-P4 é discretamente mais rápido. Esse fato pode ser explicado por não ser mais necessário identificar o conjunto dos vértices separadores. Neste caso os

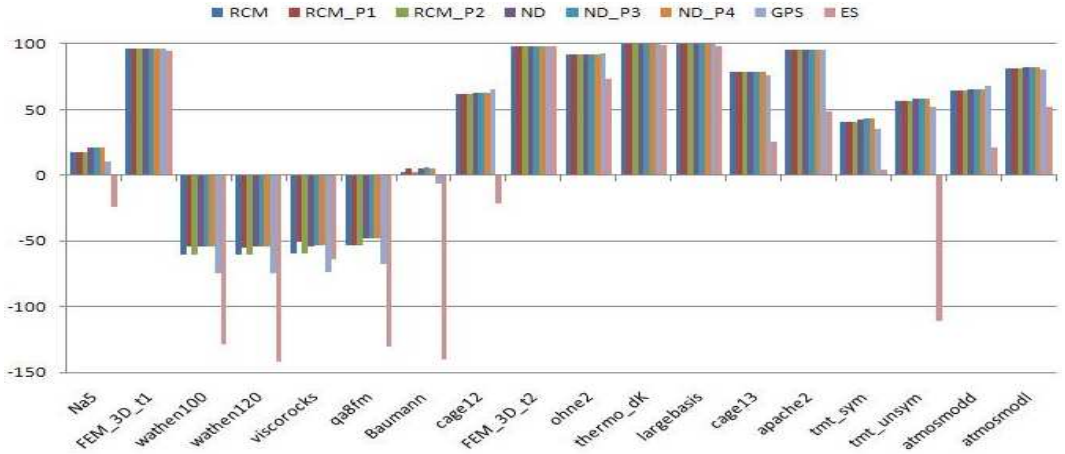
vértices extremos de todas as arestas do corte são utilizados. Observa-se também uma pequena melhora quanto à qualidade de solução. O algoritmo GPS se mostrou bem mais lento que o RCM, não havendo diferença significativa quanto à qualidade de solução. Este fato se explica já que o passo 2 deste algoritmo é computacionalmente mais custoso e demanda maior utilização de memória.



(a) Tempo de processamento para o reordenamento (seg)



(b) Redução do tamanho do envelope $Red(env(.))$



(c) Redução do tamanho da banda $Red(lb(.))$

Figura 1: Tempo de processamento e redução do tamanho do envelope e da banda do conjunto de matrizes analizadas para os algoritmos RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4, GPS e ES

O algoritmo ES apresenta tempos de execução menores que o RCM para algumas das matrizes testadas (por exemplo: FEM_3D_t1, wathen120, qa8fm, FEM_3D_t2) e maiores para outras (por exemplo: cage12, apache2, tmt_sym, atmosmodl). Os casos de melhor desempenho do ES são caracterizados por matrizes menos esparsas. Em geral, observamos que os algoritmos baseados em estratégias de busca em largura (RCM, RCM-P1, RCM-P2, ND, ND-P3, ND-P4 e GPS), se mostraram extremamente sensíveis a esparsidade da matriz. Esse comportamento pode ser explicado pela complexidade computacional da busca em largura que é $O(n + nnz)$. Por outro lado, podemos dizer que o gargalo computacional do ES é o cálculo do autovalor e autovetor. Além de ser uma operação custosa, quanto maior for a precisão exigida para os componentes do autovetor, maior é o tempo de processamento deste cálculo e melhor será a qualidade da solução. Quanto maior a precisão, maior a possibilidade das componentes do autovetor serem distintas, fato que contribui para a melhoria na qualidade da solução. A Fig. 2(a) mostra a configuração original da matriz

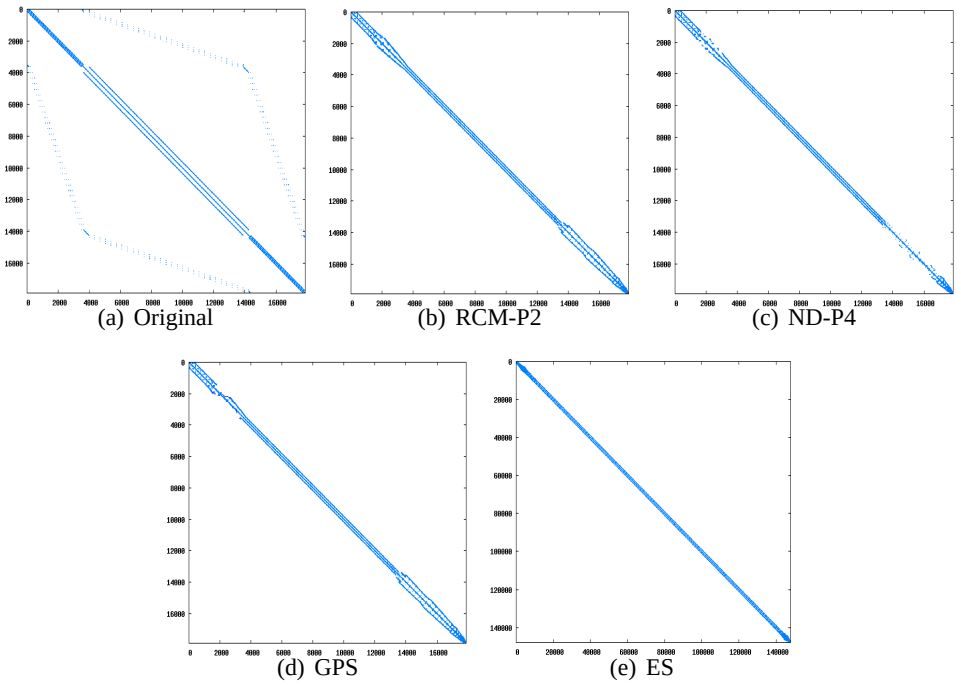

(a) Original
(b) RCM-P2
(c) ND-P4
(d) GPS
(e) ES

Figura 2: Configuração da matriz FEM-3D-t1 para os algoritmos RCM-P2, ND-P4, GPS e ES

esparsa FEM_3D_t1 e as Figs. 2(b), 2(c), 2(d), 2(e) mostram, respectivamente, as configurações obtidas pelos algoritmos RCM-P2, ND-P4, GPS e ES. Podemos observar que todos os algoritmos

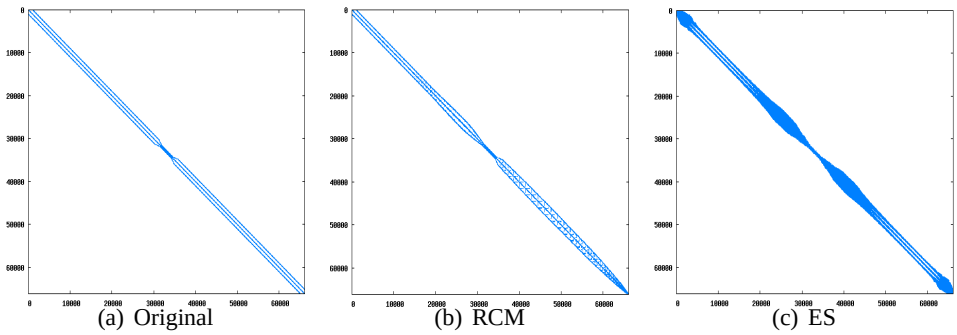

(a) Original
(b) RCM
(c) ES

Figura 3: Configuração da matriz qa8fm para os algoritmos RCM e ES

reduzem a banda e o envelope da matriz. Porém, observamos nas Figs. 1(b) e 1(c) que existe um

grupo de matrizes, para as quais todos os algoritmos testados pioram a qualidade da solução inicial, como por exemplo a matriz qa8fm. Para os algoritmos baseados em busca em largura, podemos notar que este fato acorre em grafos que possuem rotulações próximas umas das outras. No algoritmo ES, não há melhoria na qualidade da solução quando os valores das componentes do autovetor são muito próximos uns dos outros. Este fato ocorre quando as distâncias entre os vértices são próximas. A Fig. 3 mostra a configuração original da matriz qa8fm e as configurações para os algoritmos RCM e ES onde pode ser observado que não houve redução de largura de banda nem de envelope.

6 Conclusões e Trabalhos Futuros

Neste trabalho foi realizado um estudo de quatro – RCM, GPS, ND e ES – dos principais algoritmos de reordenamento de matrizes existentes na literatura, com o intuito de comparar qualidade de solução e tempo de execução. Além disso, foram propostas modificações para o RCM (RCM-P1 e RCM-P2) e para o ND (ND-P3 e ND-P4). Os testes computacionais foram realizados sobre um conjunto de dezoito matrizes esparsas de grande porte com esparsidade superior a 99%.

A modificação proposta para o algoritmo RCM, denominada RCM-P2, apresenta o menor tempo em todos os casos testados, com qualidade de solução idêntica ao RCM. A proposta ND-P4 apresentou os melhores resultados quando comparada com os demais algoritmos ND, mesmo sendo mais lenta que o RCM. O algoritmo ES possui solução equiparável ao RCM e se comportou melhor em termos de tempo de execução, apenas para as matrizes menos esparsas. Para todas as matrizes testadas, o tempo de execução do GPS foi maior que do RCM, uma vez que a minimização da largura de nível (passo 2 do Algoritmo 4) é computacionalmente mais custosa do que realizar uma busca em largura (passo 3 do Algoritmo 1). Estes fatos justificam a larga utilização do algoritmo RCM para reordenamento de matrizes esparsas. Podemos observar também que as propostas de modificação apresentadas neste trabalho apresentaram soluções melhores que aquelas obtidas pelos algoritmos originais levando em consideração o equilíbrio entre qualidade de solução e tempo computacional.

Como trabalhos futuros, pretendemos estudar a influência dos algoritmos de reordenamento de matrizes na resolução de sistemas lineares utilizando métodos iterativos não estacionários. Além disso, desejamos investigar o comportamento dos algoritmos de Grau Mínimo e Colônia de formigas aplicados ao problema de reordenamento de matrizes. Outro estudo previsto diz respeito à escolha do vértice inicial para o algoritmo RCM: pode-se variar a escolha desse vértice tendo como base os vértices de menor grau, de maior grau, de grau aleatório para escolha do vértice periférico.

Referências

Barnad, S., Photen, A., & Simon, H. (1995). A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Aplications*, 3:317–334.

Barnad, S. & Simon, H. (1993). A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Proceedings of the 6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 711–718. SIAM.

Benzi, M., Szyld, D. B., & van Duin, A. (1999). Ordering for incomplete factorization preconditioning of nonsymmetric problems. *SIAM J. Sci. Comput*, 20(5):1652–1670.

Boisvert, R., Pozo, R., & Remington, K. (1996). The matrix market exchange formats: Initial design. Technical report, National Institute of Standards and Technology.

Camata, J., Rossa, A., Valli, A., Catabriga, L., Carey, G., & Coutinho, A. (2010). Reordering in incomplete factorization preconditioning for serial and parallel amr solutions. applied to *Int. J. Numer. Meth. Engng.*

Caprara, A. & Salazar, G. (2005). Laying out sparse graphs with provably minimum bandwidth. *INFORMS Journal on Computing*, 17(3):356–373.

Carvalho, M., Junqueira, N., & Soma, N. (2009). Uma heurística para o problema de minimização de largura de banda em matrizes. *Pesquisa operacional*, pages 2886–2897.

Coleman, T. (1984). *Large sparse numerical optimization*. Springer-Verlag Berlin Heidelberg.

Cuthill, E. & McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th ACM national conference*, pages 157–172, New York, NY, USA. ACM Press.

Dueck, G. & Jeffs, J. (1995). A heuristic bandwidth reduction algorithm. *Journal of Combinatorial Mathematics and Combinatorial Computing*, pages 97–108.

Fiduccia, C. & Mattheyses, R. (1982). A linear-time heuristic for improving network partitions. In *Proceedings of the IEEE 19th Design Automation Conference,*, pages 175–181.

George, A. (1973). Nested dissection of a regular finite element mesh. *SIAM Journal of Numerical Analysis*, 10:345–363.

George, A. & Liu, W. (1979). An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software*, 5:236–250.

George, J. A. (1971). *Computer implementation of the finite element method*. Tese de Doutorado, Stanford University, Stanford, CA, USA.

Gibbs, N. E., Poole, W. G., & Stockemeyer, P. K. (1976). An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal of Numerical Analysis*, 13(2):236–250.

Hendrickson, B. & Leland, R. (1995). The chaco user's guide version 2.0. Technical Report SAND-95-2344, Sandia National Laboratories, New Maxico, USA.

Lim, A., Rodrigues, B., & Xiao, F. (2006). Heuristics for matrix bandwidth reduction. *European Journal of Operational Research*, 174:69–91.

Liu, W. & Sherman, A. (1976). Comparative analysis of the cuthill-mckee and the reverse cuthill-mckee ordering algorithms for sparse matrices. *SIAM J. Num. Anal.*, 13:198–213.

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

Marti, R., Laguna, M., Gloves, F., & Campos, V. (2001). Reducing the bandwidth of a sparse matrix with tabu search. *European Journal of Operational Research*, 135:450–459.

Menezes, I. (1995). *Técnicas de reordenação para solução de sistemas esparsos*. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ.

Piñana, E., Plana, I., Campos, V., & Marti, R. (2004). Grasp and path relinking for the matrix bandwidth minimization. *European Journal of Operacional Research*, 153:200–210.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Second edition, Boston.

Wang, Q., Guo, Y., & Shi, X. (2009). A generalized gps algorithm for reducing the bandwidth and profile of a sparse matrix. *Progress In Electromagnetics Research*, 153(PIER 90):121–136.