ALGORITMOS DE REORDENAMENTO DE MATRIZES ESPARSAS APLICADOS A PRECONDICIONADORES ILU(p)

Brenno Lugon
Lucia Catabriga
Universidade Federal do Espírito Santo - UFES
Departamento de Informática
Av. Fernando Ferrari, 514 - Goiabeiras - 29075910 - Vitória/ES - Brasil
brennolugon@gmail.com, luciac@inf.ufes.br

RESUMO

O uso de precondicionadores para solução de sistemas lineares de equações é uma técnica eficiente para acelerar a convergência de métodos iterativos não estacionários. Nesse contexto, podemos reduzir o tempo de execução desses métodos aplicando uma simples troca de linhas e colunas na matriz dos coeficientes. Esse processo, chamado de reordenamento, visa reduzir o número de operações com ponto flutuante durante a montagem das matrizes precondicionadoras. Este trabalho faz um estudo comparativo dos algoritmos de reordenamento: *Sloan*, *Reverse Cuthill Mckee*, *Espectral*, *Nested Dissection* e *Approximate Minimum Degree*, avaliando o impacto que causam quando utilizamos o método de resolução GMRES com o precondicionador baseado na fatoração LU incompleta (ILU(p)) para sistemas lineares cuja matriz dos coeficientes é esparsa.
PALAVRAS CHAVES. matrizes esparsas, reordenamento, precondicionadores ILU, GMRES.

ABSTRACT

The use of preconditioning for solving linear systems of equations is an efficient technique to accelerate the convergence of nonstationary iterative methods. In this context, we can reduce runtime of these methods by applying a simple exchange of rows and columns in the coefficient matrix. This process, called reordering, aims to reduce the number of floating point operations during the preconditioner matrices computation. This work makes a comparative study of reordering algorithms: *Sloan*, *Reverse Cuthill Mckee*, *Spectral*, *Nested Dissection* and *Approximate Minimum Degree*, evaluating the impact they have when using the solver method GMRES applied to a preconditioner based on incomplete LU factorization, ILU(p), for linear systems whose coefficient matrix is sparse.
KEYWORDS. sparse matrix, reordering, ILU preconditioner, GMRES.

1 Introdução

No processo de solução numérica de muitas aplicações é comum recairmos na necessidade de manipular matrizes de grande porte vinculadas a sistemas lineares. Tais aplicações podem possuir milhões de variáveis, mas cada variável individual depende apenas de algumas poucas, e por isso a maior parte dos coeficientes dessas matrizes são nulos. Dinâmicas de fluidos computacionais, simulação de problemas físicos e químicos, grafos, sistemas eletromagnéticos são exemplos de aplicações cujos processos de solução recaem na necessidade de manipular matrizes esparsas.

O uso de estruturas de dados otimizadas para armazenar matrizes esparsas visa reduzir a quantidade de memória usada e a quantidade de operações de ponto flutuante. Armazenamentos como o Compressed Sparse Row (CSR) e Compressed Sparse Column (CSC) são exemplos de estruturas eficientes que armazenam apenas os elementos não nulos da matriz e consequentemente melhoram o desempenho dos algoritmos que necessitam executar operações com matrizes esparsas.

Para solução de sistemas lineares de grande porte, Saad (2003) sugere a utilização de métodos iterativos baseados em projeções de subespaços de *Krylov* por suas boas propriedades numéricas e computacionais. Além disso, técnicas de precondicionamento baseadas na decomposição LU incompleta (Benzi et al., 1999; Camata et al., 2012) aceleram a convergência e consequentemente melhoram a eficiência desses métodos na obtenção da solução.

O problema de reordenamento de linhas e colunas de uma matriz esparsa pode ser redefinido como o problema de rerrotulação do grafo associado a matriz esparsa. Pesquisas feitas nas décadas passadas desenvolveram algoritmos como o RCM (Cuthill & McKee, 1969) e GPS (Gibbs et al., 1976) que se baseiam em estratégias de busca em grafos e proporcionam boa qualidade de solução. Podemos destacar o artigo de (Benzi et al., 1999) que testou algoritmos como o RCM e Nested Dissection (George, 1973) em métodos de solução de sistemas como o Bi-CGStab, GMRES e TFQMR (Saad, 2003). Ghidetti et al. (2011) comparou alguns algoritmos de reordenamento como o RCM e Espectral (Barnard et al., 1993) avaliando o impacto que causam no tempo de CPU do método GMRES com precondicionador ILU(p). Carmo (2005) também comparou a influência de algoritmos de reordenamento RCM e AMD (Davis et al., 1994) aplicados em matrizes esparsas simétricas no desempenho do método Cholesky Controlado Gradiente Conjugado CCCG(n).

Neste trabalho, iremos estudar classes de algoritmos de reordenamento que têm por objetivo a redução do preenchimento (ou *fill-in*) que ocorre ao utilizarmos precondicionador baseado na decomposição LU incompleta. Assim, espera-se diminuir o número de operações com ponto flutuante, melhorando o tempo de execução dos métodos iterativos não estacionários.

2 Método de solução GMRES precondicionado

O método do Resíduo Mínimo Generalizado (GMRES) desenvolvido por Saad & Schultz (1986) é um método iterativo não-estacionário utilizado para determinar a solução aproximada de um sistema linear da forma $Ax = b$. Sendo x_0 uma condição inicial, $\{v_j\}_{j=1,n}$ uma base de um espaço vetorial de *Krylov* e $y_1, y_2, ..., y_n$ coeficientes de uma combinação linear, a solução do sistema pode ser representada por

$$x = x_0 + \sum_{j=1,n} y_j v_j \tag{1}$$

A fim de reduzir o número de operações de ponto flutuante e o armazenamento, o algoritmo pode ser implementado considerando um número fixo k de elementos na base. Essa técnica, chamada de reinicialização (ou *restart*) (Saad, 2003), consiste em gerar um novo espaço vetorial de *Krylov* com k elementos na base a cada k iterações. A aproximação é representada por

$$x_{i+k} = x_i + \sum_{j=1,k} y_j v_j \tag{2}$$

O precondicionamento é um ingrediente chave para o sucesso de métodos iterativos. A ideia básica é substituir o sistema $Ax = b$ pelo sistema $M^{-1}Ax = M^{-1}b$, onde M^{-1} deve ser uma boa aproximação de A^{-1}. Na prática, ao invés de determinar M^{-1} explicitamente, a ação do precondicionador se dá na operação produto matriz-vetor, $p = M^{-1}Av$. Primeiro, calcula-se $z = Av$, e em seguida, a ação de M^{-1} em $p = M^{-1}z$ é calculada resolvendo o sistema linear $Mp = z$. O precondicionador usado neste trabalho é baseado na fatoração aproximada de $M = \tilde{L}\tilde{U}$ chamada de fatoração LU incompleta (ou $ILU(p)$). O parâmetro p determina quantos níveis de preenchimento serão considerados no processo da decomposição LU e os fatores $\tilde{L}$ e $\tilde{U}$ são os fatores aproximados obtidos na fatoração incompleta.

3 Métricas de Minimização

O processo de fatoração LU quando aplicado em uma matriz esparsa tende a preencher muitos elementos nulos, diminuindo consideravelmente o grau de esparsidade da matriz. Com isso, o tempo computacional para resolver o sistema por um método iterativo não-estacionário com precondicionador pode aumentar muito, uma vez que precisamos calcular os fatores $\tilde{L}$ e $\tilde{U}$ da matriz precondicionada M. Heurísticas baseadas na simples troca de linhas e colunas da matriz, visam reduzir o preenchimento (ou *fill-in*) gerado na construção de M, consequentemente diminuindo o número de operações ponto flutuante no método iterativo de solução.

Seja A uma matriz estruturalmente simétrica, a *largura de banda* de A é definida como:

$$lb(A) = \max_{i=1,..,n}\{b_i\} \tag{3}$$

$$b_i = (i - j) \; \forall \; a_{ij} \neq 0, \; i = 1, ..., n \tag{4}$$

ou seja, a *largura de banda* pode ser definida como a maior distância entre o primeiro elemento não-nulo da linha i até a diagonal principal (Alan George, 1994).

Seja A uma matriz estruturalmente simétrica, o *envelope* de A é definido como:

$$env(A) = \sum_{i=1}^{n} b_i \tag{5}$$

Assim, podemos definir o *envelope* como a soma das distâncias entre o primeiro elemento não-nulo de cada linha i até a diagonal principal (Alan George, 1994).

4 Algoritmos de Reordenamento

4.1 Sloan

Proposto por Sloan (1986), o método visa reduzir o envelope e largura de banda de matrizes esparsas estruturalmente simétricas. O algoritmo funciona atribuindo estados e prioridades a cada vértice, mantendo uma fila de possíveis escolhas. Essa fila vai sendo atualizada com informações do grafo e o vértice com a maior prioridade é o próximo a ser escolhido para receber o novo rótulo. O algoritmo implementado neste trabalho foi baseado em Sloan (1986) e Sloan (1989).

4.2 Reverse Cuthill McKee (RCM)

Proposto por Cuthill & McKee (1969), o algoritmo Cuthill Mckee é um método de reordenamento aplicado a matrizes esparsas para redução do envelope e largura de banda. Estudos feitos posteriormente por George (1971) mostraram que a solução do Cuthill Mckee pode ser melhorada simplesmente invertendo a ordem de numeração. Foi então definido o Reverse Cuthill

Mckee, que passou a ser umas das heurísticas mais utilizadas para o problema de minimização da largura de banda e do envelope devido a boa qualidade de solução, o baixo tempo de execução e a facilidade de implementação. Neste trabalho adotou-se a estratégia de implementação sugerida por Ghidetti (2011); Ghidetti et al. (2010). A autora percebeu que o processo de percorrer os vértices adjacentes do grafo em ordem crescente de graus é equivalente a uma busca em largura neste grafo, que é feita no processo de encontrar o vértice inicial usado pelo algoritmo.

4.3 Espectral

Proposto por Barnard et al. (1993), o algoritmo Espectral tem como principal objetivo reduzir o tamanho do envelope. O novo reordenamento é calculado a partir da permutação do vetor de autovetores associado ao segundo menor autovalor λ_2 da matriz laplaciana associada ao grafo. Segundo Fiedler (1973), o segundo menor autovalor λ_2 define a conectividade algébrica e seu autovetor associado tem relação com a distância entre os vértices do grafo. Para encontrar os autovalores e autovetores da matriz utiliza-se neste trabalho a biblioteca CHACO 2.2 (Hendrickson & Leland, 2013) que implementa o algoritmo Multilevel Symmlq/RQI para o cálculo desses autovalores e autovetores. Este algoritmo é baseado no método iterativo RQI (Rayleigh Quocient Iteration) cuja qualidade da solução depende da tolerância escolhida.

4.4 Nested Dissection (ND)

O algoritmo Nested Dissection, proposto por George (1973), tem como objetivo reduzir o preenchimento (*fill-in*) em fatorações de matrizes esparsas. Consiste em encontrar um conjunto S de vértices separadores de um grafo G, cuja remoção divide G em dois subgrafos disjuntos, A e B. O subgrafo A é rerrotulado primeiro, em seguida B, e por último o conjunto de vértices separadores S. Assim, os vértices separadores são movidos para o final da matriz, e o processo é aplicado recursivamente para A e B. O desafio do algoritmo é encontrar o conjunto de vértices separadores, que divide G em duas partições. Logo, um bom método para bisseção de G impacta diretamente na qualidade da solução. Este trabalho faz uso da biblioteca METIS (Karypis, 2013), que implementa heurísticas eficientes baseadas em algoritmos multinível (Karypis & Kumar, 1998).

4.5 Approximate Minimum Degree (AMD)

Algoritmos de Grau Mínimo são heurísticas muito usadas para redução de preenchimento (*fill-in*) da fatoração LU aplicadas em matrizes esparsas de grande porte. Esses algoritmos baseiam-se na evolução do grau dos vértices, eliminando a cada passo o vértice de menor grau, fazendo com que seus vizinhos se tornem adjacentes. O novo grafo gerado pela remoção de um vértice é chamado de Grafo de Eliminação. O AMD usa técnicas baseadas em grafos quocientes que permite obter aproximações com baixo custo computacional para o grau mínimo. Davis et al. (1994) mostra que essas aproximações são frequentemente iguais ao grau do vértice, e melhoram o tempo do algoritmo AMD com relação aos outros algoritmos de Grau Mínimo. Neste trabalho utilizamos o algoritmo AMD implementado na biblioteca de Timothy A. Davis (2013b), versão 2.3.1.

5 Testes computacionais

A Tabela 1 apresenta as principais características de um conjunto de matrizes estruturalmente simétricas usadas nos testes computacionais. Essas matrizes estão no formato Matrix Market disponíveis nos repositórios da Universidade da Flórida em (Timothy A. Davis, 2013a). Uma descrição mais detalhada de todos os algoritmos e testes pode ser consultada no relatório técnico TEC-REP-01/2013 disponível em http://www.lcad.inf.ufes.br/wiki/index.php/Relatórios_Técnicos.

Os testes apresentados foram realizados em um notebook com processador Intel Core i5-460M 2.53GHz, com 4GB de memória RAM e sistema operacional Ubuntu 12.10.

Nome	k	nnz	n	ε	spa	Área de Aplicação
RAIL_5177	200	35.185	5.177	10^{-10}	99,87	Transferência de Calor
AFT01	50	125.567	8.205	10^{-10}	99,81	Problema Acústico
FEM_3D_THERMAL1	5	430.740	17.880	10^{-10}	99,87	Problema Térmico
THERMOMECH_TK	150	711.558	102.158	10^{-06}	99,99	Problema Térmico
BAUMANN	200	748.331	112.211	10^{-10}	99,99	Problema Químico
DUBCOVA2	20	1.030.225	65.025	10^{-10}	99,97	Problema de EDP
BONES01	300	6.715.152	127.224	10^{-10}	99,96	Modelagem 3D

Tabela 1: Características das matrizes apresentadas.

As Tabelas 2 e 3 apresentam uma análise geral do conjunto de matrizes testadas, onde foi escolhido o nível de preenchimento para a fatoração ILU que obteve o melhor comportamento. A Tabela 2 refere-se a porcentagem de redução do preenchimento gerado, considerando o preenchimento obtido sem aplicação de algoritmos de reordenamento. A Tabela 3 apresenta o tempo total de processamento (em segundos) da solução do sistema linear pelo método GMRES com o precondicionador ILU(p) considerando o tempo sem reordenamento e com reordenamento para os algoritmos descritos. É importante ressaltar que as comparações devem ser efetuadas por linha, uma vez que os níveis de preenchimentos e as esparsidades das matrizes são distintos.

Matriz	Preenchimento	Sloan (%)	RCM (%)	Espectral (%)	ND (%)	AMD (%)
RAIL_5177	ILU(20)	73,02	69,47	63,80	88,73	90,10
AFT01	ILU(2)	26,96	28,37	-13,75	-34,19	-2,17
FEM_3D_THERMAL1	ILU(1)	4,64	4,14	1,26	-38,90	-41,46
THERMOMECH_TK	ILU(8)	55,18	51,68	51,37	58,89	60,78
BAUMANN	ILU(5)	36,27	36,27	12,57	44,51	46,00
DUBCOVA2	ILU(4)	89,48	89,47	82,99	87,99	90,10
BONES01	ILU(1)	24,56	26,61	57,20	12,35	17,03

Tabela 2: Porcentagem de redução do preenchimento.

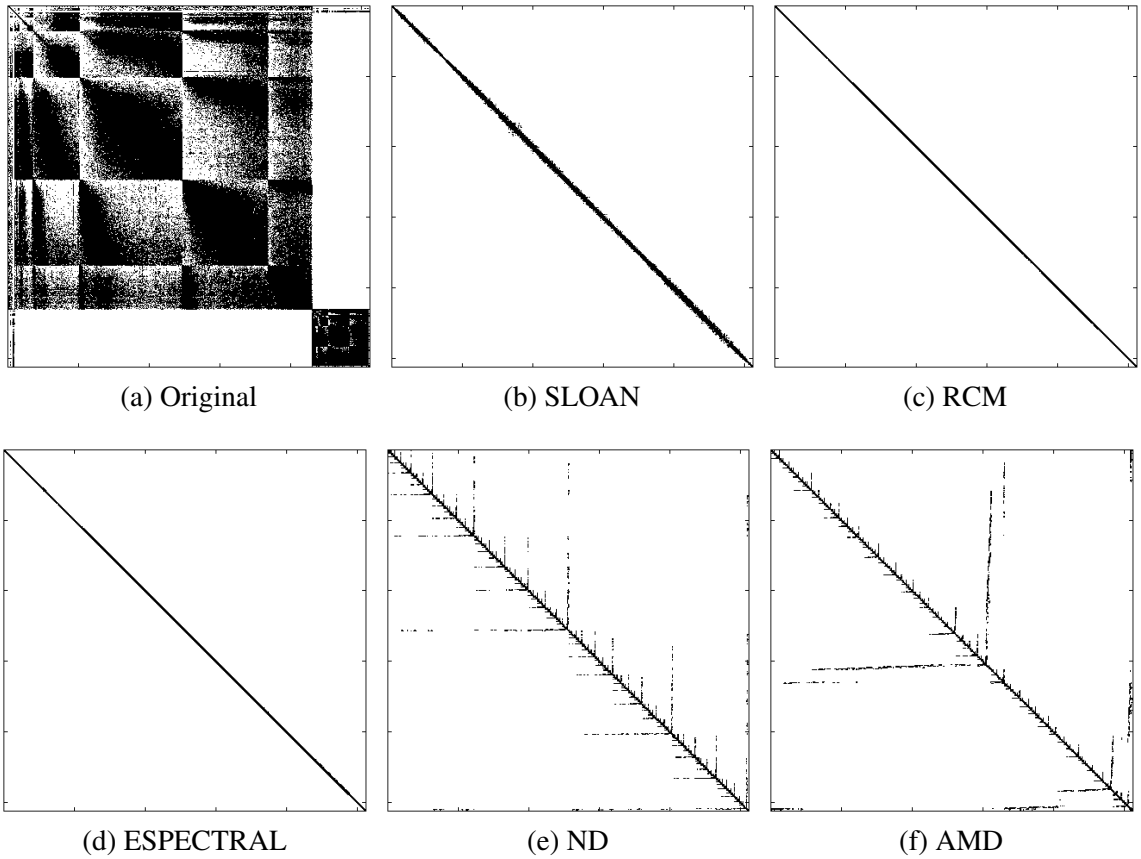
Matriz	Preenchimento	S/ reord.	Sloan	RCM	Espectral	ND	AMD
RAIL_5177	ILU(20)	31,78	0,78	1,04	1,92	0,40	0,33
AFT01	ILU(2)	0,80	0,77	0,80	0,85	1,71	1,76
FEM_3D_THERMAL1	ILU(1)	2,72	2,96	2,82	2,76	5,27	4,83
THERMOMECH_TK	ILU(8)	95,87	44,07	43,45	45,45	52,72	55,04
BAUMANN	ILU(5)	148,95	97,11	58,66	69,92	73,56	76,63
DUBCOVA2	ILU(4)	978,78	20,37	18,99	23,24	23,82	21,24
BONES01	ILU(1)	372,92	307,84	242,10	221,28	392,03	307,08

Tabela 3: Tempo total de CPU (em segundos).

Analisando as Tabelas 2 e 3, podemos perceber que nem sempre a melhor redução do preenchimento garante o melhor tempo computacional. Isso se dá uma vez que o reordenamento altera o condicionamento das matrizes, fazendo com que elas necessitem de mais ou menos iterações para convergir. Entretanto, nas matrizes $rail\_5177$, $aft01$ e em $bones01$ o melhor preenchimento resultou no melhor tempo de CPU.

A seguir, apresenta-se uma análise detalhada de duas matrizes da Tabela 1. O símbolo † significa que o algoritmo GMRES não convergiu, onde o critério decisivo para a não-convergência foi a análise da redução do resíduo a cada iteração do algoritmo. Nas tabelas de Tempo de CPU o símbolo • indica o melhor resultado para cada nível de preenchimento. Já o símbolo • indica os tempos que não foram vantajosos, ou seja, o tempo para resolver a matriz reordenada não foi melhor que o tempo para resolvê-la sem reordenamento.

5.1 Matriz $thermomech\_TK$



(a) Original (b) SLOAN (c) RCM

(d) ESPECTRAL (e) ND (f) AMD

Figura 1: Esparsidades da matriz $thermomech\_TK$.

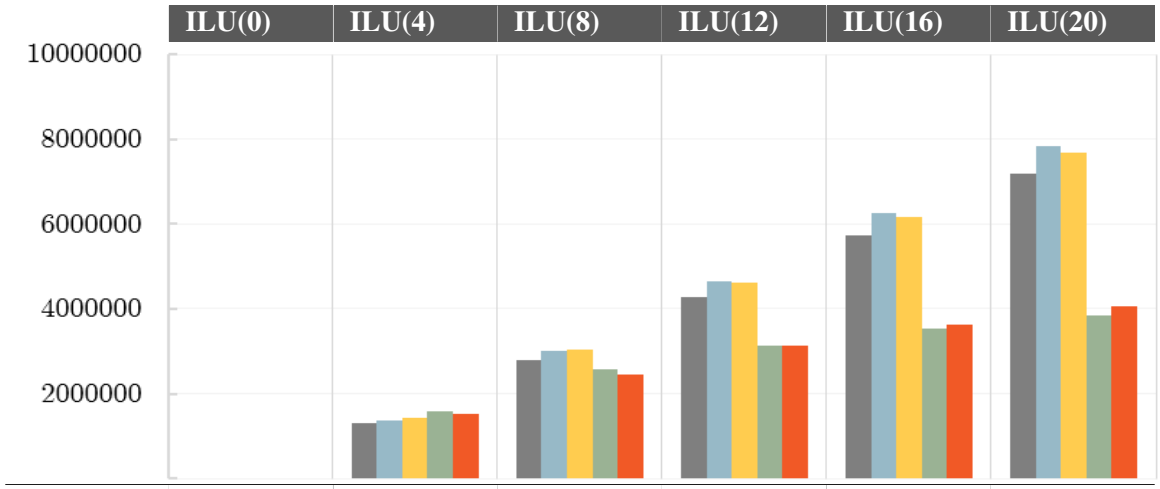
	env(.)	lb(.)	Tempo de CPU (seg)
Original	2.667.823.445	102.138	-
Sloan	16.072.717	676	2,112
RCM	17.882.411	253	0,581
Espectral	14.635.485	389	2,738
ND	-	-	0,863
AMD	-	-	0,192

Tabela 4: Medições da matriz $thermomech\_TK$.

É interessante observar as esparsidades de cada matriz reordenada na Figura 1. Os algoritmos Sloan, RCM e Espectral foram capazes de aproximar os elementos não nulos bastante dispersos na Figura 1a para perto da diagonal principal nas Figuras 1b, 1c e 1d.

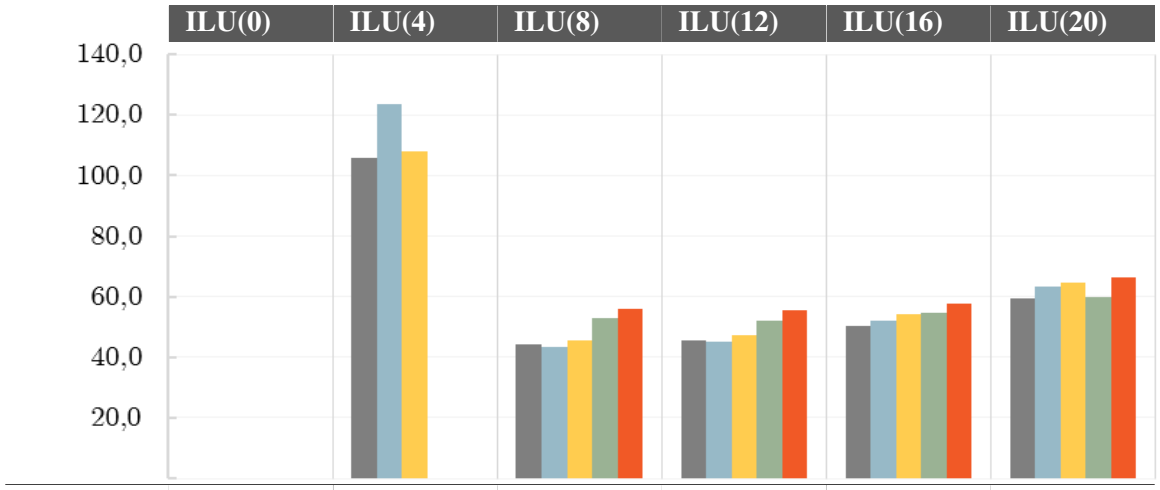
Com relação a Tabela 4, podemos destacar o Espectral com melhor redução do envelope, porém maior tempo de CPU, e o RCM com melhor redução de largura de banda e pior redução de envelope.

Todos os algoritmos reduziram consideravelmente o preenchimento na Figura 2, mantendo um comportamento bem semelhante ao longo dos ILU's. Quanto ao tempo de CPU (Figura 3) percebemos que as matrizes geradas pelos algoritmos não convergem para o ILU(0). A não-convergência se mantém em ILU(4) para o ND e AMD, passando a convergir com o Sloan, RCM e Espectral. Todos os tempos foram vantajosos, sendo o melhor resultado para o RCM em ILU(8).



S/ reord.	-	2.418.982	6.234.694	11.386.296	17.735.910	25.168.278
Sloan	-	1.299.766	2.794.478	4.281.708	5.745.876	7.175.972
RCM	-	1.377.086	3.012.390	4.648.154	6.261.578	7.840.310
Espectral	-	1.416.644	3.031.658	4.619.810	6.168.354	7.666.906
ND	-	1.594.042	2.563.166	3.137.532	3.530.792	3.833.678
AMD	-	1.505.844	2.445.502	3.118.058	3.636.438	4.055.210

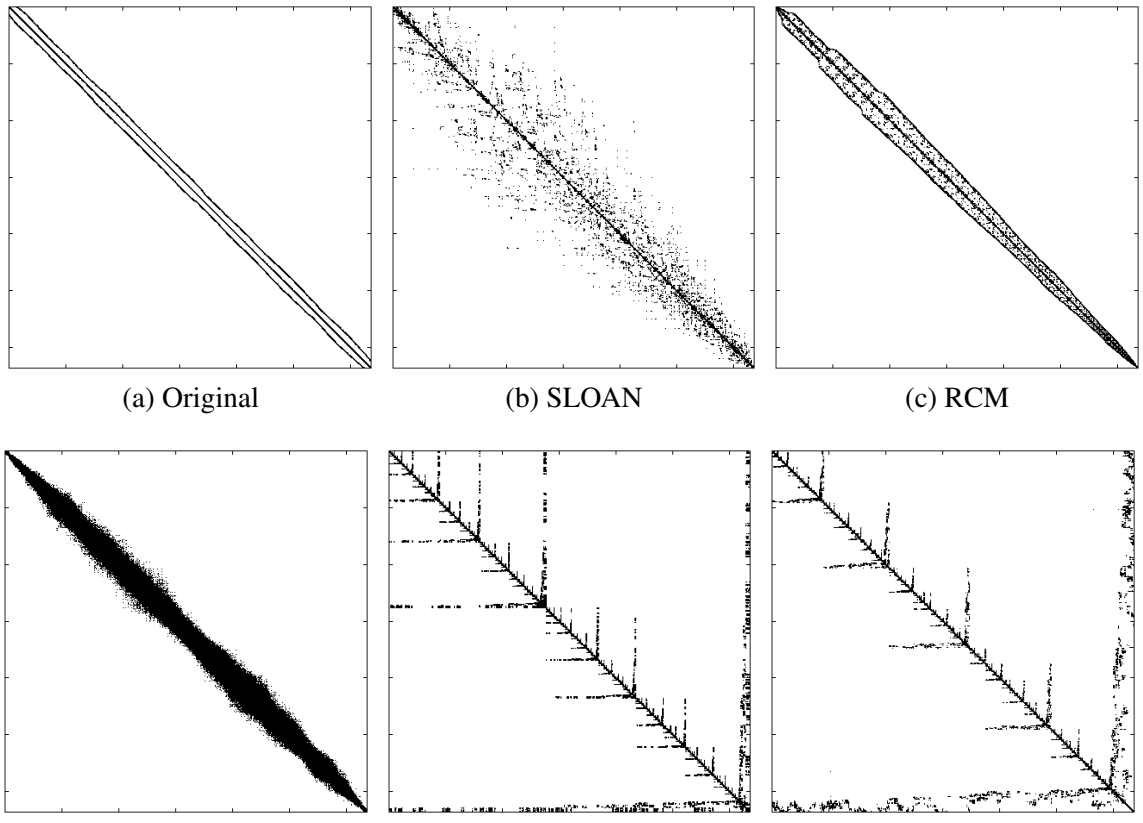
Figura 2: Preenchimento da matriz $thermomech\_TK$.



S/ reord.	†	†	95,87	244,41	797,94	2155,92
Sloan	†	105,90 •	44,07	45,71	50,32 •	59,32 •
RCM	†	123,45	43,45 •	45,11 •	52,04	63,24
Espectral	†	107,80	45,45	47,09	54,03	64,46
ND	†	†	52,72	51,90	54,73	59,82
AMD	†	†	55,94	55,47	57,61	66,34

Figura 3: Tempo de CPU da matriz $thermomech\_TK$ (em segundos).

5.2 Matriz $boneS01$



(a) Original (b) SLOAN (c) RCM

(d) ESPECTRAL (e) ND (f) AMD

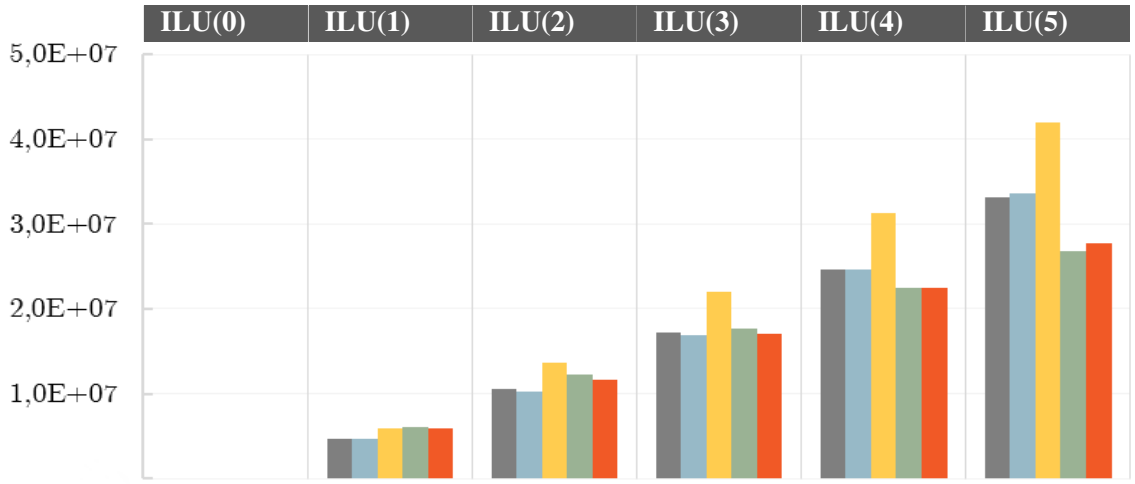
Figura 4: Esparsidades da matriz $boneS01$.

	env(.)	lb(.)	Tempo de CPU (seg)
Original	331.330.356	3.722	-
Sloan	317.794.617	49.757	102,027
RCM	391.130.964	7.604	41,944
Espectral	308.790.384	13.622	3,932
ND	-	-	0,985
AMD	-	-	0,067

Tabela 5: Medições da matriz $boneS01$.

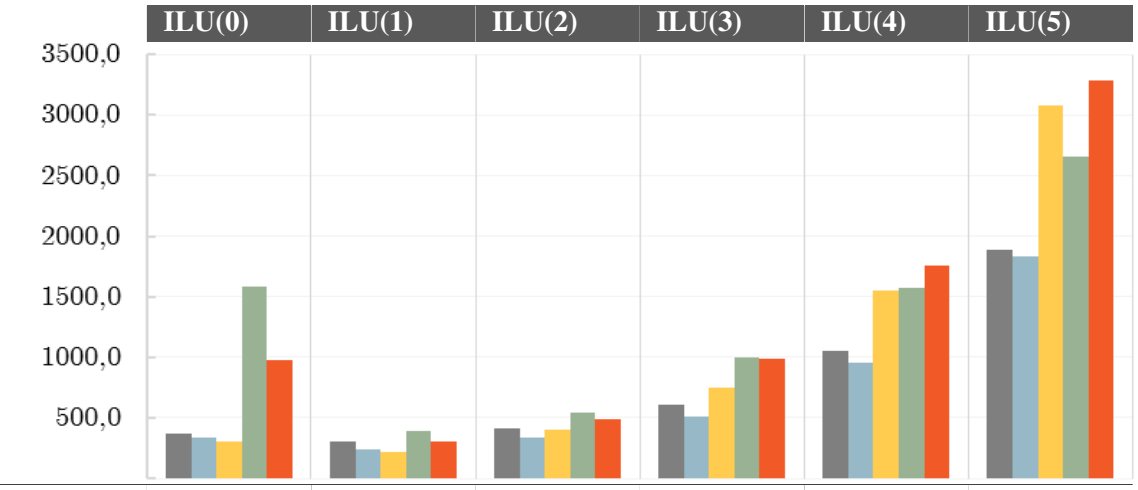
Os dados da Tabela 5 mostram que o Espectral foi o algoritmo que mais reduziu o envelope. Já a largura de banda aumentou em todos os algoritmos, com destaque para o Sloan que além de ter tido a pior largura de banda, teve pior desempenho computacional. O padrão de esparsidade de matriz $boneS01$ (Figura 4) já apresenta os elementos não nulos próximos da diagonal principal, portanto a ação dos algoritmos de reordenamento testados não apresentam resultados aparentemente satisfatórios quando analisadas as métricas largura de banda e envelope.

Na Figura 5 observamos que o Sloan e RCM tiveram o menor preenchimento nos níveis 1,2 e 3, perdendo para o ND e AMD nos níveis seguintes. Apesar disso, a redução do preenchimento no ND e AMD não foi capaz de diminuir o tempo de CPU (Figura 6). Mais do que isso, os tempos finais do ND e AMD não foram, na maioria das vezes, ao menos vantajosos. Diferente dos algoritmos Sloan, RCM e Espectral que, apesar de pouco, reduziram os tempos de CPU em todos os níveis de preenchimento. O menor tempo ocorreu com o algoritmo Espectral em ILU(1).



S/ reord.	-	6.467.202	13.943.844	22.348.404	31.932.252	42.936.012
Sloan	-	4.622.274	10.519.650	17.157.564	24.662.070	33.182.568
RCM	-	4.586.148	10.233.648	16.854.336	24.583.032	33.636.384
Espectral	-	5.967.684	13.586.562	21.973.428	31.306.446	41.928.624
ND	-	6.116.868	12.221.472	17.629.524	22.429.008	26.729.430
AMD	-	5.932.548	11.569.346	17.072.138	22.464.376	27.746.400

Figura 5: Preenchimento da matriz $boneS01$.



S/ reord.	850,93	372,92	439,06	786,47	1641,07	3300,51
Sloan	364,73	307,84	417,29	605,25	1047,95	1884,14
RCM	335,38	242,10	336,44 •	513,85 •	958,76 •	1835,23 •
Espectral	303,54 •	221,28 •	405,99	753,66	1551,48	3082,77
ND	1582,48 •	392,03 •	538,84 •	1001,08 •	1570,17	2651,05
AMD	981,76 •	307,08	492,61 •	983,66 •	1761,45 •	3287,21

Figura 6: Tempo de CPU da matriz $boneS01$ (em segundos).

6 Conclusões e Trabalhos Futuros

Para cada algoritmo estudado, a Tabela 6 mostra suas principais características, com base no conjunto de matrizes testadas.

	Preenchimento	Iterações	Tempo de CPU	Implementação	Condicionamento
Sloan	●◐	●	●◐	▲	■■
RCM	●●	●●	●	▲	■
Espectral	●●●	◐	●●	▲▲	■
ND	●	●●●	●●●	▲▲▲	■■■
AMD	◐	●●◐	●●◐	▲▲▲	■■■

Tabela 6: Análise final.

A primeira coluna da tabela refere-se a quantidade de preenchimento gerado por cada algoritmo. Quanto mais ●, maior é o preenchimento gerado pelo algoritmo. Como pudemos perceber nos testes, os algoritmos ND e AMD foram os que, em geral, mais reduziram o preenchimento, enquanto que o Espectral foi o que causou menos reduções. Já o Sloan e o RCM tiveram reduções bem próximas.

O número de iterações, apresentado na segunda coluna da tabela, também deve ser lido da mesma forma que o preenchimento. Quanto mais ●, maior o número necessário de iterações para a matriz reordenada convergir pelo método GMRES. Podemos notar que os algoritmos que tiveram pouco preenchimento, como ND e AMD, tiveram que realizar mais iterações para convergir. Da mesma forma, o elevado preenchimento causado pelo Espectral fez diminuir a quantidade de iterações necessárias.

Quanto ao tempo de CPU final na terceira coluna, destacamos o algoritmo RCM como o mais vantajoso, seguido pelo Sloan e Espectral. Analisando alguns dos testes, podemos reparar na tendência do algoritmo RCM de reduzir mais a largura de banda do que o envelope. Já o Sloan e o Espectral tendem a reduzir mais o envelope do que a largura de banda. Podemos conjecturar, então, que a redução da largura de banda tem um impacto mais positivo no tempo final de CPU do que a redução do envelope. Tal análise não inclui os algoritmos ND e AMD que não reordenam segundo essas métricas.

A quarta coluna mede o quão fácil é a implementação de cada algoritmo. Quanto menos ▲, mais fácil é a implementação. Portanto, o RCM foi categorizado como o mais fácil de se implementar, seguido pelo Sloan. Esses dois algoritmos foram os únicos totalmente implementados pelos autores desse trabalho, sendo a única dificuldade que possa surgir ao implementá-los é na construção da estrutura de nível para encontrar os vértices pseudo-periféricos. Para os algoritmos Espectral, ND e AMD foram utilizadas parcialmente ou totalmente bibliotecas, cujos códigos foram adaptados para nossas estruturas. Esses algoritmos foram classificados como médio, difícil e difícil, respectivamente. Para o algoritmo Espectral, não é simples construir um procedimento para encontrar o autovetor associado ao segundo menor autovalor. O ND recai na necessidade de construir um método de bisseção que divide o grafo em duas partições, além de precisar encontrar o conjunto de vértices separadores. Já no algoritmo AMD é necessário construir um Grafo de Eliminação a cada passo, tornando sua implementação não trivial.

Por fim, a quinta coluna refere-se ao condicionamento das novas matrizes geradas pelos reordenamentos. Muitos ■ significam que o reordenamento, no geral, piorou o condicionamento

da matriz. Logo, baseado nos testes com a matriz $thermomech\_TK$, podemos observar que os algoritimos RCM, Sloan e Espectral tendem a melhorar o condicionamento, enquanto que nos algoritmos ND e AMD este fato não ocorreu.

Analisando globalmente a Tabela 6, os algoritmos com menos itens, são considerados aqueles que obtiveram o melhor desempenho para o conjunto de matrizes analisadas. Logo, nessa análise, o RCM foi o que apresentou melhores resultados, seguido do Sloan e Espectral. Apesar disso, o ND e AMD tiveram casos muito mais vantajosos para níveis de preenchimento grandes.

Um fato interessante é que, as matrizes com os elementos não nulos muito dispersos, longe da diagonal principal, tiveram todos os tempos de CPU vantajosos para todos os algoritmos. Podemos perceber isso analisando as figuras da configuração da esparsidade da matriz $thermomech\_TK$ (Figura 1). A qualidade de redução do tempo já não foi sempre vantajosa para a matriz $boneS01$ (Figura 4) que já possui os elementos não nulos muito próximos da diagonal principal.

Como trabalhos futuros, destaco o estudo do comportamento dos algoritmos em um conjunto de matrizes com elementos não nulos muito dispersos da diagonal. Esse conjunto de matrizes poderia influenciar no resultado final comparativo da Tabela 6. Além disso, pretendemos estudar implementações paralelas desses algoritmos.

Agradecimentos

O primeiro autor agradece a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de mestrado concedida e a segunda autora agradece ao CNPq pelo apoio recebido dentro do escopo dos projetos CNPq 552630/2011-0 e 307020/2012-6.

Referências

Alan George, Joseph Liu, E. N. (1994). *Computer Solution of Sparse Linear Systems*.

Barnard, S. T., Pothen, A., & Simon, H. D. (1993). A spectral algorithm for envelope reduction of sparse matrices. In *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, Supercomputing '93, pages 493–502, New York, NY, USA. ACM.

Benzi, M., Szyld, D. B., & Duin, A. V. (1999). Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM J. SCI. COMPUT*, 20(5):1652–1670.

Camata, J., Rossa, A., Valli, A., Catabriga, L., Valli, A. M. P., Carey, G., & Coutinho, A. (2012). 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.

Carmo, F. C. (2005). Análise da influência de algoritmos de reordenação de matrizes esparsas no desempenho do método CCCG(n). Dissertação de Mestrado, Universidade Federal de Minas Gerais.

Cuthill, E. & McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, ACM '69, pages 157–172, New York, NY, USA. ACM.

Davis, T. A., Amestoy, P., Duff, I. S., Iain, & Duff, S. (1994). An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17:886–905.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305.

George, A. (1973). Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363.

George, J. A. (1971). *Computer implementation of the finite element method*. Tese de Doutorado, Calif School of Humanities and Sciences Stanford Department of Computer Science, Stanford, CA, USA. AAI7205916.

Ghidetti, K., Boeres, M., & Catabriga, L. (2010). Um estudo comparativo de métodos heurísticos para reordenamento de matrizes esparsas. In *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional*, volume 1, pages 2066–2077, Bento Gonçalves. Rio de Janeiro. SOBRAPO.

Ghidetti, K., Catabriga, L., Boeres, M., & Rangel, M. (2011). A study of the influence of sparse matrices reordering algorithms for ilu(p) preconditioner on the gmres method. In *Proceedings of the Fifth SIAM Workshop on Combinatorial Scientific Computing*, Technical Report - Aachener Informatik-Bericht (AIB) 2011-09, pages 125–127, Darmstadt, German. Darmstadt: RWTH Aachen University.

Ghidetti, K. R. (2011). O impacto do reordenamento de matrizes esparsas nos métodos iterativos não estacionários precondicionados. Dissertação de Mestrado, Universidade Federal do Espírito Santo.

Gibbs, N. E., Poole, W. G., & Stockmeyer, P. K. (1976). An algorithm for reducing the profile and bandwidth of a sparse matrix. *SIAM J. Numer. Anal.*, 13:236–250.

Hendrickson, B. & Leland, R. (2013). Chaco: Software for partitioning graphs. http://www.sandia.gov/ bahendr/chaco.html.

Karypis, G. (2013). METIS - a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.

Karypis, G. & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392.

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

Saad, Y. & Schultz, M. H. (1986). GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869.

Sloan, S. W. (1986). An algorithm for profile and wavefront reduction of sparse matrices. *Internacional Journal for Numerical Methods in Engineering*, 23:239–251.

Sloan, S. W. (1989). A fortran program for profile and wavefront reduction. *An Algorithm for Profile and Wavefront Reduction of Sparse Matrices*, 28:2651–2679.

Timothy A. Davis, Yifan Hu, A. R. (2013a). The University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices/.

Timothy A. Davis, Patrick R. Amestoy, I. S. D. (2013b). AMD: approximate minimum degree ordering. http://www.cise.ufl.edu/research/sparse/amd/.