# On sparse matrix orderings in interior point methods

Csaba Mészáros

Group of Operations Research and Decision Systems,
Hungarian Academy of Sciences
Address: H-1518, Budapest, P.O. Box 63.
e-mail: meszaros@sztaki.hu

**Abstract**

The major computational task of most interior point implementations is solving systems of equations with symmetric coefficient matrix by direct factorization methods, therefore, the performance of Cholesky-like factorizations is a critical issue. In case of sparse and large problems the efficiency of the factorizations is closely related to the exploitation of the nonzero structure of the problem. A number of techniques were developed for fill–reducing sparse matrix orderings which make Cholesky factorizations more efficient by reducing the necessary floating point computations. We present a variant of the nested dissection algorithm incorporating special techniques that are beneficial for graph partitioning problems arising in the ordering step of interior point implementations. We illustrate the behavior of our algorithm and provide numerical results and comparisons with other sparse matrix ordering algorithms.

**Keyword(s):** Interior point methods; Fill–reducing orderings; Sparsity; Large-scale problems

## 1 Introduction

In the past 20 years interior point methods (IPMs) proved their efficiency in practice, especially when solving large-scale problems. One key point of their practical success is the ability to reduce the computational work by exploitation

of structures presented in the optimization problems. In practice optimization problems tend to have large dimensions but relatively small number of nonzero elements in the constraint matrix. Nowadays we can solve problems with millions of variables and constraints if such sparsity structures are exploited efficiently. Exploitation of sparsity in interior point methods is done on different levels. In [20] the author described how disadvantageous cases for the traditional Cholesky factorization can be identified and perform an quasidefinite factorization instead. The approach is crucial when solving problems from some special application areas, such as when solving two-stage stochastic optimization [17]. The other level of exploitation of sparsity is the symbolic ordering step of interior point methods [1, 23].

In the paper we describe an implementation of a nested dissection ordering algorithm, which belongs to the most popular and powerful ordering methods [8, 10, 9, 14] nowadays for large-scale problems. The described ordering scheme is incorporated to our interior point implementation, called BPMPD [19]. In Section 2 we outline the basics of an implementation of the primal-dual interior point method and describe the most popular sparse matrix ordering techniques. In section 3 our implementation of the nested dissection ordering algorithm is discussed. Section 4 contains numerical experiments and a comparison with the public available Metis sparse matrix ordering package [14]. Section 5 summarizes our findings.

# 2 Sparsity in interior point methods

In this paper the following linear programming problem is considered

$$\begin{aligned} \min \quad & c^T x, \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{1}$$

where $A$ is an $m \times n$ matrix, $c$ and $x$ are $n$ vectors, and $b$ is an $m$ vector. Without loss of generality we may assume that $rank(A) = m$. Several papers, including [16, 1] give a survey of the developments achieved with the different algorithmic variants of interior point methods of linear programming. In all these IPMs the most costly operation is to determine the search direction. For most problems the most efficient way is to solve the so–called *system of normal equations*

$$AD^2A^T\Delta y = h, \tag{2}$$

where $D$ is an $n \times n$ diagonal matrix with positive diagonal values, and $h$ is an $m$ vector. In every iteration the diagonal matrix $D$ is updated, and system (2) is to be solved several times with different right–hand sides to compute the next iterate. The big advantage of the normal equation approach is that it deals with a system of linear equations with a positive definite coefficient matrix. As such, the Cholesky factorization can be used with a great success. Since theoretically all pivots are positive, one can pay full attention to the sparsity issues in the symbolic factorization phase. The symbolic factorization is made once in advance, i.e., the pivot agenda is the same during iterations. The pivot order of the Cholesky factorization has a very strong influence on the number of fill–in and, therefore, on the storage requirement and the speed of factorizations. In the symbolic phase we try to find a good row permutation of $A$ that minimizes the number of nonzeros in the Cholesky factors of $AA^T$. Since finding the "optimal" permutation is an NP complete problem [24], heuristics are used in practice.

Graph theory was identified as a convenient framework for investigation of sparse matrix computations. After Parter applied undirected graphs to model symmetric Gaussian elimination [21], most of the ordering algorithms in IPMs use this methodology. The *minimum degree* ordering [6] was one of the most important and popular methods, it is still widely used in interior point implementations. The implementation of this ordering algorithm is a well–developed area, the modern implementations are extremely powerful in the sense of speed and storage requirements. The most important aspect of this ordering is the connection with graph theory, and the technique of the implicit representation of the elimination, which makes compact storage and efficient implementation possible. For further details on the above, the reader is referred to the summary in [7].

Some authors considered an other local heuristic approach, the *minimum local fill* [5] ordering. It was included in some of the implementations as an alternative to the minimum degree ordering. This method computes the number of fills produced by each pivot candidate, and chooses that one where this number is minimal for the next pivot. Generally, the minimum local fill algorithm produces sparser factorization but at higher initial cost to obtain the ordering, because the analysis in the pivot search process is very expensive [16]. To avoid the expense of the computation of the true fill–in, approximation schemes were proposed and incorporated to the minimum degree algorithm to improve its quality [18, 22].

# 3 The nested dissection ordering algorithm

Whereas in implementations of interior point methods the minimum degree was the standard default ordering algorithm at the beginning of the practical use of IPMs, at the same time methods based on bisections and multisections were extensively studied in other application areas such in finite element methods (FEMs). The ordering methods based on graph partitioning became standard in IPMs in the past 10 years only. An important property of sparse matrices arising in IPMs is that they are usually more dense than those derived in other applications. Usually it can be observed that the number of nonzeros in $AA^T$ is of the same order the number of nonzeros in the constraint matrix and therefore the importance of sparse matrix ordering increases with the problem dimensions.

The methods based on graph partitioning have a more global view on the graphs. In this algorithm we try to divide the original graph to smaller parts by excluding a possibly small set of separator vertices. There are several variants of this method, such as the multilevel orderings, the domain decomposition and spectral bisection methods.

The aim of the ordering schemes based on graph partitioning is that a decomposition of $G(V, E)$ is computed as

$$G = P_1 \cup P_2 \cup ... \cup P_k \cup S$$

such that for any $i \neq j$ pairs

$$\text{if } i \in P_i \text{ and } j \in P_j \text{ then } (i, j) \notin V.$$

The subgraphs corresponding $P_i$ are ordered independently for $i = 1, ..., k$ and the vertices in $S$ are placed at the end of the elimination sequence.

For the interior point code called BPMPD we implemented a multilevel nested dissection scheme [13] which computes bisections $G = W \cup B \cup S$ recursively, where $S$ is the separator and $W$ and $B$ are the two partitions. The goal of bisection is to compute a separator set that is small and splits the graph into well–balanced partitions. Our multilevel bisection scheme has the following steps:

- Preprocessing.
- Coarsening.
- Do

  - Initial bisection.
  - Partial uncoarsening and refining.
  - Separator evaluation.

- Until enough bisections tried
- Reload the best partitioning found.
- Uncoarsening and refining.
- Local perturbations and refinements.

We selected a set of test problems from various applications to represent problems of different structures for our experiments. The characteristics of these problems are summarized in Table 1. Figures include the name of the problems, the number of constraints, representing the vertices of the graph, the number of nonzero entries in the matrix $AA^T$, representing the edges of the graph, the average degree of the vertices and the diameter of the graph. We define the diameter in the usual way, i.e. as the length of the longest shortest path in the graph.

Table 1: Test problems

| Name | $m$ | $NZ(AA^T)$ | avg. degree | diameter |
|---|---|---|---|---|
| 25fv47 | 754 | 21442 | 28.4 | 7 |
| dfl001 | 5471 | 74402 | 13.6 | 6 |
| epa | 232125 | 8872022 | 38.2 | 11 |
| fome13 | 43768 | 396016 | 9.0 | 6 |
| gasbaucp | 155841 | 4054680 | 26.0 | 13 |
| dek–10 | 333187 | 14840604 | 44.5 | 10 |
| munn–1 | 162882 | 3243188 | 20.0 | 12 |
| munn–2 | 52297 | 1051518 | 20.1 | 9 |
| nug20 | 15240 | 1201600 | 78.8 | 3 |
| nsct2 | 7797 | 1818658 | 233.3 | 6 |
| pds–70 | 111874 | 990914 | 8.9 | 21 |
| pds–100 | 152289 | 1272220 | 8.4 | 22 |
| rail4284 | 4176 | 2133370 | 510.9 | 4 |
| smdvds | 61245 | 1253070 | 20.5 | 12 |
| spal–004 | 10203 | 51889356 | 5085.7 | 2 |

Naturally, graph bisections are easier when the graph is less connected and has a large diameter, but, Table 1 indicates that the graph representation of sparse symmetric matrices arising in interior point methods are rather too dense compared to graphs arising in FEMs, calling for techniques that take this property into account.

The preprocessing step of our bisection algorithm detects and removes vertices with high degree and compresses those that share the same set of neighbors. In the bisection phase the graph is compressed by combining vertices and weights are generated on vertices and edges, until the number of vertices decrease to about 100. Our goal in the coarsening phase is to keep the coarse graphs sparse as possible and prevent the rapid decrease of the diameter of the coarse graphs. To achieve this, we implemented a matching scheme that maximizes the common neighbors and deliver the largest possible reduction of edges at each step. We stop coarsening if the diameter of the graph becomes less or equal 3, or the number of vertices decrease bellow 200.

In the initial bisection we implement a "greedy growing" algorithm, starting from two vertices defined by the diameter of the graph as $B$ and $W$ while $S$ contains all other vertices. At each step of the greedy algorithm we select a vertex from $S$ to move to $B$ or $W$ which has the smallest number of neighbors in $S$. Different initial bisections are computed from different starting partitions and the one resulting the best partitioning is kept.

After an initial bisection is computed, the coarse graph is projected back to the finer levels and the separator is refined with two techniques. The first one is the local refinement technique of Kernigham-Lin, called primitive moves [15] that modifies the partitioning by moving vertices with the best score, where the score is defined as the reduction of the separator set. Our other implemented method is the refinement technique based on bipartite matching [2] that decreases the number of vertices in the separator or improves the balancing. We would like to note that Ashcraft and Liu extended this refinement technique by using "wider" separators and network flows [3] but our experiments showed that this extension is less effective on graphs arising in interior point methods. We found that the described technique tends to generate imbalanced partitioning due to the dense edge structure of the graph.

We evaluate the generated partitioning at the 3rd coarse level. For the evaluation we use the utility function defined by Hendrickson [11] as

$$f(S, W, B) = \frac{|S|}{|W| * |B|}.$$

We try 5 initial bisections and select the best in this pass. The best bisection is

then projected back to the original graph while the Kernigham-Lin followed by the bipartite macthing refinement techniques are applied at each uncoarsening step. Similarly to the technique described in [12], on the final partitioning the following refinement procedure is applied:

- Do
  - Generate edge separator from vertex separator.
  - Refine edge separator.
  - Generate vertex separator from edge separator.
  - Refine vertex separator.
- While the partitioning improves.

The process is started on the whole graph and applied on the resulting partitioning $S$ and $W$ recursively, while the separator is removed from the graph. This recursive process stops if number of vertices in the subgraphs decrease bellow 200, or the process generates bisections with large number of vertices in the separator. In this latter case the partitioning is rejected.

After the process the partitions $B$ and $W$ are ordered independently by a minimum degree algorithm with approximate fill–in utility function [22] and the separator vertices are unified as a multisector [4] and placed to the end of the ordering sequence.

# 4 Numerical experiments

In our numerical experiments we compare our nested dissection scheme with the freely available graph partitioning software package, Metis 4.0 [14] and the standard minimum degree algorithm with approximate fill-in utility function. From the Metis package the METIS_NodeND procedure was applied with the following option vector:

$$[1, 3, 1, 2, 0, 1, 100, 2, \max(900, \min(100, m/40))].$$

We collected the results in Table 2. Figures given include the number of nonzero elements in the Cholesky factors $L$ in thousands, the number of floating point operations for computing the Cholesky decomposition in millions achieved by the minimum degree ordering as references, and the relative values of those

Table 2: Comparison of ordering algorithms

| Problem | Min.Deg. | | $nz(L)$ | | flops | |
|---|---|---|---|---|---|---|
| name | $nz(L) \times 10^3$ | $flops \times 10^6$ | Metis | Bpmpd | Metis | Bpmpd |
| 25fv47 | 30 | 1 | 96 % | 92 % | 92 % | 78 % |
| dfl001 | 1355 | 517 | 84 % | 85 % | 62 % | 63 % |
| epa | 11663 | 967 | 119 % | 101 % | 364 % | 102 % |
| fome13 | 11015 | 4267 | 85 % | 79 % | 64 % | 57 % |
| gasbaucp | 29891 | 48478 | 86 % | 83 % | 63 % | 58 % |
| dek–10 | 21570 | 2828 | 116 % | 104 % | 287 % | 112 % |
| munn–1 | 307603 | 950974 | 32 % | 24 % | 12 % | 4 % |
| munn–2 | 90598 | 212859 | 36 % | 16 % | 24 % | 5 % |
| nug20 | 64295 | 232129 | 75 % | 72 % | 58 % | 52 % |
| nsct2 | 4945 | 3327 | 80 % | 81 % | 57 % | 59 % |
| pds–70 | 76945 | 202324 | 39 % | 32 % | 12 % | 5 % |
| pds–100 | 114587 | 354107 | 33 % | 27 % | 15 % | 6 % |
| rail4284 | 5588 | 5811 | 102 % | 99 % | 101 % | 99 % |
| smdvs | 195305 | 585640 | 34 % | 24 % | 17 % | 6 % |
| spal–004 | 45127 | 133256 | 112 % | 98 % | 109 % | 99 % |

achieved by Metis and our implementation. We selected test problems to cover different problem types from different application areas.

The experiments show that on some cases, typically on problems which produce $AA^T$ matrix with significant density, the graph partitioning algorithm is not able to improve the quality of the minimum degree and sometimes decreases the quality of the ordering. Let us note that our criteria based on the monitoring the diameter of the graph during the coarsening phase is able to identify such cases and stop the nested dissection algorithm quickly.

We can observe that on several cases a significant reduction in the generated nonzero elements can be achieved by the ordering methods based on graph partitioning. This results in decrease in the necessary floating point operations needed for the iterations of the interior point algorithm, giving a proportional speedup in the execution time. These problems often come from assignment or scheduling applications (munn-1, munn-2, pds-100) in which an underlying network is present in the constraints, providing an advantageous structure which is exploited by the graph partitioning methods.

We can derive that our implementation gives usually better orderings than Metis, but we would like to note that while Metis is a general–purpose graph

partitioning package in which we tuned its behavior by its options only. The ordering algorithm in BPMPD was specifically developed for ordering sparse symmetric matrices arising in interior point methods, exploiting their special properties outlined in Section 3.

Since the minimum degree algorithm is a very fast procedure, interior point implementations always perform it beside the nested dissection ordering and select that result which gives the better quality. This requires little computational effort and guarantees that the interior point algorithm will use an ordering which is always at least as good as the result of the minimum degree.

# 5 Concluding remarks

In the paper we described our ordering scheme for interior point methods which is based on the multilevel nested dissection graph partitioning scheme. We implemented techniques that take the special properties of the sparse matrices arising in IPMs into consideration and our experiments showed that they provide benefits when comparing with general–purpose graph partitioning codes. Our experiments showed that the developed method successfully exploits the underlying advantageous structures in the $AA^T$ matrix.

# 6 Acknowledgements

This work was supported in part by Hungarian Research Fund OTKA K-77420 and K-60480.

# References

[1] E.D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programs. In T. Terlaky, editor, *Interior Point Methods of Mathematical Programming*, pages 189–252. Kluwer Academic Publishers, 1996.

[2] C. Ashcraft and J. W. H. Liu. Using domain decomposition to find graph bisectors. *BIT Numerical Mathematics*, 37(3):506–534,, 1997.

[3] C. Ashcraft and J. W. H. Liu. Applications of the dulmage-mendelsohn decomposition and network ow to graph bisection improvement. *SIAM Journal on Matrix Analysis and Applications*, 19:325–354, 1998.

[4] C. Ashcraft and J. W. H. Liu. Robust ordering of sparse matrices using multisection. *SIAM Journal on Matrix Analysis and Applications*, 19(3):816–832, 1998.

[5] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct methods for sparse matrices*. Oxford University Press, New York, 1986.

[6] A. George and J.W.H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[7] A. George and J.W.H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Reviews*, 31:1–19, 1989.

[8] J. A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.

[9] A. Gupta. Watson graph partitioning package. Technical Report RC 20453, IBM T. J. Watson Research Center, 1996.

[10] B. Hendrickson and R. Leland. The chaco user's guide, version 2.0. Technical report, Sandia National Laboratories, 1995.

[11] B. Hendrickson and E. Rothberg. Improving the run time and quality of nested dissection ordering. *SIAM Journal on Scientific Computing*, 20(2):468–489, 1998.

[12] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. Technical report, Supercomputing '95, Proceedings of the 1995 ACM/IEEE conference on Supercomputing, 1995.

[13] G. Karypis and V. Kumar. Metis a software package for partitioning unstructured graphs, partitioning meshes, and computing fill–reducing orderings of sparse matrices version 3.0. Technical report, University of Minnesota, 1998.

[14] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[15] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. Technical Report 49, Bell Systems Technical Journal, 1970.

[16] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA J. on Computing*, 6(1):1–15, 1994.

[17] C. Mészáros. The augmented system variant of IPMs in two–stage stochastic linear programming. *European J. on Operational Research*, 101(2):317–327, 1997.

[18] C. Mészáros. Ordering heuristics in interior point LP methods. In F. Gianessi, S. Komlósi, and T. Rapcsák, editors, *New Trends in Mathematical Programming*, pages 203–221. Kluwer Academic Publishers, 1998.

[19] C. Mészáros. The BPMPD interior-point solver for convex quadratic problems. *Optimization Methods and Software*, 11&12:431–449, 1999.

[20] C. Mészáros. Detecting dense columns in linear programs for interior point methods. *Computational Optimization and Application*, 36(2–3):309–320, 2007.

[21] S.V. Parter. The use of linear graphs in gaussian elimination. *SIAM Rev.*, 3:191–130, 1961.

[22] E. Rothberg. Ordering sparse matrices using approximate minimum local fill. Technical report, Silicon Graphics Inc., Mountain View, CA 94043, 1996.

[23] E. Rothberg and B. Hendrickson. Sparse matrix ordering methods for interior point linear programming. *INFORMS J. on Computing*, 10(1):107–113, 1998.

[24] M. Yannakakis. Computing the minimum fill–in is NP–complete. *SIAM J. Algebraic Discrete Methods*, 2:77–79, 1981.