# Hybridizing Nested Dissection and Halo Approximate Minimum Degree for Efficient Sparse Matrix Ordering[⋆] [⋆⋆]

François Pellegrini[1], Jean Roman[1], and Patrick Amestoy[2]

[1] LaBRI, UMR CNRS 5800, Université Bordeaux I & ENSERB
351, cours de la Libération, F-33405 Talence, France
{pelegrin|roman}@labri.u-bordeaux.fr
[2] ENSEEIHT-IRIT
2, rue Charles Camichel, 31071 Toulouse Cédex 7, France
Patrick.Amestoy@enseeiht.fr

**Abstract.** Minimum degree and nested dissection are the two most popular reordering schemes used to reduce fill-in and operation count when factoring and solving sparse matrices. Most of the state-of-the-art ordering packages hybridize these methods by performing incomplete nested dissection and ordering by minimum degree the subgraphs associated with the leaves of the separation tree, but most often only loose couplings have been achieved, resulting in poorer performance than could have been expected. This paper presents a tight coupling of the nested dissection and halo approximate minimum degree algorithms, which allows the minimum degree algorithm to use exact degrees on the boundaries of the subgraphs passed to it, and to yield back not only the ordering of the nodes of the subgraph, but also the amalgamated assembly subtrees, for efficient block computations.
Experimental results show the performance improvement of this hybridization, both in terms of fill-in reduction and increase of concurrency on a parallel sparse block symmetric solver.

## 1 Introduction

When solving large sparse linear systems of the form $Ax = b$, it is common to precede the numerical factorization by a symmetric reordering. This reordering is chosen in such a way that pivoting down the diagonal in order on the resulting permuted matrix $PAP^T$ produces much less fill-in and work than computing the factors of $A$ by pivoting down the diagonal in the original order (the fill-in is the set of zero entries in $A$ that become non-zero in the factored matrix).

The two most classically-used reordering methods are *minimum degree* and *nested dissection*. The minimum degree algorithm [24] is a local heuristic that performs its pivot selection by selecting from the graph a node of minimum degree. The nested dissection algorithm [9] is a global heuristic recursive algorithm

⋆ This work is supported by the French *Commissariat à l'Énergie Atomique* CEA/CESTA under contract No. 7V1555AC, and by the GDR ARP of the CNRS.
⋆⋆ This paper extends and improves results presented in a paper of same title that appeared in the proceedings of Irregular'99.

which computes a vertex set $S$ that separates the graph into two parts $A$ and $B$, ordering $S$ last. It then proceeds recursively on parts $A$ and $B$ until their sizes become smaller than some threshold value. This ordering guarantees that no non zero term can appear in the factorization process between unknowns of $A$ and unknowns of $B$.

The minimum degree algorithm is known to be a very fast and general purpose algorithm, and has received much attention over the last three decades (see for example [1, 10, 19]). However, the algorithm is intrinsically sequential, and very little can be theoretically proven about its efficiency.

On the other hand, many theoretical results have been carried out on nested dissection ordering [6, 18], and its divide and conquer nature makes it easily parallelizable. In practice, nested dissection produces orderings which, both in terms of fill-in and operation count, compare well to the ones obtained with minimum degree [12, 16, 21]. Moreover, the elimination trees induced by nested dissection are broader, shorter, and better balanced, and therefore exhibit much more concurrency in the context of parallel Cholesky factorization [4, 7, 8, 12, 21, 23, and included references].

Due to their complementary nature, several schemes have been proposed to hybridize the two methods [5, 13, 17, 21]. In the first attempts of hybridization, only loose couplings had been achieved: incomplete nested dissection was performed on the graph to order, and the resulting subgraphs were passed to some minimum degree algorithm. This resulted in the fact that the minimum degree algorithm did not have exact degree values for all of the boundary vertices of the subgraphs, leading to a misbehavior of the vertex selection process. Several tighter coupling schemes have then been proposed to avoid such inaccuracies [13, 22].

In this paper, we propose a tight coupling of the nested dissection and minimum degree algorithms, that allows each of them to take advantage of the information computed by the other. First, the nested dissection algorithm provides exact degree values for the boundary vertices of the subgraphs passed to the minimum degree algorithm (called *halo* minimum degree since it has a partial visibility of the neighborhood of the subgraph); this idea is very much related to an original idea proposed by Liu [20], recently experimented by [5, 13] in the context of nested dissection orderings. Second, the minimum degree algorithm returns the assembly tree that it computes for each subgraph, thus allowing for supervariable amalgamation, in order to obtain column-blocks of a size suitable for BLAS3 block computations.

This tight coupling has been implemented in the SCOTCH 3.4 software package for static mapping, graph partitioning, and sparse matrix ordering [21] being developed by the ALiENor (ALgorithmics and ENvironments for parallel computing) team at the *Laboratoire Bordelais de Recherche en Informatique* (LaBRI) of the Université Bordeaux I. The ALiENor team is currently developing a complete software chain for parallel solving of sparse linear systems [15], which comprises the PaStiX parallel sparse block fan-in Cholesky solver that has been used to validate out results.

The rest of the paper is organized as follows. Section 2 describes the metrics we use to evaluate the quality of orderings, and outlines the capabilities of the PaStiX solver. Section 3 presents the principle of our halo minimum degree algorithm, and evaluates its efficiency. Section 4 outlines the amalgamation process that is carried out on the assembly trees, and validates its interest in the context of block computations. Then comes the conclusion.

## 2 Metrics

All of the algorithms described in this paper have been integrated into version 3.4 of the Scotch static mapping and sparse matrix ordering software package [21] developed at the LaBRI, which has been used for all of our tests.

The quality of orderings is evaluated with respect to several criteria. The first one, called NNZ, is the overall number of non zero terms to store in the factored reordered matrix, including extra logical zeros that must be stored when block storage is used. The second one, OPC, is the operation count, that is, the number of arithmetic operations required to factor the matrix. To comply with existing RISC processor technology, the operation count that we consider in this paper accounts for all operations (additions, subtractions, multiplications, divisions) required by sequential Cholesky factorization, except square roots; it is equal to $\sum_c n_c^2$, where $n_c$ is the number of non-zeros of column $c$ of the factored matrix, diagonal included. The third criterion is the size, in integer words, of the secondary storage used to index the array of non-zeros. It is defined as $\mathrm{SS_B}$ when block storage is used, and $\mathrm{SS_C}$ when column compressed storage is used.

Although the use of solving times to compare orderings might lead to biased interpretation, because of possible artefacts resulting from interactions between the ordering, the implementation of the solver, and the operating system, it of absolute necessity to show such results, due to the lack of simple and efficient metrics to measure the impact of amalgamation on instruction-level parallelism and cache friendliness induced by BLAS block computations.

All the sequential computations of orderings were run on a 332MHz PowerPC 604E-based RS6000 machine with 512 Mb of main memory.

All the parallel solving experiments with PaStiX were run at CNUSC on an IBM SP2 with 120 MHz Power2SC thin nodes having 256 MBytes of physical memory each. However, we only ran test cases modeled by connected graphs, since PaStiX does not handle disconnected graphs yet, and that did not cause the nodes to swap, since swapping dramatically decreases solver performance. The times recorded are the best elapsed wall-clock times over three consecutive runs of the solver, to level operating-system artefacts. At the time being, PaStiX handles 1D block distributions only, but will be able to handle 2D block distributions soon, yielding even better solving times.

**Table 1.** Description of our test problems. $NNZ_A$ is the number of extra-diagonal terms in the triangular matrix after it has been symmetrized, whenever necessary.

| Name | Columns | $NNZ_A$ | Description |
| --- | --- | --- | --- |
| 144 | 144649 | 1074393 | 3D finite element mesh |
| 598A | 110971 | 741934 | 3D finite element mesh |
| AATKEN | 42659 | 88237 | † |
| AUTO | 448695 | 3314611 | 3D finite element mesh † |
| B5TUER | 162610 | 3873534 | Parasol matrix |
| BCSSTK29 | 13992 | 302748 | 3D stiffness matrix † |
| BCSSTK30 | 28924 | 1007284 | 3D stiffness matrix |
| BCSSTK31 | 35588 | 572914 | 3D stiffness matrix † |
| BCSSTK32 | 44609 | 985046 | 3D stiffness matrix |
| BMW32 | 227362 | 5530634 | 3D stiffness matrix |
| BMW7ST1 | 141347 | 3599160 | 3D stiffness matrix † |
| BMWCRA1 | 148770 | 5247616 | Parasol matrix |
| BRACKET | 62631 | 366559 | 3D finite element mesh |
| CRANKSEG1 | 52804 | 5280703 | 3D stiffness matrix |
| CRANKSEG2 | 63838 | 7042510 | 3D stiffness matrix |
| M14B | 214765 | 3358036 | 3D finite element mesh |
| OCEAN | 143437 | 409593 | 3D finite element mesh |
| OILPAN | 73752 | 1761718 | 3D stiffness matrix |
| QUER | 59122 | 1403689 | Parasol matrix |
| ROTOR | 99617 | 662431 | 3D finite element mesh |
| SHIP001 | 34920 | 2304655 | Parasol matrix |
| SHIP003 | 121728 | 3982153 | Parasol matrix |
| SHIPSEC1 | 140874 | 3836265 | Parasol matrix † |
| SHIPSEC5 | 179860 | 4966618 | Parasol matrix |
| SHIPSEC8 | 114919 | 3269240 | Parasol matrix |
| THREAD | 29736 | 2220156 | Parasol matrix |
| TOOTH | 78136 | 452591 | 3D finite element mesh |
| X104 | 108384 | 5029620 | Parasol matrix |

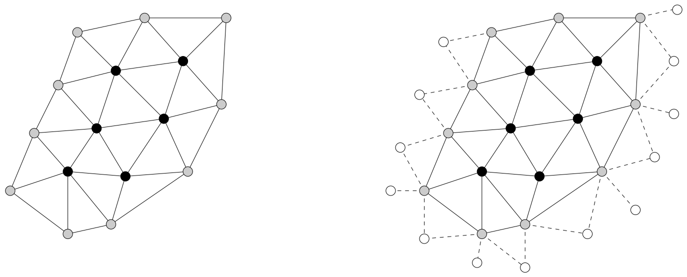
† Disconnected graph.

## 3 Halo Minimum Degree Ordering

In the simpler hybridizations of nested dissection and minimum degree, incomplete nested dissection is performed on the original graph, and the subgraphs at the leaves of the separation tree are ordered using a minimum degree heuristic. The switching from nested dissection to minimum degree is controlled by a criterion which is typically a threshold on the size of the subgraphs. It is for instance the case for ON-METIS 4.0 [17], which switches from nested dissection to multiple minimum degree at a threshold varying between 100 and 200 vertices, depending on the structure of the original graphs.

The problem in this approach is that the degrees of the boundary vertices of the subgraphs passed to the minimum degree algorithm are much smaller than they would be if the original graph were ordered using minimum degree only. Consequently, the minimum degree algorithm will tend to order boundary vertices first, creating a "skin" of low-degree vertices that will favor fill-in between separator vertices and interior vertices, and across neighbors of the boundary vertices.

To avoid such a behavior, the minimum degree algorithm must work on subgraphs that have real vertex degrees on their boundary vertices, to compute good orderings of all of the subgraph vertices. In order to do that, our nested dissection algorithm adds a topological description of the immediate surroundings of the subgraph with respect to the original graph (see figure 1). This additional information, called *halo*, is then exploited by our modified approximate minimum degree [1] algorithm to reoder subgraph nodes only. The assembly tree (that is, the tree of supervariables) of both the subgraph and halo nodes is then built, for future use (see section 4). We refer to this modified approximate minimum degree as Halo-AMD, or HAMD.



**a.** Without halo. **b.** With halo.

**Fig. 1.** Knowledge of subgraph vertex degrees by the minimum degree algorithm without and with halo. Black vertices are subgraph internal vertices, grey vertices are subgraph boundary vertices, and white vertices are halo vertices, which are seen by the halo minimum degree algorithm but do not belong to the subgraph.

Since it is likely that, in most cases (and especially for finite-element graphs), all of the vertices of a given separator will be linked by paths of smaller reordered numbers in the separated subsets, leading to completely filled diagonal blocks, we order separators using a breadth-first traversal of each separator subgraph from a pseudo-peripherial node [11], which proves to reduce the number of extra-diagonal blocks. Hendrickson and Rothberg report [13, 14] that performing minimum degree to reorder last the subgraph induced by all of the separators found in the nested dissection process can lead to significant NNZ and OPC improvement, especially for stiffness matrices. This hybridization type could also be investigated using our HAMD algorithm, but might be restricted to sequential solving, since breaking the separator hierarchy is likely to reduce concurrency in the elimination tree.

To experiment with the HAMD algorithm, we have run our orderer on several test graphs (see table 1) with three distinct strategies. In the first one, we perform graph compression [3, 13] whenever necessary (referred to as CP), then multi-level nested dissection (ND) with a subgraph size threshold of 120 to match ON-METIS's threshold, and finally the approximate minimum degree of [1] (AMD, see table 3). In the second strategy, we replace the AMD algorithm by our halo counterpart, HAMD (table 2). In the third one, AMD is used to reorder the whole graph (see table 4).

Using HAMD instead of AMD for ordering the subgraphs reduces both the NNZ and the OPC (see tables 2 and 3). For a threshold value of 120, gains for our test graphs range from 0 to 13 percent, with an average of about 5.25 percent for NNZ and 3.29 percent for OPC.

When the threshold value increases (see figures 2 and 3), gains tend to increase, as more space is left for HAMD to outperform AMD, and then eventually decrease to zero since AMD and HAMD are equivalent on the whole graph. However, for most graphs in our collection, the NNZ and OPC computed by CP+ND+HAMD increase along with the threshold. Therefore, the threshold should be kept small.

Two classes of graphs arise in all of these experiments. The first one, containing finite-element-like meshes, is very stable with respect to the threshold (see figure 2). The second one, which includes stiffness matrices, has a very unstable behavior for small threshold values (see figure 3). However, the orderings produced by HAMD are more stable with respect to threshold, and, for every given threshold, almost always outperform the orderings produced by AMD.

Due to the small threshold used, the orderings computed by our CP+ND+HAMD algorithm are only slightly better on average than the ones computed by ON-METIS 4.0, although the latter does not use halo information (see table 5). This is especially true for stiffness matrices.

These experiments also confirm the current overall superiority of hybrid ordering methods against pure minimum degree methods (see table 4); concurring results have been obtained by other authors [12, 13, 16]. Let us also note that using block secondary storage always leads to much smaller secondary structures than when using column compressed storage (see tables 2 and 5).

**Table 2.** Ordering results with CP+ND+HAMD, with threshold 120. Bold means best over tables 2 to 5.

| Name | NNZ | OPC | $SS_B$ |
|---|---|---|---|
| 144 | **4.678789e+07** | **5.461371e+10** | 1.957599e+06 |
| 598A | **2.593652e+07** | **1.862919e+10** | 1.490999e+06 |
| AATKEN | 3.455104e+07 | 1.689186e+11 | **7.089580e+05** |
| AUTO | **2.257955e+08** | **4.775824e+11** | 6.489490e+06 |
| B5TUER | 2.418426e+07 | 1.334469e+10 | **2.169160e+05** |
| BCSSTK29 | **1.554762e+06** | **3.053062e+08** | 3.884200e+04 |
| BCSSTK30 | 4.408501e+06 | 1.200915e+09 | 6.721100e+04 |
| BCSSTK31 | **4.346033e+06** | **1.173140e+09** | **1.905510e+05** |
| BCSSTK32 | 5.473940e+06 | 1.296364e+09 | 1.269430e+05 |
| BMW32 | **4.443599e+07** | **2.894965e+10** | **5.069520e+05** |
| BMW7ST1 | **2.494193e+07** | 1.168608e+10 | **2.771390e+05** |
| BMWCRA1 | **6.517092e+07** | **5.499588e+10** | 4.945300e+05 |
| BRACKET | **5.936583e+06** | **1.837834e+09** | 8.208650e+05 |
| CRANKSEG1 | **3.234555e+07** | **3.162551e+10** | **1.069970e+05** |
| CRANKSEG2 | **4.147166e+07** | **4.390315e+10** | **1.219820e+05** |
| M14B | **6.232474e+07** | **5.929754e+10** | 2.934853e+06 |
| OCEAN | **1.889604e+07** | **1.043513e+10** | 1.996008e+06 |
| OILPAN | **8.730750e+06** | **2.618641e+09** | **1.006520e+05** |
| QUER | **9.150519e+06** | **3.285157e+09** | **7.022000e+04** |
| ROTOR | **1.576639e+07** | **9.196470e+09** | 1.345494e+06 |
| SHIP001 | **1.448393e+07** | **9.295417e+09** | 4.961700e+04 |
| SHIP003 | **5.828901e+07** | 7.654139e+10 | 3.124900e+05 |
| SHIPSEC1 | **3.824817e+07** | 3.690482e+10 | 2.514140e+05 |
| SHIPSEC5 | 5.373636e+07 | 6.040098e+10 | 3.543250e+05 |
| SHIPSEC8 | 3.626482e+07 | 4.042770e+10 | 2.518370e+05 |
| THREAD | **2.424793e+07** | 3.766394e+10 | **6.538700e+04** |
| TOOTH | **1.063089e+07** | **6.536985e+09** | 1.029272e+06 |
| X104 | 2.569631e+07 | 1.506416e+10 | **1.072220e+05** |

**Table 3.** Ordering results with CP+ND+AMD, with threshold 120, compared to CP+ND+HAMD (table 2). Bold means better than CP+ND+HAMD.

| Name | NNZ | % | OPC | % | $SS_B$ |
|---|---|---|---|---|---|
| 144 | 4.738654e+07 | 1.28 | 5.471596e+10 | 0.19 | **1.841084e+06** |
| 598A | 2.631221e+07 | 1.45 | 1.868418e+10 | 0.30 | **1.443926e+06** |
| AATKEN | 3.472070e+07 | 0.49 | 1.689293e+11 | 0.01 | 9.351200e+05 |
| AUTO | 2.272260e+08 | 0.63 | 4.778201e+11 | 0.05 | **5.896011e+06** |
| B5TUER | 2.619512e+07 | 8.31 | 1.381057e+10 | 3.49 | 2.427990e+05 |
| BCSSTK29 | 1.680634e+06 | 8.10 | 3.264788e+08 | 6.93 | **2.222300e+04** |
| BCSSTK30 | 4.777067e+06 | 8.36 | 1.316765e+09 | 9.65 | **6.617400e+04** |
| BCSSTK31 | 4.601165e+06 | 5.87 | 1.224240e+09 | 4.36 | 1.936810e+05 |
| BCSSTK32 | 5.872585e+06 | 7.28 | 1.389394e+09 | 7.18 | **1.259680e+05** |
| BMW32 | 4.775072e+07 | 7.46 | 2.995499e+10 | 3.47 | 5.122640e+05 |
| BMW7ST1 | 2.715318e+07 | 8.87 | 1.237822e+10 | 5.92 | 2.828700e+05 |
| BMWCRA1 | 6.797851e+07 | 4.31 | 5.651181e+10 | 2.76 | **4.597520e+05** |
| BRACKET | 6.132834e+06 | 3.31 | 1.857957e+09 | 1.09 | **7.739580e+05** |
| CRANKSEG1 | 3.340571e+07 | 3.28 | 3.255758e+10 | 2.95 | 1.098290e+05 |
| CRANKSEG2 | 4.317407e+07 | 4.10 | 4.546202e+10 | 3.55 | 1.256000e+05 |
| M14B | 6.322984e+07 | 1.45 | 5.945245e+10 | 0.26 | **2.709939e+06** |
| OCEAN | 1.914709e+07 | 1.33 | 1.045682e+10 | 0.21 | **1.844229e+06** |
| OILPAN | 9.675960e+06 | 10.83 | 2.833273e+09 | 8.20 | 1.090240e+05 |
| QUER | 9.755718e+06 | 6.61 | 3.442419e+09 | 4.79 | 8.159200e+04 |
| ROTOR | 1.617002e+07 | 2.56 | 9.243690e+09 | 0.51 | **1.340153e+06** |
| SHIP001 | 1.524306e+07 | 5.24 | 9.850410e+09 | 5.97 | **4.773400e+04** |
| SHIP003 | 6.107896e+07 | 4.79 | 7.808003e+10 | 2.01 | **2.622760e+05** |
| SHIPSEC1 | 4.106704e+07 | 7.37 | 3.786139e+10 | 2.59 | 2.832120e+05 |
| SHIPSEC5 | 5.809485e+07 | 8.11 | 6.196687e+10 | 2.59 | 3.828750e+05 |
| SHIPSEC8 | 3.870575e+07 | 6.73 | 4.134933e+10 | 2.28 | **2.388970e+05** |
| THREAD | 2.499077e+07 | 3.06 | 3.825862e+10 | 1.58 | 7.033600e+04 |
| TOOTH | 1.087384e+07 | 2.29 | 6.560264e+09 | 0.36 | **9.681470e+05** |
| X104 | 2.915018e+07 | 13.44 | 1.639143e+10 | 8.81 | 1.254700e+05 |
| Average | | 5.25 | | 3.29 | |

**Table 4.** Ordering results with AMD alone, compared to CP+ND+HAMD (table 2). Bold means better than CP+ND+HAMD.

| Name | NNZ | % | OPC | % | $SS_B$ |
|---|---|---|---|---|---|
| 144 | 9.393420e+07 | 100.77 | 2.406120e+11 | 340.57 | 2.963011e+06 |
| 598A | 4.592155e+07 | 77.05 | 6.224629e+10 | 234.13 | 2.293154e+06 |
| AATKEN | **2.589400e+07** | -25.06 | **1.202562e+11** | -28.81 | 9.855260e+05 |
| AUTO | 5.386292e+08 | 138.55 | 2.701255e+12 | 465.61 | 1.044489e+07 |
| B5TUER | 2.707972e+07 | 11.97 | 1.554106e+10 | 16.46 | 2.364930e+05 |
| BCSSTK29 | 1.786234e+06 | 14.89 | 4.714500e+08 | 54.42 | **3.819000e+04** |
| BCSSTK30 | **3.852124e+06** | -12.62 | **9.446337e+08** | -21.34 | 7.488900e+04 |
| BCSSTK31 | 5.568769e+06 | 28.13 | 2.909362e+09 | 148.00 | 2.123520e+05 |
| BCSSTK32 | **4.989134e+06** | -8.86 | **9.535496e+08** | -26.44 | 1.404360e+05 |
| BMW32 | 5.071095e+07 | 14.12 | 4.721008e+10 | 63.08 | 5.406270e+05 |
| BMW7ST1 | 2.613003e+07 | 4.76 | 1.466054e+10 | 25.45 | 2.983190e+05 |
| BMWCRA1 | 9.718952e+07 | 49.13 | 1.308578e+11 | 137.94 | 5.339190e+05 |
| BRACKET | 7.286548e+06 | 22.74 | 3.057435e+09 | 66.36 | 8.804010e+05 |
| CRANKSEG1 | 4.036152e+07 | 24.78 | 5.256665e+10 | 66.22 | 1.396250e+05 |
| CRANKSEG2 | 5.866411e+07 | 41.46 | 9.587950e+10 | 118.39 | 1.685880e+05 |
| M14B | 1.105011e+08 | 77.30 | 1.880238e+11 | 217.09 | 4.030059e+06 |
| OCEAN | 3.306994e+07 | 75.01 | 3.557219e+10 | 240.89 | 2.322347e+06 |
| OILPAN | 1.013450e+07 | 16.08 | 4.183709e+09 | 59.77 | 1.128240e+05 |
| QUER | 9.859892e+06 | 7.75 | 4.096786e+09 | 24.71 | 1.042230e+05 |
| ROTOR | 2.475696e+07 | 57.02 | 2.491224e+10 | 170.89 | 1.615861e+06 |
| SHIP001 | 1.561146e+07 | 7.78 | 1.131159e+10 | 21.69 | 5.714400e+04 |
| SHIP003 | 7.258900e+07 | 24.53 | 1.360651e+11 | 77.77 | **2.888790e+05** |
| SHIPSEC1 | 3.892702e+07 | 1.77 | 4.254132e+10 | 15.27 | **2.442960e+05** |
| SHIPSEC5 | **5.102028e+07** | -5.05 | 6.547374e+10 | 8.40 | **3.161130e+05** |
| SHIPSEC8 | 4.423392e+07 | 21.97 | 8.425908e+10 | 108.42 | **2.197700e+05** |
| THREAD | 2.693583e+07 | 11.09 | 4.841782e+10 | 28.55 | 7.051100e+04 |
| TOOTH | 1.613290e+07 | 51.75 | 1.810155e+10 | 176.91 | 1.136381e+06 |
| X104 | **2.296873e+07** | -10.61 | **9.055344e+09** | -39.89 | 1.074140e+05 |
| Average | | 29.22 | | 98.95 | |

**Table 5.** Ordering results with ON-METIS 4.0, compared to CP+ND+HAMD (table 2). Bold means better than CP+ND+HAMD.

| Name | NNZ | % | OPC | % | $SS_C$ |
|---|---|---|---|---|---|
| 144 | 5.006260e+07 | 7.00 | 6.288269e+10 | 15.14 | 3.020012e+06 |
| 598A | 2.670144e+07 | 2.95 | 1.945231e+10 | 4.42 | 2.119415e+06 |
| AATKEN | **3.297272e+07** | -4.57 | **1.545334e+11** | -8.52 | 5.789739e+06 |
| AUTO | 2.358388e+08 | 4.45 | 5.140633e+11 | 7.64 | 1.023613e+07 |
| B5TUER | **2.403873e+07** | -0.60 | **1.236666e+10** | -7.33 | 1.522689e+06 |
| BCSSTK29 | 1.699841e+06 | 9.33 | 3.502626e+08 | 14.73 | 1.753670e+05 |
| BCSSTK30 | 4.530240e+06 | 2.76 | 1.216158e+09 | 1.27 | 3.120820e+05 |
| BCSSTK31 | 4.386990e+06 | 0.94 | 1.178024e+09 | 0.42 | 4.363570e+05 |
| BCSSTK32 | 5.605579e+06 | 2.40 | **1.230208e+09** | -5.10 | 5.073400e+05 |
| BMW32 | 4.593307e+07 | 3.37 | 2.815020e+10 | -2.76 | 2.619580e+06 |
| BMW7ST1 | 2.558782e+07 | 2.59 | **1.097226e+10** | -6.11 | 1.600835e+06 |
| BMWCRA1 | 6.980987e+07 | 7.12 | 6.123804e+10 | 11.35 | 2.534796e+06 |
| BRACKET | 6.025575e+06 | 1.50 | 1.851941e+09 | 0.77 | 9.413400e+05 |
| CRANKSEG1 | 3.343857e+07 | 3.38 | 3.284621e+10 | 3.86 | 7.667890e+05 |
| CRANKSEG2 | 4.320028e+07 | 4.17 | 4.521963e+10 | 3.00 | 9.279760e+05 |
| M14B | 6.652286e+07 | 6.74 | 6.791640e+10 | 14.53 | 4.399879e+06 |
| OCEAN | 1.950496e+07 | 3.22 | 1.178200e+10 | 12.91 | 2.247596e+06 |
| OILPAN | 9.064734e+06 | 3.83 | 2.750608e+09 | 5.04 | 6.773740e+05 |
| QUER | 9.585737e+06 | 4.76 | 3.447989e+09 | 4.96 | 5.740880e+05 |
| ROTOR | 1.627034e+07 | 3.20 | 9.460553e+09 | 2.87 | 1.702922e+06 |
| SHIP001 | 1.480857e+07 | 2.24 | 9.462321e+09 | 1.80 | 4.950340e+05 |
| SHIP003 | 5.909580e+07 | 1.38 | **7.587018e+10** | -0.88 | 1.992344e+06 |
| SHIPSEC1 | 3.903160e+07 | 2.05 | **3.617320e+10** | -1.98 | 1.660915e+06 |
| SHIPSEC5 | **5.256446e+07** | -2.18 | **5.508516e+10** | -8.80 | 2.153776e+06 |
| SHIPSEC8 | **3.585068e+07** | -1.14 | **3.715488e+10** | -8.10 | 1.402800e+06 |
| THREAD | 2.429792e+07 | 0.21 | **3.582543e+10** | -4.88 | 4.717110e+05 |
| TOOTH | 1.094157e+07 | 2.92 | 7.153447e+09 | 9.43 | 1.247352e+06 |
| X104 | 2.727631e+07 | 6.15 | **1.411544e+10** | -6.30 | 1.066218e+06 |
| Average | | 2.86 | | 1.91 | |

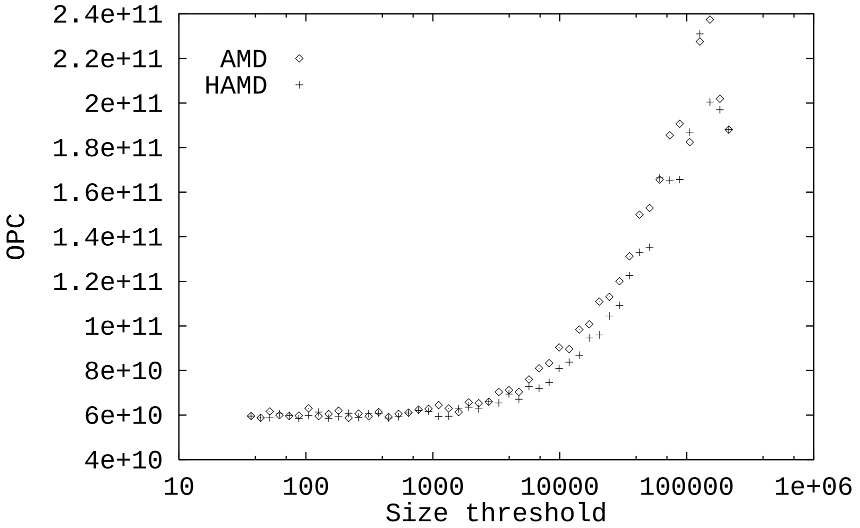



**Fig. 2.** Values of OPC obtained with CP+ND+AMD and CP+ND+HAMD for various threshold values, for graph M14B.

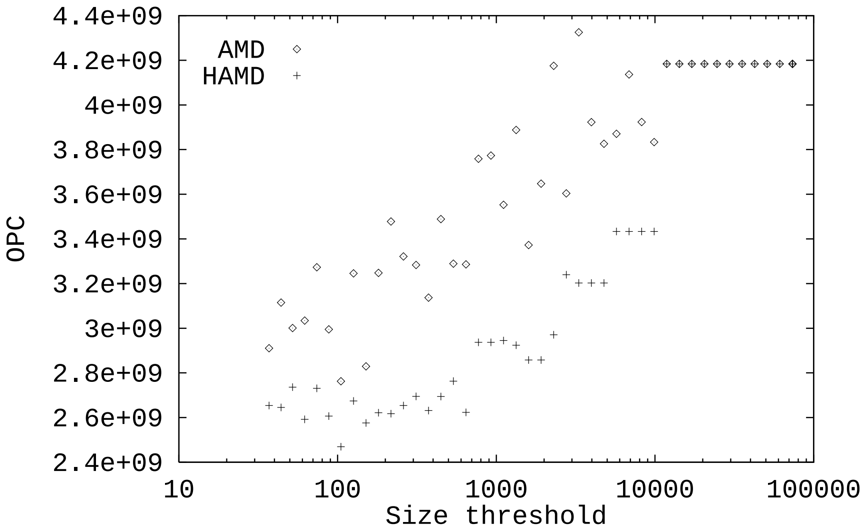



**Fig. 3.** Values of OPC obtained with CP+ND+AMD and CP+ND+HAMD for various threshold values, for graph OILPAN.

**Table 6.** Solving times, in seconds, for graphs ordered with AMD, CP+ND+AMD, and CP+ND+HAMD, with no amalgamation, for 2 and 16 processors. All other methods are compared to CP+ND+HAMD. Dashes indicate swapping.

| Name | CP+ND+HAMD | | CP+ND+AMD | | | |
|---|---|---|---|---|---|---|
| | 2 | 16 | 2 | % | 16 | % |
| 144 | – | 21.96 | – | – | **21.64** | -1.45 |
| 598A | – | **8.97** | – | – | 9.22 | 2.78 |
| B5TUER | **28.96** | **5.32** | 31.32 | 8.15 | 5.58 | 4.89 |
| BCSSTK30 | **3.75** | **1.07** | 4.01 | 6.93 | 1.07 | 0.00 |
| BCSSTK32 | **4.44** | **1.13** | 4.80 | 8.11 | 1.13 | 0.00 |
| BMW32 | – | **10.80** | – | – | 11.22 | 3.88 |
| BMWCRA1 | – | 17.21 | – | – | **17.02** | -1.10 |
| BRACKET | **12.12** | 1.97 | 12.29 | 1.40 | **1.84** | -6.59 |
| CRANKSEG1 | – | **10.32** | – | – | 10.42 | 0.97 |
| CRANKSEG2 | – | **13.47** | – | – | 13.77 | 2.23 |
| M14B | – | 24.44 | – | – | **23.94** | -2.04 |
| OCEAN | – | 6.73 | – | – | **6.20** | -7.88 |
| OILPAN | **7.15** | **1.41** | 7.96 | 11.32 | 1.50 | 6.38 |
| QUER | **8.28** | **1.89** | 8.77 | 5.91 | 2.06 | 8.99 |
| ROTOR | **36.21** | **5.71** | 36.94 | 2.01 | 5.92 | 3.68 |
| SHIP001 | **20.05** | 3.64 | 21.07 | 5.09 | **3.53** | -3.02 |
| SHIP003 | – | 24.86 | – | – | **24.07** | -3.18 |
| SHIPSEC5 | – | 20.66 | – | – | **20.60** | -0.29 |
| SHIPSEC8 | – | **14.14** | – | – | 14.68 | 3.82 |
| THREAD | – | 13.39 | – | – | **13.34** | -0.37 |
| TOOTH | 22.87 | **4.24** | **22.49** | -1.66 | 4.28 | 0.94 |
| X104 | – | 7.74 | – | – | **7.14** | -7.75 |
| Average | | | | 5.25 | | 0.27 |

To validate the HAMD algorithm, we have run our PaStiX solver on the connected matrices, that had been reordered with the CP+ND+HAMD and CP+ND+AMD algorithms, respectively, with no amalgamation at all (see section 4). We cannot compare the running times of the two above orderings against the ones computed by our AMD algorithm, since the ND algorithm clusters separator vertices in blocks, while our AMD algorithm either does not compute super-nodes at all, or performs amalgamation on the whole graph. Elapsed solving times, which are given for 2 and 16 processors in table 6, match rather well the behavior of the NNZ metric.

In most of the cases does CP+ND+HAMD yield shorter solving times than CP+ND+AMD, by 5.25 percent on average for 2 processors, and by 0.27 percent for 16 processors. The gain is amortized as the number of processor increases because at the time being PaStiX performs 1D block distributions only, so that the processing of the largest blocks at the root of the elimination tree is less efficient as the number of processors increases, thus hiding local gains.

## 4 Amalgamating supervariables

In order to perform efficient factorization, BLAS3-based block computations must be used with blocks of sufficient sizes. Therefore, it is necessary to amalgamate columns so as to obtain large enough blocks while keeping fill-in low.

Since, in order for the HAMD algorithm to behave well, the switching between the ND and HAMD methods must occur when subgraphs are large enough, the amalgamation process only takes place for the columns ordered by HAMD, as separators are of sufficient sizes. The amalgamation criterion that we have chosen is based on three parameters, cmin, cmax, and frat: a leaf of the assembly tree is merged to its father either if it has less columns than some threshold value cmin or if it induces less fill-in than a user-defined fraction frat of the surface of the merged block, provided that the merged block has less columns than some threshold value cmax.

The partition of the original graph into supervariables is achieved by merging the partition of the separators and the partition of the supervariables amalgamated in each subgraph. If the graph had been previously compressed, its partition is projected back to the uncompressed graph. One can therefore perform efficient block symbolic factorization in quasi-linear space and time complexities (the linearity is proven in the case of complete nested dissection [6]).

To show the impact of supervariable amalgamation, we have run our orderer with strategy CP+ND+HAMD, for the size threshold of 120, with a maximum desired fill-in ratio of 8 percent (frat $= 0.08$) alone, and with a maximum desired fill-in ratio of 8 percent and a minimum column block width of 16 columns (frat $= 0.08$, cmin $= 16$). The value of 8 percent has been chosen as it already prooved to yield a good trade-off between amalgamation and fill-in for the 2D multifrontal code MUMPS [2] (partially supported by the PARASOL project, EU ESPRIT IV LTR project 20160). OPC and $SS_B$ results are summarized in tables 7 and 8, respectively.

**Table 7.** OPC for CP+ND+HAMD, for threshold 120, and amalgamation criteria of frat $= 0.08$ and frat $= 0.08$, cmin $= 16$, compared to case cmax $= 0$ (table 3).

| Name | frat=0.08, cmin=1 | % | frat=0.08, cmin=16 | % |
|---|---|---|---|---|
| 144 | 5.468517e+10 | 0.13 | 5.495113e+10 | 0.62 |
| 598A | 1.867331e+10 | 0.24 | 1.886785e+10 | 1.28 |
| AATKEN | 1.689214e+11 | 0.00 | 1.689818e+11 | 0.04 |
| AUTO | 4.778411e+11 | 0.05 | 4.787798e+11 | 0.25 |
| B5TUER | 1.336942e+10 | 0.19 | 1.723610e+10 | 29.16 |
| BCSSTK29 | 3.093917e+08 | 1.34 | 3.253718e+08 | 6.57 |
| BCSSTK30 | 1.243483e+09 | 3.54 | 1.494921e+09 | 24.48 |
| BCSSTK31 | 1.189820e+09 | 1.42 | 1.382428e+09 | 17.84 |
| BCSSTK32 | 1.320103e+09 | 1.83 | 1.800437e+09 | 38.88 |
| BMW32 | 2.908825e+10 | 0.48 | 3.331269e+10 | 15.07 |
| BMW7ST1 | 1.178471e+10 | 0.84 | 1.451557e+10 | 24.21 |
| BMWCRA1 | 5.661030e+10 | 2.94 | 5.802631e+10 | 5.51 |
| BRACKET | 1.849821e+09 | 0.65 | 1.920520e+09 | 4.50 |
| CRANKSEG1 | 3.275112e+10 | 3.56 | 3.492279e+10 | 10.43 |
| CRANKSEG2 | 4.554016e+10 | 3.73 | 4.865639e+10 | 10.83 |
| M14B | 5.941112e+10 | 0.19 | 5.981520e+10 | 0.87 |
| OCEAN | 1.048062e+10 | 0.44 | 1.062923e+10 | 1.86 |
| OILPAN | 2.627800e+09 | 0.35 | 4.352106e+09 | 66.20 |
| QUER | 3.288626e+09 | 0.11 | 5.035545e+09 | 53.28 |
| ROTOR | 9.223602e+09 | 0.30 | 9.359407e+09 | 1.77 |
| SHIP001 | 9.796208e+09 | 5.39 | 1.121096e+10 | 20.61 |
| SHIP003 | 7.674996e+10 | 0.27 | 8.191355e+10 | 7.02 |
| SHIPSEC1 | 3.699502e+10 | 0.24 | 4.162324e+10 | 12.79 |
| SHIPSEC5 | 6.055100e+10 | 0.25 | 6.605023e+10 | 9.35 |
| SHIPSEC8 | 4.059116e+10 | 0.40 | 4.452713e+10 | 10.14 |
| THREAD | 3.840692e+10 | 1.97 | 3.862608e+10 | 2.55 |
| TOOTH | 6.553192e+09 | 0.25 | 6.638526e+09 | 1.55 |
| X104 | 1.528426e+10 | 1.46 | 1.800651e+10 | 19.53 |
| Average | | 1.16 | | 14.18 |

**Table 8.** $SS_B$ for CP+ND+HAMD, for threshold 120, and amalgamation criteria of $\mathsf{frat} = 0.08$ and $\mathsf{frat} = 0.08, \mathsf{cmin} = 16$, compared to case $\mathsf{cmax} = 0$ (table 3).

| Name | frat=0.08, cmin=1 | % | frat=0.08, cmin=16 | % |
|---|---|---|---|---|
| 144 | 9.676890e+05 | -50.57 | 4.786550e+05 | -75.55 |
| 598A | 8.214920e+05 | -44.90 | 3.566440e+05 | -76.08 |
| AATKEN | 6.964410e+05 | -1.77 | 3.853260e+05 | -45.65 |
| AUTO | 3.257927e+06 | -49.80 | 1.657746e+06 | -74.45 |
| B5TUER | 2.159150e+05 | -0.46 | 2.858900e+04 | -86.82 |
| BCSSTK29 | 2.358200e+04 | -39.29 | 1.524400e+04 | -60.75 |
| BCSSTK30 | 3.473200e+04 | -48.32 | 1.254700e+04 | -81.33 |
| BCSSTK31 | 1.242520e+05 | -34.79 | 3.921600e+04 | -79.42 |
| BCSSTK32 | 7.994200e+04 | -37.03 | 2.173600e+04 | -82.88 |
| BMW32 | 3.403040e+05 | -32.87 | 9.589900e+04 | -81.08 |
| BMW7ST1 | 1.930970e+05 | -30.32 | 5.187200e+04 | -81.28 |
| BMWCRA1 | 1.374490e+05 | -72.21 | 7.957100e+04 | -83.91 |
| BRACKET | 5.995440e+05 | -26.96 | 2.060600e+05 | -74.90 |
| CRANKSEG1 | 3.187000e+04 | -70.21 | 1.658900e+04 | -84.50 |
| CRANKSEG2 | 3.559600e+04 | -70.82 | 1.906900e+04 | -84.37 |
| M14B | 1.442502e+06 | -50.85 | 7.175060e+05 | -75.55 |
| OCEAN | 1.380497e+06 | -30.84 | 6.186290e+05 | -69.01 |
| OILPAN | 1.002360e+05 | -0.41 | 1.269600e+04 | -87.39 |
| QUER | 6.847600e+04 | -2.48 | 8.262000e+03 | -88.23 |
| ROTOR | 7.436850e+05 | -44.73 | 3.195390e+05 | -76.25 |
| SHIP001 | 2.137100e+04 | -56.93 | 9.210000e+03 | -81.44 |
| SHIP003 | 2.589300e+05 | -17.14 | 9.364500e+04 | -70.03 |
| SHIPSEC1 | 1.915170e+05 | -23.82 | 5.861000e+04 | -76.69 |
| SHIPSEC5 | 2.637950e+05 | -25.55 | 9.507300e+04 | -73.17 |
| SHIPSEC8 | 1.908400e+05 | -24.22 | 6.758000e+04 | -73.17 |
| THREAD | 1.960700e+04 | -70.01 | 1.573000e+04 | -75.94 |
| TOOTH | 7.538740e+05 | -26.76 | 2.768840e+05 | -73.10 |
| X104 | 7.070000e+04 | -34.06 | 2.222700e+04 | -79.27 |
| Average | | -36.36 | | -76.86 |

Performing amalgamation using the fill-in ratio frat alone has only a small impact on fill-in, since the average increase of OPC is just a bit more than 1 percent, but yet has a noticeable effect on the number of blocks, since the size of the secondary storage decreases by more than 36 percent. This reduction is not important in terms of memory occupation, since the amount of secondary storage is more than one order of magnitude less than NNZ, but it shows well that efficient amalgamation can be performed only locally on the subgraphs, at a cost much lower than if global amalgamation were performed in a post-processing phase.

Imposing a threshold on the size of the column blocks leads to much more aggressive amalgamation, since $SS_B$ decreases dramatically by more than 75 percents, but at the expense of an increase of OPC by about 14 percent on average, which may not be bearable on practice if it is not compensated by increased instruction-level parallelism.

**Table 9.** Solving times, in seconds, for graphs ordered with CP+ND+HAMD with amalgamation criteria frat = 0.08 and frat = 0.08, cmin = 16, compared to case cmax = 0 (table 6). Dashes indicate swapping.

| Name | frat=0.08 | | | | frat=0.08,cmin=16 | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | % | 16 | % | 2 | % | 16 | % |
| 144 | – | – | **20.43** | -6.97 | – | – | 20.72 | -5.65 |
| 598A | – | – | 8.73 | -2.68 | – | – | **8.32** | -7.25 |
| B5TUER | 29.10 | 0.48 | 5.33 | 0.18 | – | – | 6.25 | 17.48 |
| BCSSTK30 | **3.51** | -6.41 | **1.00** | -6.55 | 3.90 | 4.00 | 1.05 | -1.87 |
| BCSSTK32 | **4.05** | -8.79 | **1.04** | -7.97 | 4.60 | 3.60 | 1.19 | 5.30 |
| BMW32 | – | – | **10.46** | -3.15 | – | – | 11.22 | 3.88 |
| BMWCRA1 | – | – | **16.65** | -3.25 | – | – | 16.80 | -2.38 |
| BRACKET | 10.58 | -12.71 | 1.73 | -12.19 | **7.39** | -39.03 | **1.36** | -30.97 |
| CRANKSEG1 | – | – | **10.24** | -0.78 | – | – | 10.67 | 3.39 |
| CRANKSEG2 | – | – | 13.69 | 1.63 | – | – | 14.76 | 9.57 |
| M14B | – | – | 22.38 | -8.43 | – | – | **21.21** | -13.22 |
| OCEAN | 38.58 | – | 5.92 | -12.04 | **32.76** | – | **5.27** | -21.70 |
| OILPAN | **7.07** | -1.12 | **1.39** | -1.42 | 10.27 | 43.63 | 1.83 | 29.78 |
| QUER | **8.05** | -2.78 | **1.89** | 0.00 | 11.38 | 37.43 | 2.24 | 18.51 |
| ROTOR | 31.08 | -14.17 | 5.09 | -10.86 | **26.20** | -27.65 | **4.59** | -19.62 |
| SHIP001 | 20.64 | 2.94 | **3.52** | -3.30 | 22.77 | 13.56 | 4.03 | 10.71 |
| SHIP003 | – | – | **24.78** | -0.33 | – | – | 26.04 | 4.74 |
| SHIPSEC5 | – | – | 20.70 | 0.19 | – | – | 21.49 | 4.01 |
| SHIPSEC8 | – | – | 14.81 | 4.73 | – | – | 15.08 | 6.64 |
| THREAD | – | – | **13.14** | -1.87 | – | – | 13.21 | -1.35 |
| TOOTH | 21.76 | -4.86 | 4.19 | -1.18 | **18.02** | -21.21 | **3.78** | -10.85 |
| X104 | – | – | 7.86 | 1.55 | – | – | **6.67** | -13.83 |
| Average | | -5.26 | | -3.40 | | 1.79 | | -0.67 |

The measured solving times account for the benefits of amalgamation. Using the amalgamation criterion $\mathsf{frat} = 0.08$ alone results in an improvement of solving times by about 4.5 percent on average on our test cases, irrespective of the number of processors.

Adding the $\mathsf{cmin} = 16$ constraint has a very strong impact, with opposite results according to the class to which the graphs belong. It is extremely beneficial for the 3D meshes, which have incurred relatively low fill-in with the $\mathsf{cmin} = 16$ constraint (see table 7), and for which the solving time is reduced by more then 10 percent on average. On the contrary, it is detrimental for the stiffness matrices, for which the gain in efficiency of the BLAS routines brought by the increase of the average size of the blocks is more than compensated by the increase of the number of operations to perform. Setting automatically the additional cmin criterion for mesh graphs might therefore be of interest.

## 5 Conclusion and perspectives

In this paper, we have presented a tight coupling of the nested dissection and approximate minimum degree algorithms which always improves the quality of the produced orderings.

This work is still in progress, and finding exactly the right criterion for switching between the two methods is still an open question. As a matter of fact, we have evidenced that performing nested dissection too far limits the benefits of the Halo-AMD algorithm. On the opposite, switching too early limits the potentiality of concurrency in the elimination trees.

An other open question is the amount of amalgamation that has to be performed in order to improve solving time. Although this problem is, in its largest extent, solver-dependent, we investigate general criteria applicable to all block-oriented solvers.

## Acknowledgements

The authors would like to thank Pierre Ramet, developer of the PaStiX solver, for the time he spent on evaluating the orderings.

## References

[1] P. Amestoy, T. Davis, and I. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal of Matrix Analysis and Applications*, 17:886–905, 1996.

[2] P. Amestoy, I. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computing Methods in Applied Mechanical Engineering*, 1999. To appear in a special issue on Domain Decomposition and Parallel Computing.

[3] C. Ashcraft. Compressed graphs and the minimum degree algorithm. *SIAM Journal of Scientific Computing*, 16(6):1404–1411, 1995.

[4] C. Ashcraft, S. Eisenstat, J. W.-H. Liu, and A. Sherman. A comparison of three column based distributed sparse factorization schemes. In *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, 1991.

[5] C. Ashcraft and J. W.-H. Liu. Robust ordering of sparse matrices using multisectors. *SIAM Journal of Matrix Analysis and Applications*, 19:816–832, 1998.

[6] P. Charrier and J. Roman. Algorithmique et calculs de complexité pour un solveur de type dissections emboîtées. *Numerische Mathematik*, 55:463–476, 1989.

[7] G. A. Geist and E. G.-Y. Ng. Task scheduling for parallel sparse Cholesky factorization. *International Journal of Parallel Programming*, 18(4):291–314, 1989.

[8] A. George, M. T. Heath, J. W.-H. Liu, and E. G.-Y. Ng. Sparse Cholesky factorization on a local memory multiprocessor. *SIAM Journal of Scientific and Statistical Computing*, 9:327–340, 1988.

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

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

[11] N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer. A comparison of several bandwidth and profile reduction algorithms. *ACM Transactions on Mathematical Software*, 2:322–330, 1976.

[12] A. Gupta, G. Karypis, and V. Kumar. Scalable parallel algorithms for sparse linear systems. In *Proceedings of Stratagem'96, Sophia-Antipolis*, pages 97–110, July 1996.

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

[14] B. Hendrickson and E. Rothberg. Effective sparse matrix ordering: Just around the BEND. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[15] P. Hénon, P. Ramet, and J. Roman. A mapping and scheduling algorithm for parallel sparse fan-in numerical factorization. In *Proceedings of Europar'99, Toulouse*, August 1999. To appear.

[16] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. TR 95-035, University of Minnesota, June 1995.

[17] G. Karypis and V. Kumar. MeTiS – *A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices – Version 4.0*. University of Minnesota, September 1998.

[18] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis*, 16(2):346–358, April 1979.

[19] J. W.-H. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11(2):141–153, 1985.

[20] J. W.-H. Liu. On the minimum degree ordering with constraints. *SIAM Journal of Matrix Analysis and Applications*, 10:1136–1145, 1989.

[21] F. Pellegrini and J. Roman. Sparse matrix ordering with SCOTCH. In *Proceedings of HPCN'97, Vienna, LNCS 1225*, pages 370–378, April 1997.

[22] F. Pellegrini, J. Roman, and P. Amestoy. Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. In *Proceedings of Irregular'99, San Juan, LNCS 1586*, pages 986–995, April 1999.

[23] R. Schreiber. Scalability of sparse direct solvers. Technical Report TR 92.13, RIACS, NASA Ames Research Center, May 1992.

[24] W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *J. Proc. IEEE*, 55:1801–1809, 1967.