Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

An experimental analysis of Hager's exchange methods in conjunction with heuristics for bandwidth and profile reductions applied to accelerate the ICCG method

Sanderson L. Gonzaga de Oliveira and Júnior Assis Barreto Bernardes[1]
Departamento de Ciência da Computação, UFLA, Lavras, MG

Abstract. This paper evaluates the original Hager's adjacent exchange methods in conjunction with heuristics for bandwidth and profile reductions with the objective of reducing computing times of the zero–fill incomplete Cholesky–preconditioned conjugate gradient method. The numerical results obtained in this computational experiment show that the original Hager's exchange methods, although capable of reducing the profile of the instances, are not useful when reducing processing times of the zero–fill incomplete Cholesky–preconditioned conjugate gradient method.

Keywords. Sparse matrices, graph labeling, graph algorithm, bandwidth reduction, profile reduction, conjugate gradient method.

1 Introduction

A fundamental step in various applications in science and engineering is the solution of large-scale sparse linear systems in the form $Ax = b$, where A is an $n \times n$ large-scale sparse matrix, x is the unknown n–vector solution which is sough, and b is a known n–vector. Generally, it is a step in the simulation that demands a high computing time.

Paging policies and modern hierarchical memory architecture favor programs that consider locality of reference into account [4]. An appropriate vertex labeling is desirable for the low–cost solution of large-scale and sparse linear systems. An adequate vertex labeling provides that the corresponding coefficient matrix A will have narrow bandwidth and small profile [4]. Furthermore, this does not depend on the storage scheme used to represent the matrix. Thereby, the use of a heuristic for bandwidth and profile reductions is an alternative to achieve a sequence of graph vertices with spatial locality. Heuristics for bandwidth and profile reductions are employed to obtain low computing times for solving large-scale sparse linear systems [4].

Let $A = [a_{ij}]$ be an $n \times n$ symmetric matrix associated with an undirected graph $G = (V, E)$ composed of a set of vertices V and a set of edges E. The bandwidth of row i is $\beta_i(A) = i - \min\limits_{1 \leq j \leq i} [j : a_{ij} \neq 0]$. The overall bandwidth $\beta(A)$ is defined as $\beta(A) = \max\limits_{1 \leq i \leq n} [\beta_i(A)]$. Equivalently, the bandwidth of G for a vertex labeling

[1]sanderson@dcc.ufla.br,jrassis@posgrad.ufla.br

$S = \{s(v_1), s(v_2), \cdots, s(v_{|V|})\}$ (i.e., a bijective mapping from V to the set $\{1, 2, \cdots, |V|\}$) is $\beta(G) = \max[(v \in V)\ (\{v, u\} \in E)\ |s(v) - s(u)|]$. The profile of A can be defined as $profile(A) = \sum_{i=1}^{n} \beta_i(A)$ (or equivalently, $profile(G) = \sum\limits_{v \in V} [(\{v, u\} \in E)\ |s(v) - s(u)|]$). The problems of bandwidth and profile minimizations are NP-hard [8, 10].

A prominent method for solving large sparse linear systems is the conjugate gradient method (CGM), used when the coefficient matrix is symmetric and strictly diagonally dominant. Among several preconditioners that researchers have proposed (e.g., see [3]) for the conjugate gradient method, the incomplete Cholesky (IC) factorization is especially important (see [4] and references therein). IC(l) with $l > 0$ will yield a better approximation to A than IC(0) at the cost of increased storage and processing times so that we employed the IC(0) preconditioner. In particular, practitioners have been successfully applying the zero–fill incomplete Cholesky–preconditioned conjugate gradient (ICCG for short) method in the solution of several problems (see [4] and references therein).

A previous publication [4] evaluated several heuristics for bandwidth and profile reductions when applied to various large-scale instances contained in the SuiteSparse matrix collection [1] with the objective of reducing the ICCG method. This publication [4] indicated the most promising heuristics for each of the instances. Hager's adjacent exchange methods [5] were proposed to deliver further profile reductions in matrices after the use of a heuristic for profile optimization. This paper evaluates these methods in conjunction with heuristics for bandwidth and profile reductions to verify whether Hager's adjacent exchange methods [5] along with state-of-the-art heuristics for bandwidth and profile reductions can reduce execution times of the ICCG method. To the best of our knowledge, this is the first (published) instance in the field to evaluate Hager's exchange methods along with heuristics for bandwidth and profile reductions with the objective of reducing the execution times of the preconditioned conjugate gradient method.

The remainder of this manuscript is structured as follows. Section 2 gives a brief description of Hager's adjacent exchange methods. Section 3 describes how we conducted the experiments in this computational experiment. Section 4 presents and analyzes the results. Finally, Section 5 provides the conclusions.

2 A brief description of Hager's adjacent exchange methods

Hager proposed the down and up exchange methods [5], which, in short, perform symmetric permutations of rows and columns in a matrix to reduce its profile. The down exchange algorithm is first applied, followed by the up exchange algorithm, and they are executed alternately for a number of times previously defined.

To save space, we give only a brief notion of Hager's adjacent exchange methods below. Considering an $n \times n$ matrix A and $k < l \leq n$, let $D_{k:l}(A) = |\{j \geq k : k < f_j(A) \leq l\}| - \left(\sum\limits_{j \in \mathcal{F}_k} (min\,\{l, g_j(A) - 1\} - k)\right)$, where $f_i(A) = \min\{j : 1 \leq j \leq i, \text{with } a_{ij} \neq 0\}$, $\mathcal{F}_k = \{j : f_j(A) = k\}$ is the set of columns in line k, and $g_i(A) = min\,\{j : f_i(A) < j, a_{ij} \neq 0\}$ is the second non-null coefficient in column j, if it exists, or $g_j(A) = n+1$ otherwise [5]. This formula denotes the change in the profile associated with a series of adjacent exchanges, i.e., interchange row k with $k+1$, row $k+1$ with $k+2$, ..., row $l-1$ with l, and perform the symmetric interchange of columns. The elements of the frontier set $\mathcal{F}_k$ are adjacent vertices to k and on the boundary of the envelope. Based on this formula, Hager [5] proposed the adjacent exchange methods. From the bottom of the matrix, these methods identify a set of lines whose adjacent down interchanges provide the smaller increase in profile and performs the corresponding interchanges if this increase is negative. For each value of k ranging from $n-1$ to 1, this algorithm chooses a value L, with $k < L \leq n$, which provides the smaller result in $D_{k:l}$, calculating $D_{k:L} < 0$. After selecting L, if it exists, this algorithm performs the adjacent down interchanges between lines k and L. Then, the procedure performs similar symmetric up adjacent interchanges.

3 Description of the tests

Sloan's [12] and NSloan [7] heuristics have important parameters that may influence the results. We use the parameters recommended by the heuristics' authors, i.e., $w_1 = 1$ and $w_2 = 2$ corresponding respectively to global and local criteria related to the distance of each vertex from the target end vertex and related to the degree of each vertex within Sloan's [12] heuristic; and $w_1 = 2$ and $w_2 = 1$ within the NSloan heuristic [7]. The other heuristics do not have parameters that affect the results [4].

The results obtained using the KP–band heuristic [6] are highly dependent on the choice of a starting vertex of the labeling. Koohestani and Poli [6] did not describe the pseudo-peripheral vertex finder employed. Then, we applied the George-Liu algorithm (see [2] and references therein) to find a pseudo-peripheral vertex to be the starting vertex of this reordering algorithm.

We used the C++ programming language to write the heuristics. In a previous publication (see [4] and references therein), we describe the testing and calibration performed to compare our implementations with the original heuristics. We used a data structure based on the Compress Row Storage, Compress Column Storage, and Skyline Storage Scheme data structures to implement the zero–fill incomplete Cholesky–preconditioned conjugate gradient method [4].

The workstations used in the execution of the simulations with instances of the SuiteSparse matrix collection [1] contained an Intel® Core^{TM} i7-4770 (CPU 3.40GHz, 8MB Cache, 8GB of main memory DDR3 1.333GHz) (Intel; Santa Clara, CA, United States). In this machine, we use an Ubuntu 14.04.4 LTS 64-bit operating system with Linux kernel-version 4.2.0-36-generic.

Our procedure employs a precision of 10^{-16} using double-precision floating-point arithmetic to the preconditioned conjugate gradient method. If the procedure does not achieve this precision, the preconditioned conjugate gradient method stops when the number of iterations is the size of the instance.

4 Results and analysis

This section presents and analyzes the results obtained in simulations using the ICCG method computed after executing heuristics for bandwidth and profile reductions and Hager's adjacent exchange methods [5]. Specifically, this section shows the results obtained from the solutions of linear systems (ranging from 60,000 to 141,347 unknowns) contained in the SuiteSparse matrix collection [1].

Table 1 shows the instance's name, its size ($|V|$) and number of non-null coefficients ($|E|$), the name of the heuristic for bandwidth and profile reductions, the results with respect to bandwidth and profile reductions, the results of the algorithms in relation to execution times, in seconds (s), including times obtained by Hager's adjacent exchange [5] and the ICCG methods. Additionally, the first line of each instance presented in this table shows the results for linear systems solved using the ICCG method without applying a reordering algorithm. These lines are indicated by the symbol "—" in this table. With this result, one can verify the speed-up (or speed-down) (i.e., the time of the ICCG method without applying a reordering algorithm divided by the time of the ICCG method executed in conjunction with reordering algorithms) of the ICCG method provided when applying reordering algorithms. The last column of this table shows these speed-ups/downs. Numbers in boldface are the best results.

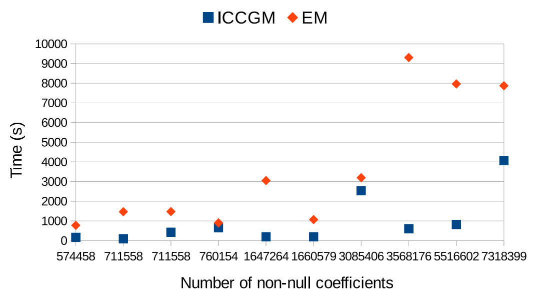
Table 1 also shows the results obtained by the heuristic for bandwidth and profile reductions that reached the best results in a specific instance [4]. The RCM–GL [2], Sloan [12], MPG [9], NSloan [7], KP–band [6], or RBFS-GL [4] heuristics were applied to these instances. Then, our implementation employs Hager's adjacent exchange methods [5] in conjunction with this heuristic for bandwidth and profile reductions. Our executions perform only one iteration of Hager's (down and up) adjacent exchange methods [5] with the objective of reaching low execution times.

Table 1 shows that Hager's adjacent exchange methods [5] are effective in reducing the profile of matrices (see column Profile in this table). Except when applied to the *qa8fm* instance, Hager's adjacent exchange methods [5] reduced the profile of matrices when computed after a heuristic for bandwidth and profile reductions. Even applying only one iteration (down and up), Hager's exchange methods reduced on average 10% the profile of the matrices resulting from the heuristics for bandwidth and profile reductions evaluated. Thereby, more iterations might provide further profile reductions of the instances at higher computing times. In general, the use of a heuristic for bandwidth and profile reductions without using Hager's adjacent exchange methods [5] reduced executions times of the ICCG method (see column ICCGM in Table 1). In simulations with four instances (*thermal1*, *2cubes_sphere*, *thermomech_TC*, and *boneS01*), the execution times of the ICCG method was even lower when applying Hager's adjacent exchange methods [5]. In the case of solving the linear system multiple times, the iterative process would amortize the processing times of Hager's adjacent exchange method. However, Figure 1(a) shows that the execution times of Hager's adjacent exchange methods [5] are in general much higher than the execution times of the ICCG method. The objective is to minimize the total computing time of the simulation, including the execution time of the heuristic for bandwidth and profile reductions and Hager's adjacent exchange methods, at least when

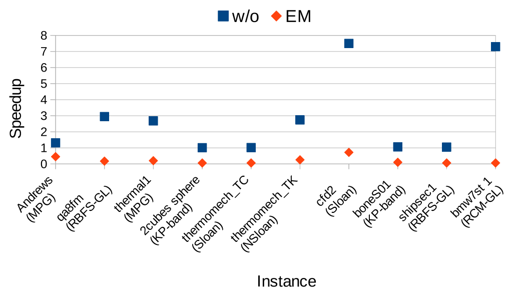
Table 1: Results from the solution of 10 linear systems contained in the SuiteSparse matrix collection [1] using the ICCG method and vertices labeled by heuristics for bandwidth and profile reductions and Hager's adjacent exchange methods (EM) [5].

Instance	Algorithm	β	Profile	heur.	EM	IC(0)	CGM	ICCGM	Iter.	Sp.
Andrews	—	59925	1501971521	—	—	**28**	630.29	658	60000	—
$\|V\| = 60000$	MPG	**51374**	113418624	0.9	—	33	**467.14**	**500**	60000	**1.31**
$\|E\| = 760154$	MPG+EM	53344	**103809612**	1.1	901	34	523.00	557	60000	0.45
qa8fm	–	**1048**	**64862523**	—	—	196	0.46	196	21	—
$\|V\| = 66127$	RBFS-GL	2016	76755688	0.1	—	**66**	**0.14**	**67**	18	**2.95**
$\|E\| = 1660579$	RBFS-GL+EM	2778	77260043	0.1	1068	70	0.20	71	18	0.17
thermal1	–	80916	175625317	—	—	65	101.00	166	8643	—
$\|V\| = 82654$	MPG	1448	10403099	0.2	—	**55**	6.60	62	**577**	**2.68**
$\|E\| = 574458$	MPG+EM	**1310**	**10081742**	0.2	777	49	**5.30**	**54**	616	0.20
2cubes_sphere	—	100407	483241271	—	—	195	0.30	196	13	—
$\|V\| = 101492$	KP–band	**4927**	353914520	0.3	—	192	**0.20**	193	13	**1.01**
$\|E\| = 1647264$	KP–band+EM	11546	**217022477**	0.3	3049	**151**	0.23	**151**	**12**	0.06
thermomech_TC	—	102138	2667823445	—	—	98	0.40	99	20	—
$\|V\| = 102158$	Sloan	**1032**	16021130	0.3	—	96	0.30	96	16	**1.02**
$\|E\| = 711558$	Sloan+EM	1725	**15258586**	0.3	1468	**89**	**0.19**	**89**	16	0.06
thermomech_TK	—	102138	2667823445	—	—	**75**	350.00	425	23417	—
$\|V\| = 102158$	NSloan	2799	15493278	1.1	—	80	**73.60**	**154**	**7911**	**2.74**
$\|E\| = 711558$	NSloan+EM	**2785**	**15211001**	1.2	1475	88	98.70	187	8606	0.26
cfd2	—	4501	156107322	—	—	**336**	2202.00	2538	123440	—
$\|V\| = 123440$	Sloan	1176	139135583	0.9	—	337	**0.04**	**337**	1	**7.50**
$\|E\| = 3085406$	Sloan+EM	**1015**	**134876558**	0.9	3197	341	0.05	341	1	0.72
boneS01	—	3722	**331330356**	—	—	759	59.30	818	1309	—
$\|V\| = 127224$	KP–band	**8077**	588599769	1.0	—	741	31.00	772	684	**1.06**
$\|E\| = 5516602$	KP–band+EM	8912	410207049	0.6	7963	**609**	**16.00**	**625**	**598**	0.10
shipsec1	—	**5237**	**431771001**	—	—	583	23.94	607	960	—
$\|V\| = 140874$	RBFS-GL	5879	451937073	0.2	—	**576**	**0.05**	**576**	1	**1.05**
$\|E\| = 3568176$	RBFS-GL+EM	5963	449908701	0.2	9303	791	0.07	791	1	0.06
bmw7st_1	—	121856	1371669955	—	—	**534**	3526.70	4061	141347	—
$\|V\| = 141347$	RCM–GL	**3652**	289187078	0.5	—	556	**0.10**	**556**	1	**7.30**
$\|E\| = 7318399$	RCM–GL+EM	12454	**272733259**	0.6	7871	765	0.08	765	1	0.06

the process solves only a single linear system [4]. Our implementation of Hager's exchange method was on average almost four times slower than the ICCG method. For instance, when applied to the *2cubes_sphere* instance, where Hager's exchange methods delivered reasonable results in reducing the execution time of the ICCG method (i.e., 151s against 193s yielded by executing the KP–band heuristic [6] alone), the computing time of Hager's exchange methods was higher (3049s) than executing the ICCG method alone (196s). It would take $\lceil \frac{3049s+151s}{193s} \rceil = 22$ executions of the ICCG method in conjunction with the KP–band heuristic to amortize the time of Hager's exchange methods. Table 1 and Figure 1(b) show that the use of heuristics for bandwidth and profile reductions obtained better results to reduce execution times of the ICCG method than using them along with Hager's adjacent exchange methods [5] (see column Speedup in Table 1).



(a)



(b)

Figure 1: Results of the ICCG method along with heuristics for bandwidth and profile reductions and Hager's adjacent exchange methods [5] applied to 10 symmetric instances contained in the SuiteSparse matrix collection [1]: a) execution times obtained by the ICCG and Hager's adjacent exchange methods (EM); b) speed-up/down of the ICCG method obtained when applying heuristics for bandwidth and profile reductions with (EM) and without (w/o) the use of Hager's adjacent exchange methods.

5 Conclusions

This paper shows simulations using 10 symmetric instances contained in the SuiteSparse matrix collection [1] with the objective of reducing execution times of the zero–fill incomplete Cholesky–preconditioned conjugate gradient method. A previous publication [4] exhibited the heuristics for bandwidth and profile reductions that achieved the best speedup when applied to each of these instances. Our simulations applied the selected heuristic to the specific instance to verify the speedup of the ICCG method provided by the heuristic. Afterward, a simulation employed the same heuristic for bandwidth and profile reductions along with Hager's adjacent exchange [5] and ICCG methods to the same instance. This simulation is performed to verify whether the speedup of the ICCG method could be improved. However, even one iteration of Hager's (down and up) adjacent exchange methods [5] is more time consuming than the ICCG method. The results obtained in this computational experiment show that Hager's adjacent exchange methods [5], although capable of reducing the profile of the instances, are not useful when used to reduce computing times of the ICCG method.

It is well-known that the ICCG method executes with a lower number of iterations and computing times when the condition number of the matrix is small (or the instance presents a better cluster of eigenvalues). Spatial locality significantly improves the computing time of the ICCG method (see [4] and references therein). In this context, with the simulations carried out in this paper, we suppose that the use of an expensive local search employed to reduce bandwidth or profile of matrices does not favor the computing time of a linear system solver because of the high running times of the procedure. Specifically, this paper reveals that the computing times to perform Hager's adjacent exchange methods [5] increase the total running time of a simulation when applying an iterative system solver.

Reid and Scott [11] showed a low-cost Fortran implementation of Hager's exchange methods. We plan to evaluate this implementation with the same objective of reducing the execution times of the ICCG method in future studies.

Also as a continuation of this paper, we intend to implement and evaluate other preconditioners in conjunction with the conjugate gradient method. Additionally, we plan to implement parallel approaches to the above methods.

References

[1] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, 38(1):1:1–1:25, December 2011.

[2] A. George and J. W. Liu. *Computer solution of large sparse positive definite systems.* Prentice-Hall, Englewood Cliffs, 1981.

[3] G. H. Golub and C. F. van Loan. *Matrix computations.* The Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[4] 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*, 2017, DOI: 10.1007/s40314-017-0490-5.

[5] W. W. Hager. Minimizing the profile for a symmetric matrix. *SIAM Journal on Scientific Computing*, 23(5):1799–1816, 2002.

[6] 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*, pages 93–106, London, UK, 2011. Springer London.

[7] G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT Numerical Mathematics*, 37(3):559–590, 1997.

[8] Y. X. Lin and J. J. Yuan. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica*, 10(1):107–122, 1994.

[9] S. R. P. Medeiros, P. M. Pimenta, and P. Goldenberg. Algorithm for profile and wavefront reduction of sparse matrices with a symmetric structure. *Engineering computations*, 10(3):257–266, 1993.

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

[11] J. K. Reid and J. A. Scott. Implementing Hager's exchange methods for matrix profile reduction. *ACM Trans. Math. Softw.*, 28(4):377–391, December 2002.

[12] S. W. Sloan. A Fortran program for profile and wavefront reduction. *International Journal for Numerical Methods in Engineering*, 28(11):2651–2679, 1989.