ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

ALGORITHM FOR FAST SIMULATIONS OF SPACE–TIME FINITE ELEMENT METHOD

Marcin Skotniczny[1], **Anna Paszynska**[2], **and Maciej Paszynski**[1]

[1]AGH University of Science and Technology,
al. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: {mskotn, maciej.paszynski}@agh.edu.pl

[2]Jagiellonian University
ul. Lojasiewicza 11, Krakw, Poland
anna.paszynska@uj.edu.pl

Keywords: Finite element method, Time–space formulations, h–adaptivity, Multi–frontal solver, Orderings

Abstract. *In this paper we consider element partition trees for multi–frontal solver algorithms utilized in space–time finite element method. The element partition trees are utilized for generating the ordering for the multi–frontal solver algorithm. In particular, we consider three or four dimensional finite element method grids where two or three dimensions represent space and one additional dimension represents time. Additionally, we consider computational grids resulting from h–adaptive algorithms, namely grids refined towards point, edge, face or hyper-face singularity. We perform numerical experiments and compare our method with alternative state of the art ordering algorithms available through MUMPS interface.*

1 INTRODUCTION

We propose a heuristic algorithm generating element partition trees for 3D or 4D grids refined towards singularities. This work is an extension of the algorithm proposed for two dimensional problems [1]. When we solve 3D time dependent problem over a large grid, we need to utilize a sequence of grids, that can be refined only in spatial domain, and we cannot take advantage of possible time adaptivity. In this paper we consider an alternative approach that allows for time adaptivity, i.e. allows to use different time steps over different parts of the 3D mesh, at the same time allowing the time steps to change over time. As example, we investigate four dimensional grids refined towards point, edge, face and hyper–face singularities. We generate element partition trees obtained by bisections weighted by element size and estimate the resulting number of floating point operations of the multi–frontal solver algorithm. Our analysis shows that for both three– and four–dimensional point and edge singularity we obtain the linear computational cost $O(N)$, for three– and four–dimensional face singularity we obtain computation cost $O(N^{1.5})$ and for four–dimensional hyper–face singularity we obtain cost of $O(N^2)$ where N is the number of nodes. Values for edge, face and hyper–face singularities correspond to one, two and three dimensional uniform grids respectively. Our method is dedicated to stable time-space formulations using classical finite element method [2] or the DPG method [3].

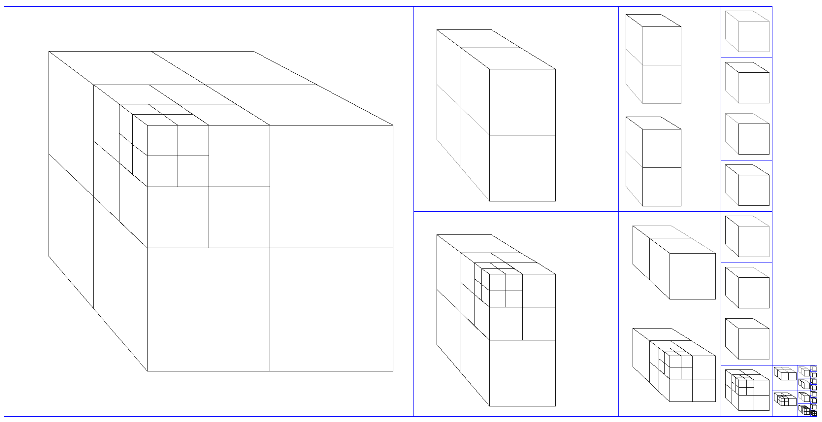


Figure 1: 3D mesh with point singularity and its element partition tree.

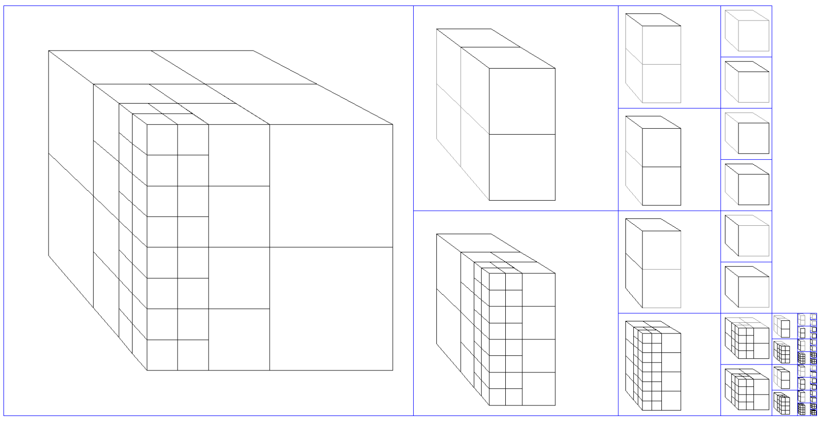


Figure 2: 3D mesh with edge singularity and its element partition tree.

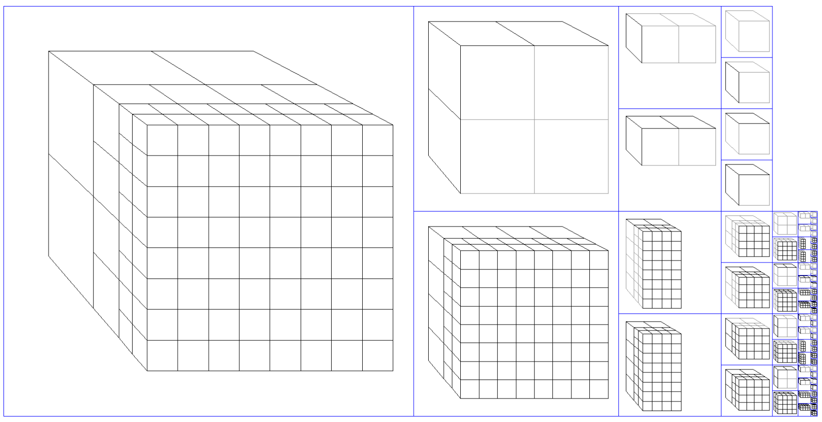


Figure 3: 3D mesh with face singularity and its element partition tree.

2 BISECTIONS WEIGHTED BY ELEMENT SIZE ALGORITHM

In this section we introduce top down algorithm for the construction of element partition trees, called bisections weighted by element size algorithm. The algorithm is based on the multilevel recursive bisection. We make the following asumptions:

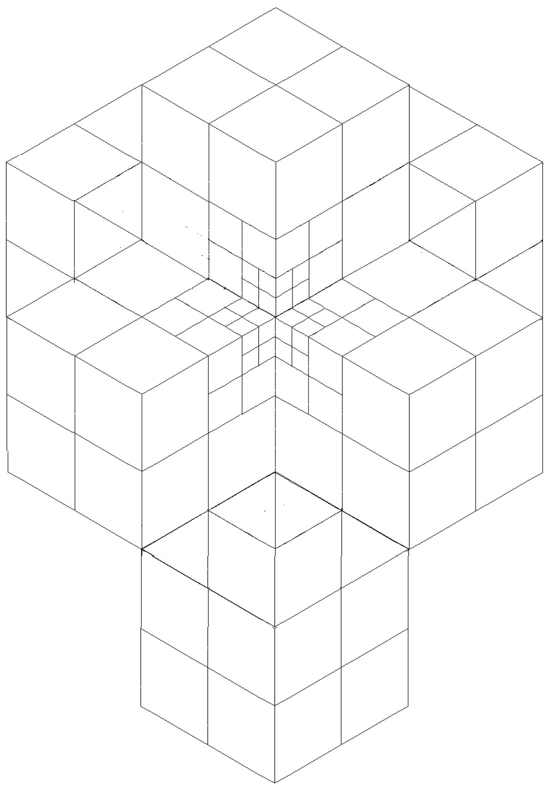
- The input is the graph in which vertices represent mesh elements and edges represent spatial adjacency relation between them.
- We assign weights to vertices equal to the volume of element that each of them represents.
- The volume is proxy for refinement level and is defined as $(\frac{1}{2})^{3R}$ in 3D and $(\frac{1}{2})^{4R}$ in 4D, where R is the refinement level of a mesh element. This way, the total volume over all elements should equal 1.

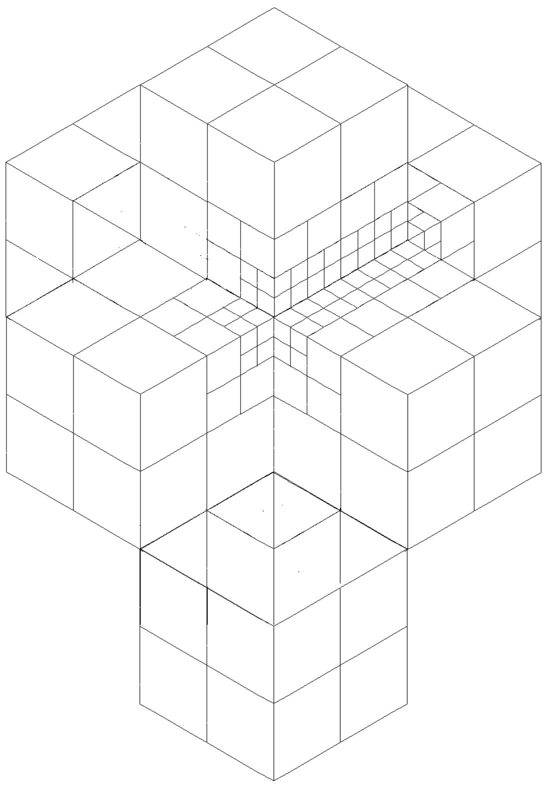
The input graph is recursively partitioned into 2 equally–weighted parts by using graph partitioning algorithm. The goal of the graph partitioning problem is to compute a balanced partitioning, such that:

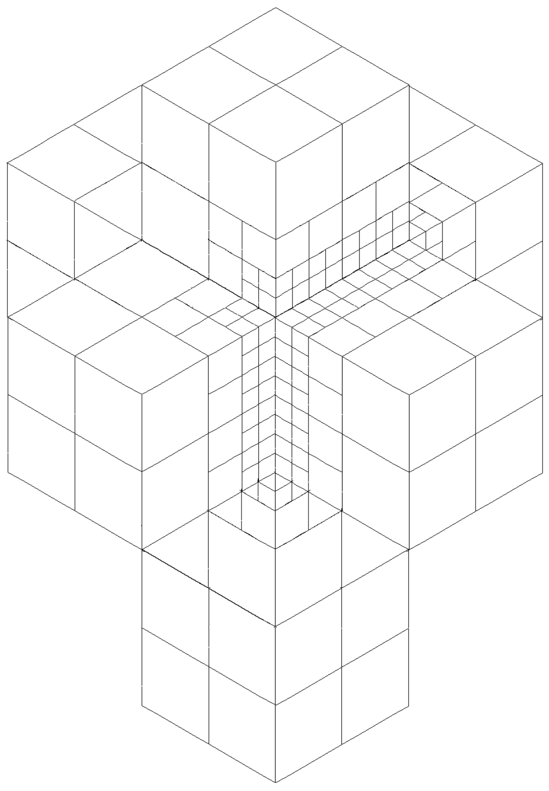
- the number of edges (or the sum of their weights) over the interface is minimal, and
- the number of vertices in each part of the graph is the same (or the sum of weights of vertices in each part of the graph).

To solve the problem, we use METIS_WPartGraphRecursive function to find a balanced partition of a graph where weights on vertices are equal to the volumes of represented mesh elements and weights on the edges equal to 1.

We illustrate the obtained partitions for the recursive partition over 3D meshes with point, edge and face singularities in Figures 1–3. We also show the boundary of corresponding 4D meshes with singularities in Figure 4. The recursive partitions generate the ordering in the following way: we start traversing the partition tree from leaves up to the root in parallel. At leaves we eliminate interiors of elements and on the internal tree nodes we eliminate mesh nodes on faces shared between two partitioned blocks of the mesh. We continue this process recursively up to the root of the partition tree.


(a) Point singularity.


(b) Edge singularity.


(c) Face singularity.

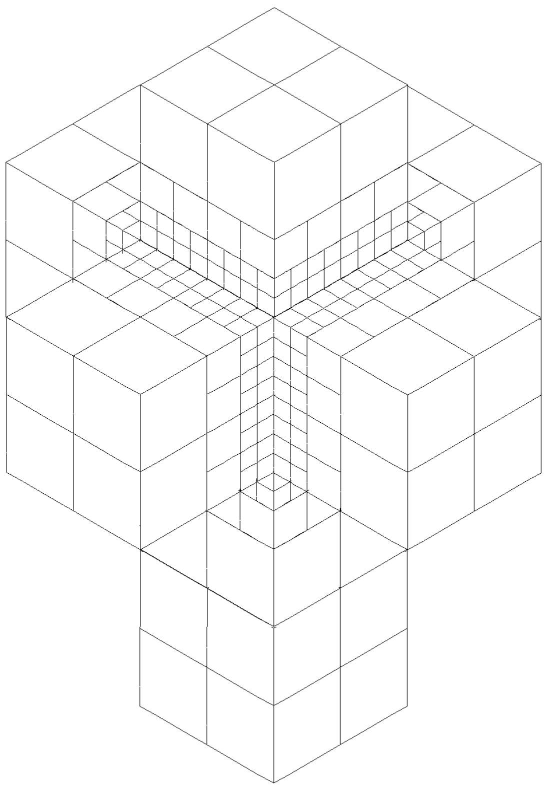
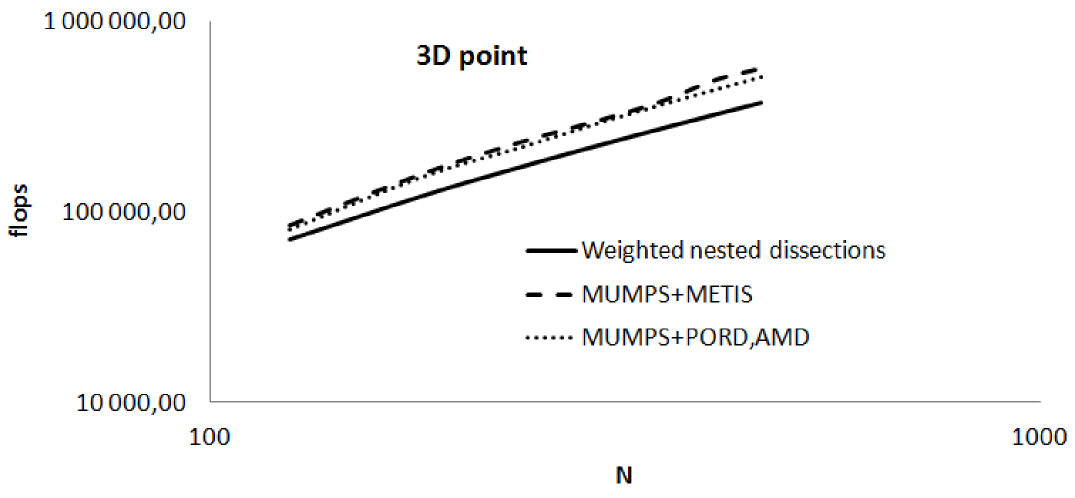
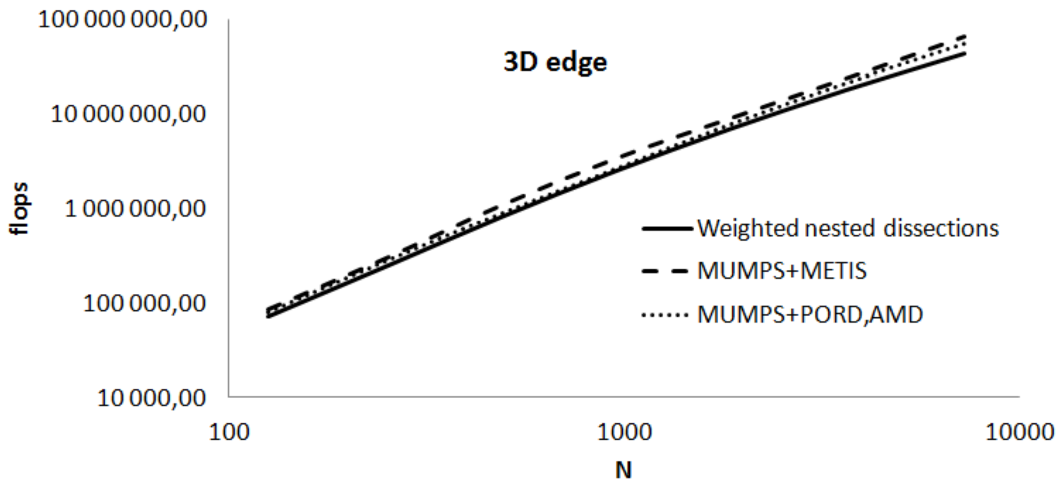

(d) Hyperface singularity.

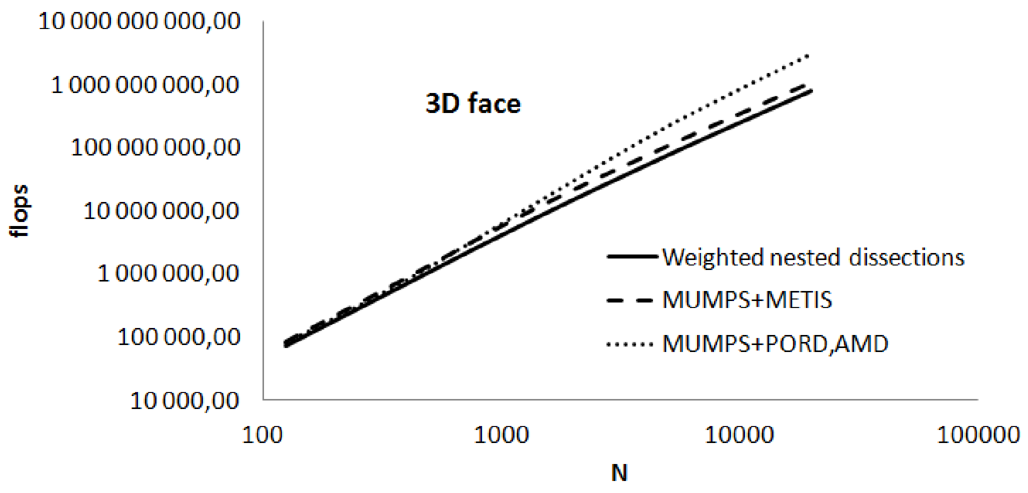
Figure 4: 4D meshes with singularities.



(a) 3D point singularity.



(b) 3D edge singularity.



(c) 3D face singularity.

Figure 5: Scalability of the weighted nested dissections algorithm in 3D.

3 NUMERICAL RESULTS

In this section we present numerical experiments concerning the execution time of our bisections weighted by element size algorithm for 3D and 4D grids with singularities. We show that our method outperforms MUMPS [4, 5] state of the art solver with (approximate) minimum degree or filling orderings [6, 7, 8], METIS [9] or PORD [10] orderings. The comparison is presented in Figures 5 and 6, for 3D and 4D respectively. The FLOPs for the 4D singularities, namely point singularity (kind=0), edge singularity (kind=1), face singularity (kind=2), hyperface singularity (kind=3) are also summarized in Table 1.

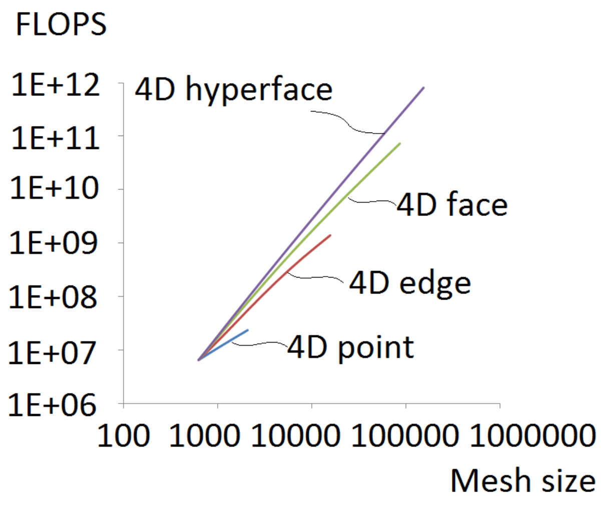


Figure 6: Scalability of the weighted nested dissections algorithm in 4D.

4 CONCLUSIONS

We have presented preliminary results for a promising method for the generation of the orderings for the multi–frontal solver algorithm — the numerical results show that the approach presented can outperform the current state of the art solutions. At the same time we have verified a novel approach to solve space–time formulations considering the time dimension as another space dimension. We intend to investigate this approach and the possibilities it presents for various finite element methods. In particular, it promises benefits over traditional algorithms in the areas of adaptivity and parallelization.

Table 1: Floating point operation costs for four dimensional singularities.

Dimensions	Singularity	Depth	Elements	Nodes	FLOPs
4	0	0	1	81	180441
4	0	1	16	625	6438573
4	0	2	31	865	9260925
4	0	3	46	1105	12083277
4	0	4	61	1345	14905629
4	0	5	76	1585	17727981
4	0	6	91	1825	20550333
4	0	7	106	2065	23372685
4	0	8	121	2305	26195037
4	0	9	136	2545	29017389
4	0	10	151	2785	31839741
4	0	11	166	3025	34662093
4	0	12	181	3265	37484445
4	0	13	196	3505	40306797
4	0	14	211	3745	43129149
4	0	15	226	3985	45951501
4	0	16	241	4225	48773853
4	0	17	256	4465	51596205
4	0	18	271	4705	54418557
4	0	19	286	4945	57240909
4	1	0	1	81	180441
4	1	1	16	625	6438573
4	1	2	46	1161	15728465
4	1	3	106	2177	45010855
4	1	4	226	4153	132140065
4	1	5	466	8049	372950179
4	1	6	946	15785	986944501
4	1	7	1906	31201	2449394343
4	1	8	3826	61977	5755536601
4	1	9	7666	123473	12948968907
4	1	10	15346	246409	28178439933
4	2	0	1	81	180441
4	2	1	16	625	6438573
4	2	2	76	1821	46109087
4	2	3	316	6121	517831573
4	2	4	1276	22389	5999916811
4	2	5	5116	85633	64220568465
4	3	0	1	81	180441
4	3	1	16	625	6438573
4	3	2	136	3291	222691306
4	3	3	1096	21485	12813363367

ACKNOWLEDGEMENTS

The work presented in this paper has been supported by National Science Centre, Poland grant no. DEC-2012/06/M/ST1/00363.

REFERENCES

[1] H. AbouEisha, V. M. Calo, K. Jopek, M. Moshkov, A. Paszynska, M. Paszynski, M.Skotniczny, Optimization of Element Partition Trees for Two-Dimensional h Refined Meshes, submitted to *Computers and Mathematics with Applications* (2016)

[2] T.J.R. Hughes, G.M. Hulbert, Space-Time Finite Element Method for Elastodynamics: Formulations and Error Estimates, *Computer Methods in Applied Mechanics and Engineering* 66 (1988) 339-363

[3] L. Demkowicz, J. Gopalakrishnan, Recent Developments in Discontinuous Galerkin Finite Element Methods for Partial Differential Equations (eds. X. Feng, O. Karakashian, Y. Xing). In: vol. 157. IMA *Volumes in Mathematics and its Applications*, (2014). An Overview of the DPG Method, 149–180

[4] P. R. Amestoy, I. S. Duff, J. Koster and J.-Y. L'Excellent, 2001. A fully asynchronous multifrontal solver using distributed dynamic scheduling, *SIAM Journal of Matrix Analysis and Applications*, 23(1) :15–41.

[5] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent and S. Pralet, 2006. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing* Vol 32(2):136-156.

[6] P. Heggernes, S.C. Eisenstat, G. Kumfert, A. Pothen, 2001. The Computational Complexity of the Minimum Degree Algorithm, ICASE Report No. 2001-42.

[7] P. R. Amestoy, T. A. Davis, I. S. Du, 1996. An Approximate Minimum Degree Ordering Algorithm, *SIAM Journal of Matrix Analysis & Application*, 17(4):886-905.

[8] G.W. Flake, R.E. Tarjan, K. Tsioutsiouliklis, 2003. Graph clustering and minimum cut trees, *Internet Mathematics* 1:385-408.

[9] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM Journal of Scientiffic Computing*, 20, 1 (1998) 359-392.

[10] J. Schulze, Towards a tighter coupling of bottom-up and top-down sparse matrix ordering methods, *BIT*, 41, 4 (2001) 800.