RT/APO/08/02

An approximate minimum degree algorithm for matrices with dense rows

P. R. Amestoy[1], H. S. Dollar[2], J. K. Reid[2], J. A. Scott[2]

ABSTRACT

We present a modified version of the approximate minimum degree algorithm for preordering a matrix with a symmetric sparsity pattern prior to the numerical factorization. The modification is designed to improve the efficiency of the algorithm when some of the rows and columns have significantly more entries than the average for the matrix. Numerical results are presented for problems arising from practical applications and comparisons are made with other implementations of variants of the minimum degree algorithm.

Keywords: approximate minimum degree ordering algorithm, quotient graph, sparse matrices, graph algorithms, ordering algorithms.

AMS(MOS) subject classifications: 65F50, 65F05.

---

[1] ENSEEIHT-IRIT UMR 5505, Université de Toulouse, France.
Email: Patrick.Amestoy@enseeiht.fr
Web page: `http://amestoy.perso.enseeiht.fr`
Current report available from `http://apo.enseeiht.fr`.

[2] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK.
Email: s.dollar@rl.ac.uk, j.k.reid@rl.ac.uk & j.a.scott@rl.ac.uk
Current reports also available from `http://www.numerical.rl.ac.uk/reports/reports.html`.
The work of Dollar and Scott was supported by EPSRC grant EP/E053351/1.

ENSEEIHT-IRIT UMR 5505, Université de Toulouse
2 Rue Camichel
31071 Toulouse Cedex 7, France

July 24, 2008

1 Introduction

The efficiency of sparse direct solvers for the solution of linear systems $Ax = b$ with a symmetric sparsity pattern, in terms of both the storage needed and work performed, is dependent upon the order in which the variables are eliminated, that is, the order in which the pivots are selected. Many solvers include a preordering step that aims to use information on the sparsity pattern of A to find a permutation P so that, if the pivots are chosen in order from the diagonal of the permuted matrix, the computed factors are sparser than if the pivots were chosen in order from the diagonal of the original matrix. If A is symmetric and positive definite, the pivot sequence chosen from the sparsity pattern alone can be used by the factorization phase without modification and a Cholesky factorization $PAP^T = LL^T$ can be computed. More generally, numerical pivoting must be incorporated during the factorization phase to maintain numerical stability and, in this case, the pivot sequence computed by the symbolic analyse phase may have to modified.

The problem of finding a permutation P that results in the smallest amount of fill-in for a Cholesky factorization is NP-complete [20] and so heuristics are used to find a good ordering. Two main classes of methods are widely used: those based on nested dissection [11] and those based on the minimum degree algorithm [19]. In recent years, nested dissection has often been found to be the method of choice for many very large problems (typically those of order greater than 50,000 [10]). However, it can be more expensive than the most efficient implementations of the minimum degree algorithm, which is preferred for more modest size problems and for very sparse problems. Many direct solvers typically offer a range of ordering options (see Gould, Scott and Hu [14] for a recent review) that include variants of both the nested dissection and minimum degree algorithms. Currently, the most successful variant of the minimum degree algorithm is the approximate minimum degree algorithm (AMD) and, in particular, the AMD algorithm introduced by Amestoy, Davis and Duff [1] is widely used. A Fortran 77 implementation of AMD is available in the HSL mathematical software library [15] within the package `MC47` (and as Algorithm 837 from ACM Transactions on Mathematical Software [2]). The AMD algorithm is more efficient than classical implementations of the minimum degree algorithm since it uses computationally cheap bounds on the minimum degree in place of the exact minimum degree. Numerical results [1] have shown that, not only is AMD more efficient, it also produces orderings of a similar quality to those obtained using the minimum degree algorithms.

Although AMD is generally very successful, an important exception is when the matrix A has some dense (or almost dense) rows and columns. In this case, the run time for AMD can be high. AMD uses the undirected graph of the matrix and selects each node in turn to have minimum (approximate) degree. Once a node is selected, it is eliminated from the graph and replaced by adding edges between its neighbours so that the neighbours become a clique. If a row is full, the corresponding node will always be adjacent to the eliminated node so its adjacency list has to be scanned and updated, requiring $\mathcal{O}(n^2)$ operations for a problems with n variables. This makes the algorithm prohibitively expensive.

The objective of this paper is to propose a variant of AMD that is designed to overcome this limitation. The rows and columns will be classified initially and then reclassified as the computation proceeds, resulting in an algorithm that computes orderings of comparable quality and is significantly faster when some of the rows and columns have many more entries than the average for the matrix.

This paper is organised as follows. In Section 2, we review the minimum degree and AMD algorithms. The proposed modified AMD algorithm is derived in Section 3 and theoretical aspects considered within Section 4. In Section 5, we describe the use of supervariables within the codes used to generate the numerical results presented in this paper.

The starting point for this work was a modification of the AMD code and algorithm, done by the first author, that has been used and distributed since 2000 within the `MUMPS` package [3, 16]. We note that Duff also included this modified code within Version 1.0.0 of the HSL sparse symmetric solver `MA57` [6]. This earlier work, which has not been documented within the literature, was the starting point for the present paper.

2 The minimum degree and AMD algorithms

The minimum degree and AMD algorithms may be presented using the graph model of Rose [17, 18]. The nonzero pattern of a symmetric n by n matrix A can be represented by an undirected graph $G^0 = \left(V^0, E^0\right)$ with nodes $V^0 = \{1, \ldots, n\}$ and edges E^0. An edge (i, j) is present in E^0 if and only if $a_{ij} \neq 0$ and $i \neq j$; i and j are said to be adjacent.

The elimination graph $G^k = \left(V^k, E^k\right)$ describes the nonzero pattern of the $n^{(k)}$ by $n^{(k)}$ *reduced matrix* $A^{(k)}$ still to be factored after k pivots have been chosen and eliminated. The external degree d_i^k of a node i is defined to be the number of nodes it is adjacent to in the elimination graph G^k. The minimum degree algorithm chooses node p as the kth pivot such that p has minimal external degree in the graph G^{k-1}. For clarity, we will sometimes drop the superscript k during our discussions. Algorithm 1 gives an outline of the minimum degree algorithm. The term *variable* is used to indicate a node that has not been removed from the elimination graph.

Algorithm 1 The minimum degree algorithm

For each $i \in V^0$, set d_i to be the exact external degree of i
$k = 1$
while $k \leq n$ **do**
  Select variable $p \in V^{k-1}$ that minimizes d_p
  Eliminate p
  For each variable i adjacent to p in G^{k-1}, update d_i
  $k = k + 1$
end while

At step k, the graph G^k depends on G^{k-1} and the kth pivot. G^k is found by selecting the kth pivot from V^{k-1}, adding edges to E^{k-1} to make the nodes adjacent to p in G^{k-1} a *clique* (a fully connected subgraph) and then removing p (and its edges) from the graph. The edges added to the graph correspond to fill-in. This addition of edges means that, if G^k is stored explicitly, we cannot know the storage requirements in advance. To remedy this, a quotient graph is used instead of an elimination graph (see Section 2.1). Unfortunately, the calculation of the exact external degree of a variable may then be expensive. Alternatively, the AMD algorithm calculates an upper bound, $\overline{d}_i$, on the exact external degree of a variable and then compares the upper bounds when choosing the next pivot. This generally results in the AMD algorithm being more efficient than the minimum degree algorithm. Numerical results have shown that the AMD algorithm also produces orderings of a similar quality to that of the minimum degree algorithm [1].

Unfortunately, the AMD algorithm can be very inefficient when the matrix A has some dense (or almost dense) rows and columns. When a pivot p is eliminated, the upper bound $\overline{d}_i$ for each i that is adjacent to p must be updated. If i is dense or almost dense, it is (almost) certainly adjacent to p and updating $\overline{d}_i$ will involve a large number of comparisons. This will make the algorithm prohibitively expensive. For clarity, we will refer to this variant of AMD as the *classical AMD algorithm* throughout the remainder of this paper.

In Table 1, we list a subset of the matrices that we used in our tests. These were selected to give a good representation as to how the different algorithms perform on the matrices in the whole of our test set. Each test example is available from the University of Florida Sparse Matrix Collection [5] and, if unsymmetric, has been symmetrized by working with the sum of the given matrix and its transpose. The order n, the number of off-diagonal nonzero entries nz of A, the maximum external degree d_{max}, the average external degree μ, and the standard deviation σ of the external degrees of each (symmetrized) matrix is given. The problems have been listed in increasing order of σ/μ. We distinguish between those with σ/μ less than or greater than one. The first seven problems in Table 1 have comparable numbers of nonzero entries in all

Problem	n	nz	d_{max}	μ	σ	σ/μ
apache1	80800	542184	7	6.71	0.48	0.07
G3_circuit	1585478	7660826	6	4.83	0.64	0.13
ncvqbqp1	50000	349968	9	7.00	2.00	0.29
ford1	18728	101576	20	5.42	1.65	0.30
tuma	22967	87760	5	3.82	1.34	0.35
bcsstk30	28924	2043492	219	70.6	31.7	0.45
inline_1	503712	36816170	843	73.1	35.6	0.49
lpl1	32460	328036	253	10.1	15.5	1.53
gupta3	16783	9323427	14672	556	1234	2.22
mip1	66463	10352819	66395	156	351	2.25
net-4	88343	2441727	4791	27.6	85.3	3.09
ckt11752_dc_1	49702	327834	2921	6.60	24.6	3.73
rajat23	110355	559733	3336	5.07	19.3	3.80
rajat22	39899	198081	3336	4.96	24.8	4.99
gupta2	62064	4248286	8413	68.5	356	5.20
gupta1	31802	2164210	8413	68.1	360	5.29
a0nsdsil	80016	355034	5003	4.44	50.0	11.3
blockqp1	60012	640033	40011	10.7	305	28.6
trans5	116835	827879	114191	7.09	371	52.3
ins2	309412	2751484	309412	8.89	590	66.4

Table 1: Problems in test set: order n, number of nonzeros nz, the maximum external degree d_{max}, the mean μ of the external degrees, and their standard deviation σ.

of the rows of the matrices and for these problems we therefore expect there to be little advantage to be gained by using a modified AMD algorithm instead of the classical AMD algorithm.

We will initially compare the performance of the minimum degree algorithm with that of the MC47 implementation of the classical AMD algorithm. The implementation of the minimum degree algorithm is based on the version found within the HSL sparse solver MA27 [9]. In Table 2, we give the time to obtain the ordering and the forecast number of real words required to hold the matrix factor L formed by the factorization phase of MA57 [6] if no pivoting is performed. Throughout this paper, the latter piece of information is used as a measure for the quality of the ordering and is generated by passing the pivot order to the analyse phase MA57. All numerical results in this paper were obtained using a 3.6 GHz Intel Xeon dual processor Dell Precision 670 with 4 Gbytes of RAM that was running Red Hat Enterprise Linux Server release 5.1 (kernel 2.6.18-53.1.13.el5). The Nag f95 compiler with the -O4 option was used. All times are CPU times in seconds.

As expected, we observe that the classical AMD algorithm is significantly more efficient than the minimum degree algorithm for the problems whose ratio σ/μ is greater than one and the quality of the orderings remains comparable. Observe that the times to obtain the pivot orders differ by at least an order of magnitude for the gupta* problems. The time required to obtain the pivot order for some of the problems is much higher than expected when compared with problems with similar values of n and nz : for example, compare problems bcsstk30 and gupta1. The algorithm proposed in Section 3 aims to substantially reduce these times while maintaining the quality of the ordering.

2.1 Quotient graphs

The quotient graph, also referred to as the generalized element model [7, 8, 9], allows us to model the factorization of A whilst avoiding the need to use more storage than that required to store the original

Problem	Minimum Degree Time	$nz(L)$	Classical AMD Time	$nz(L)$
apache1	0.39	14.8	0.24	13.4
G3_circuit	5.77	238	4.82	193
ncvxbqp1	0.20	5.88	0.19	4.33
ford1	0.03	0.33	0.02	0.31
tuma1	0.02	0.73	0.02	0.58
bcsstk30	0.05	4.71	0.05	3.79
inline_1	2.25	255	1.73	219
lpl1	0.88	0.97	0.19	0.97
gupta3	71.9	5.72	6.93	5.72
mip1	8.44	38.8	6.70	39.0
net4-1	3.65	2.46	1.49	2.38
ckt11752_dc_1	0.34	0.59	0.13	0.56
rajat23	0.43	0.45	0.23	0.46
rajat22	0.21	0.15	0.09	0.15
gupta2	1260	5.86	70.2	5.89
gupta1	239	2.02	23.3	2.06
a0nsdsil	4.95	0.34	2.33	0.34
blockqp1	15.1	0.38	5.27	0.38
trans5	112	0.68	67.6	0.68
ins2	1220	1.53	516	1.53

Table 2: The times and the number of reals in L ($\times 10^6$) for the classical minimum degree algorithm and the classical approximate minimum degree algorithm.

graph G^0 [12]. Using the terminology of the generalized element model, we refer to nodes removed from the elimination graph as *elements* and the uneliminated nodes as *variables*. In the following, we introduce our notation, summarize the properties of the quotient graph representation and recall the AMD approximation. For a detailed description please refer to [1].

The quotient graph $\mathcal{G}^k = \left(V^k, \overline{V}^k, E^k, \overline{E}^k\right)$ implicitly represents the elimination graph, where $\mathcal{G}^0 = G^0$, $V^0 = V$, $\overline{V}^0 = \emptyset$, $E^0 = E$ and $\overline{E}^0 = \emptyset$. The nodes in $\mathcal{G}$ consist of variables (in the set V) and elements (in the set $\overline{V}$). The edges are also divided into two sets: edges between variables $E \subseteq V \times V$ and between variables and elements $\overline{E} \subseteq V \times \overline{V}$. The sets E and $\overline{E}$ are enough to generate the elimination graph and, hence, the edges between elements are not required.

The nodes adjacent to a variable i in the quotient graph $\mathcal{G}$, denoted by $\mathrm{Adj}_{\mathcal{G}}(i)$, may be split into two sets $\mathcal{A}_i$ and $\mathcal{E}_i$ according to whether they are variables or elements:

$$\begin{aligned} \mathcal{A}_i &= \{j : (i,j) \in E\} \subseteq V, \\ \mathcal{E}_i &= \{e : (i,e) \in \overline{E}\} \subseteq \overline{V}. \end{aligned}$$

The set $\mathcal{A}_i^k$ is the subset of nonzero entries in row i of $\mathcal{A}_i^0$ that have not been modified by steps 1 through k of the elimination process. Hence, $\mathcal{A}_i^0 = \{j : a_{ij} \neq 0\}$ and $\mathcal{A}_i^k \subseteq \mathcal{A}_i^{k-1}$ for $1 \leq k \leq n$.

If e is an element, then the variables adjacent to e in $\mathcal{G}$ are denoted by $\mathcal{L}_e$:

$$\mathcal{L}_e \equiv \mathrm{Adj}_{\mathcal{G}}(e) = \{i : (i,e) \in \overline{E}\} \subseteq V.$$

Finally, let

$$\begin{aligned}\mathcal{A} &= \{\mathcal{A}_i : i \in V\},\\ \mathcal{E} &= \{\mathcal{E}_i : i \in V\},\\ \mathcal{L} &= \left\{\mathcal{L}_e : e \in \overline{V}\right\}.\end{aligned}$$

The quotient graph takes no more storage than the original graph since

$$\left|\mathcal{A}^k\right| + \left|\mathcal{E}^k\right| + \left|\mathcal{L}^k\right| \leq \left|\mathcal{A}^0\right|$$

for all k [12]. If i is a variable in G, it is also a variable in $\mathcal{G}$, and

$$\mathrm{Adj}_G(i) = \mathcal{A}_i \cup (\cup_{e\in\mathcal{E}_i}\mathcal{L}_e) \setminus \{i\}, \tag{1}$$

where the "\" is the standard set subtraction operator.

If variable p is selected as a pivot, element p is formed (p is removed from V and added to $\overline{V}$). The set $\mathcal{L}_p$ is then found using equation (1). Since (1) implies that $\mathcal{L}_e \setminus \{p\} \subseteq \mathcal{L}_p$ for all elements e adjacent to p, it must hold that all variables adjacent to an element $e \in \mathcal{E}_p$ are adjacent to the element p and these elements $e \in \mathcal{E}_p$ are no longer needed. They are therefore absorbed into the new element p and deleted: reference to them is replaced by reference to the new element p :

$$\mathcal{E}_i^k = \left(\mathcal{E}_i^{k-1} \cup \{p\}\right) \setminus \mathcal{E}_p^{k-1}.$$

Any entry j in $\mathcal{A}_i^{k-1}$ where both i and j are in $\mathcal{L}_p$ would now have been modified by the elimination process and, hence, needs deleting from $\mathcal{A}_i^{k-1}$:

$$\mathcal{A}_i^k = A_i^{k-1} \setminus \left(\mathcal{L}_p \cup \{p\}\right).$$

Finally, $\mathcal{A}_p$, $\mathcal{E}_p$ and $\mathcal{L}_e$ for all $e \in \mathcal{E}_p$ are deleted.

The external degree d_i^k of a variable i is given by the equation

$$\begin{aligned} d_i^k &= \left|\left(\mathcal{A}_i^k \bigcup_{e\in\mathcal{E}_i} \mathcal{L}_e\right) \setminus \{i\}\right| \\ &= \left|\mathcal{A}_i^k\right| + \left|\bigcup_{e\in\mathcal{E}_i} \mathcal{L}_e \setminus \{i\}\right|. \end{aligned} \tag{2}$$

Now,

$$\begin{aligned} d_i^k &= \left|\mathcal{A}_i^k\right| + \left|\bigcup_{e\in\mathcal{E}_i^k} \mathcal{L}_e \setminus \{i\}\right| \\ &= \left|\mathcal{A}_i^{k-1} \setminus \left(\mathcal{L}_p \cup \{p\}\right)\right| + \left|\left(\bigcup_{e\in\mathcal{E}_i^{k-1}\setminus\mathcal{E}_p^{k-1}} \mathcal{L}_e \setminus \{i\}\right) \cup \left(\mathcal{L}_p \setminus \{i\}\right)\right| \\ &\leq \left|\mathcal{A}_i^{k-1} \setminus \left(\mathcal{L}_p \cup \{p\}\right)\right| + \left|\left(\bigcup_{e\in\mathcal{E}_i^{k-1}\setminus\mathcal{E}_p^{k-1}} \mathcal{L}_e \setminus \{i\}\right)\right| + \left|\mathcal{L}_p \setminus \{i\}\right| \\ &\leq \left|\mathcal{A}_i^{k-1}\right| + \left|\left(\bigcup_{e\in\mathcal{E}_i^{k-1}} \mathcal{L}_e \setminus \{i\}\right)\right| + \left|\mathcal{L}_p \setminus \{i\}\right| \\ &= d_i^{k-1} + \left|\mathcal{L}_p \setminus \{i\}\right|, \end{aligned}$$

and

$$\begin{aligned} d_i^k &= |\mathcal{A}_i^k| + \left| \bigcup_{e \in \mathcal{E}_i^k} \mathcal{L}_e \setminus \{i\} \right| \\ &= |\mathcal{A}_i^k| + \left| (\mathcal{L}_p \setminus \{i\}) \cup \left(\bigcup_{e \in \mathcal{E}_i^k \setminus \{p\}} \mathcal{L}_e \setminus \mathcal{L}_p \right) \right| \\ &= |\mathcal{A}_i^k| + |\mathcal{L}_p \setminus \{i\}| + \left| \bigcup_{e \in \mathcal{E}_i^k \setminus \{p\}} \mathcal{L}_e \setminus \mathcal{L}_p \right| \\ &\leq |\mathcal{A}_i^k| + |\mathcal{L}_p \setminus \{i\}| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus \mathcal{L}_p|. \end{aligned}$$

Clearly $d_i^k \leq n^{(k)} - 1 = n - k - 1$ and, hence, $d_i^k \leq \overline{d}_i^k$ for all k, where the approximate external degree of a variable i is defined by

$$\overline{d}_i^k = \min \begin{cases} n - k - 1 \\ \overline{d}_i^{k-1} + |\mathcal{L}_p \setminus \{i\}| \\ |\mathcal{A}_i \setminus \{i\}| + |\mathcal{L}_p \setminus \{i\}| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus \mathcal{L}_p|. \end{cases} \tag{3}$$

Further theoretical results about the relationship between the exact external degree and the approximate minimum degree (3) may be found in [1].

The classical AMD algorithm with quotient graph notation is given in Algorithm 2.

3 An AMD algorithm for matrices with dense rows

An approximate minimum degree algorithm that takes dense rows into account can be derived using graph partitions. Let $\tau > 0$ be a threshold that we set and $\widehat{d}_i$ be an upper bound to the exact external degree of a variable i. We will partition the matrix into full, quasi dense and sparse parts as follows. A variable in the matrix A is called *full* if we can guarantee that its exact external degree is maximal; a variable i that is not full is called *quasi dense* if $\widehat{d}_i \geq \tau + 1$; finally, a variable is *sparse* if it is neither full or quasi dense. In our new algorithm, we do not update the (approximate) external degree of a quasi dense or full variable, which means that we use a different approximation to the external degree. Our new approximation of the degree, $\widehat{d}_i$, will be derived in this section.

Suppose that there are n_f full variables and n_q quasi dense variables in the reduced matrix $A^{(k)}$: the reduced matrix may then be permuted to take the form

$$A^{(k)} = \left[\begin{array}{c|c|c} A_s & A_{q1}^T & A_{f1}^T \\ \hline A_{q1} & A_q & A_{f2}^T \\ \hline A_{f1} & A_{f2} & A_f \end{array}\right],$$

where the rows and columns of A_s correspond to the sparse variables, the rows and columns of A_q correspond to the quasi dense variables, and the rows and columns of A_f correspond to the full variables.

Consider the case $\tau > n$ and $n_f > 0$. In this case, $n_q = 0$ and $A^{(k)}$ may be permuted to the form

$$A^{(k)} = \left[\begin{array}{c|c} A_s & A_{f1}^T \\ \hline A_{f1} & A_f \end{array}\right].$$

Algorithm 2 The classical AMD algorithm

$V = \{1 \dots n\}$
$\overline{V} = \emptyset$
for $i = 1, \dots, n$ **do**
  $\mathcal{A}_i = \{j : a_{ij} \neq 0 \; and \; i \neq j\}$
  $\mathcal{E}_i = \emptyset$
  $\overline{d}_i^0 = |\mathcal{A}_i|$
end for
$k = 1$
while $k \leq n$ **do**
  select variable $p \in V$ that minimizes $\overline{d}_p^{k-1}$
  $\mathcal{L}_p = \mathcal{A}_p \cup \left(\cup_{e \in \mathcal{E}_p} \mathcal{L}_e\right) \setminus \{p\}$
  for each $i \in \mathcal{L}_p$ **do**
    $\mathcal{A}_i = \mathcal{A}_i \setminus (\mathcal{L}_p \cup \{p\})$
    $\mathcal{E}_i = (\mathcal{E}_i \cup \{p\}) \setminus \mathcal{E}_p$
    $$\overline{d}_i^k = \min \begin{cases} n - k - 1 \\ \overline{d}_i^{k-1} + |\mathcal{L}_p \setminus \{i\}| \\ |\mathcal{A}_i \setminus \{i\}| + |\mathcal{L}_p \setminus \{i\}| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus \mathcal{L}_p| \end{cases}$$
  end for
  $\overline{V} = \left(\overline{V} \cup \{p\}\right) \setminus \mathcal{E}_p$
  $V = V \setminus \{p\}$
  $\mathcal{A}_p = \emptyset$
  $\mathcal{E}_p = \emptyset$
  $k = k + 1$
end while

Observing that a full variable will always remain full within the reduced matrix as the elimination process progresses, our strategy for computing a pivot order is to apply classical AMD to A_s to form a pivot order (with respect to A_s) and to then append the list of full variables to the end of this pivot order. If a variable is detected as becoming full whilst applying classical AMD to A_s, this variable is removed from the reduced matrix and added to the list of full variables. A variable i is detected as full if $|\mathcal{E}_i| \leq 2$ and its approximate minimum degree is maximal (see Section 4 for further details).

Now consider the case where $n_q > 0$ and $n_f \geq 0$. Suppose that we have a (reduced) matrix of the form

$$\begin{bmatrix}
1 & & \times & & & * & \star \\
& 2 & \times & \times & & * & \star \\
\times & \times & 3 & & \times & & \star \\
& \times & & 4 & \times & * & \star \\
& & \times & \times & 5 & * & \star \\
* & * & & * & * & 6 & \star \\
\star & \star & \star & \star & \star & \star & 7
\end{bmatrix},$$

where variable 6 is considered to be quasi dense and variable 7 is full. Eliminating variable 1 from the matrix results in the following fill-in:

$$\left[\begin{array}{c|cccccc}
1 & & \bullet & & & \bullet & \bullet \\
\hline
 & 2 & \times & \times & & * & \star \\
\bullet & \times & 3 & & \times & \square & \star \\
 & \times & & 4 & \times & * & \star \\
 & & \times & \times & 5 & * & \star \\
\bullet & * & \square & * & * & 6 & \star \\
\bullet & \star & \star & \star & \star & \star & 7
\end{array}\right].$$

Observe that variable 6 has become full within the reduced matrix and variable 7 remains full. If the entries in row 1 of the original matrix were to be in different columns, then the pattern of variable 6 might remain quasi dense:

$$\begin{bmatrix}
1 & \times & & & & * & \star \\
\times & 2 & \times & \times & & * & \star \\
 & \times & 3 & & \times & & \star \\
 & \times & & 4 & \times & * & \star \\
 & & \times & \times & 5 & * & \star \\
* & * & & * & * & 6 & \star \\
\star & \star & \star & \star & \star & \star & 7
\end{bmatrix} \rightarrow \left[\begin{array}{c|cccccc}
1 & \bullet & & & & \bullet & \bullet \\
\hline
\bullet & 2 & \times & \times & & * & \star \\
 & \times & 3 & & \times & & \star \\
 & \times & & 4 & \times & * & \star \\
 & & \times & \times & 5 & * & \star \\
\bullet & * & & * & * & 6 & \star \\
\bullet & \star & \star & \star & \star & \star & 7
\end{array}\right].$$

Hence, elimination of a sparse variable results in a quasi dense variable becoming full or remaining quasi dense. As it may be expensive to calculate the new degree of a quasi dense variable, we initially ignore both quasi dense variables and full variables and apply the approximate minimum degree algorithm to A_s. Once each of the variables in A_s has been eliminated or flagged as either full or quasi dense, the algorithm *restarts* by redeclaring each of the quasi dense variables to be sparse, calculating their exact degree within the reduced matrix and applying the algorithm to the reassigned A_s. Finally, when all of the variables in A have been eliminated or flagged as full, the full variables are appended to the end of the pivot sequence.

The adjacency lists $\mathcal{A}_i$ and $\mathcal{E}_i$ of a quasi dense variable i are not updated until a restart is performed. Hence, during a restart, these lists must be reformed: this can be costly. We compute the exact external degree for each variable, which may also be expensive. However, we hope that the cost of restarting will be low relative to the cost of applying classical AMD to our test problems. We note that our variant requires no more workspace than that required by the classical AMD method.

To limit the effect of a bad initial choice of τ and take into account the evolution of the graph structure, τ may be updated at each restart step of the algorithm. Doing so, for example, one can start with a fairly

aggressive initial value of τ (selecting many quasi dense variables) to accelerate the first set of pivot eliminations. Then, during each restart, τ can be reset with a more conservative strategy (in the sense of selecting less quasi dense variables) to preserve the quality and limit the extra cost due to numerous restarts on a large subgraph. We therefore use the notation τ_l to depict the value of τ after l restarts have been performed.

Suppose that F is the set of variables that are currently flagged as full, Q is the set of variables currently flagged as quasi dense, and $S = F \cup Q$, then an upper bound $\widehat{d}_i^k$ for d_i^k is

$$\widehat{d}_i^k \quad = \quad \min \begin{cases} n-k-1 \\ \widehat{d}_i^{k-1} + |\mathcal{L}_p \setminus \{i\}| \\ |S| + |\mathcal{A}_i \setminus (\{i\} \cup S)| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)|. \end{cases} \tag{4}$$

A rigorous derivation of equation (4) and the proof that it is an upper bound for the exact external degree can be found in Section 4. We have derived Algorithm 3 for obtaining a pivot order that should be similar to that generated by the standard AMD algorithm but will avoid the inefficiency problems caused by dense (or almost dense) variables. For the interested reader, Algorithm 4, given in the Appendix, uses the notation introduced in Section 2 to give a more detailed version of Algorithm 3. We discuss the choice of τ_l in Section 3.2.

3.1 Element absorption

In addition to the natural absorption of elements into $\mathcal{E}_p$, in the classical AMD algorithm, any element e satisfying $|\mathcal{L}_e \setminus \mathcal{L}_p| = 0$ is also absorbed into element p, even if e is not adjacent to p. This is commonly referred to as aggressive absorption and improves the degree bounds [1]. We employ this idea in our variant of AMD, but in addition to the condition $|\mathcal{L}_e \setminus \mathcal{L}_p| = 0$ we must now check that element p is adjacent to all of the quasi dense variables. This additional condition is necessary to because the adjacency lists of quasi dense variables are only updated when a restart is performed.

3.2 Choosing the threshold τ_l

The choice of the threshold τ_l in Algorithm 3 will have a significant effect on the speed and efficacy of the algorithm. If τ_l is too large, we may need to recalculate $\widehat{d}_i$ many times for variables that have a large degree: this will be slow. If τ_l is too small, a large number of restarts may be required, which will also be slow. Furthermore, τ_l being too small means that many sparse rows may be incorrectly classified as quasi dense and, hence, some $\widehat{d}_i$ are likely to be severe over-estimates of the corresponding external degrees, resulting in a poor choice of pivot. We compare various strategies for choosing τ_l in this section.

Strategy A

The first strategy that we will consider is to always set

$$\tau_l = n.$$

This value means variables are classed as either sparse or full since $\widehat{d}_i \geq \tau_l + 1$ will never hold and this strategy will result in Algorithm 3 never entering the restart stage. We observed that this strategy performed similarly to classical AMD for all but one of our test problems. For problem `ins2`, strategy A calculated a pivot order in 276 seconds (roughly half the time of classical AMD) with no loss in the quality of the pivot order.

Strategy B

Our second strategy uses the simple choice

$$\tau_l = \sqrt{n-k+1}. \tag{5}$$

Algorithm 3 The revised approximate minimum degree algorithm

For each variable i, set $\widehat{d}_i^0 = d_i^0$ (the exact external degree of variable i)
$k = 1$, $l = 0$
Compute τ_0
Using τ_0, assign each variable to be either sparse, quasi dense or full
while $k \leq n$ **do**
  if any uneliminated sparse variables are present **then**
    Select an uneliminated sparse or quasi dense variable p that minimizes $\widehat{d}_p^{k-1}$
    Eliminate p
    for each sparse i adjacent to p in $A^{(k)}$ **do**
      Use (4) to compute $\widehat{d}_i^k$
      Reassign i to be sparse, quasi dense or full using τ_l
    end for
    $k = k + 1$
  else if any quasi dense variables are present **then**
    Restart:
    $l = l + 1$
    for each quasi dense variable i **do**
      Set $\widehat{d}_i^k = d_i^k$ (the exact external degree of variable i in $A^{(k)}$)
      if $\widehat{d}_i^k$ is maximal **then**
        Assign i to be full
      else
        Assign i to be sparse
      end if
    end for
    Compute τ_l
  else
    Eliminate all full variables.
    $k = n + 1$
  end if
end while

This was motivated by the work of Carmen [4]. We observed that strategy B frequently carries out a large number of restarts for problems that initially satisfy $\sigma > \mu$. As the algorithm progresses, the variables become denser and, hence, we expect the proportion of variables classed as quasi dense to increase if (5) is used. This will, of course, result in a large number of restarts. For example, strategy B calculated a pivot order for problem `mip1` in 2.44 seconds with 188 restarts and $nz(L) = 39.0 \times 10^6$. As expected, the quality of the pivot order may suffer significantly if a large number of restarts is performed. For problem `gupta3`, strategy B calculates a pivot order in 2.46 seconds (roughly a third of the time of classical AMD) with 35 restarts. However, the number of reals in the factor L is more than 7 times greater than that of classical AMD ($nz(L) = 41.3 \times 10^6$).

Strategy C

The simple strategies A and B do not consistently improve upon the classical AMD algorithm and so we now explore more complex strategies. Our third strategy uses the mean of the maximum and minimum exact external degrees of the variables that are not full:

$$\begin{aligned} d_{min} &= \min\left(d_i : \ i \in V \setminus F\right), \\ d_{max} &= \max\left(d_i : \ i \in V \setminus F\right), \qquad (6) \\ \tau_l &= \frac{d_{min} + d_{max}}{2} + 1. \qquad (7) \end{aligned}$$

If d_{max} is much larger than the average external degree, then τ_l should capture the dense variables. Results for strategy C are included in Table 3.

Strategy D

The first author previously developed an AMD variant for matrices with some dense rows and columns that was included within the `MUMPS` package. This variant initialized the threshold to be

$$\tau_l = \alpha\mu + \beta d_{max} + 1, \qquad (8)$$

where μ is the average external degree, d_{max} is defined by (6), and the values $\alpha = 9.9$ and $\beta = 0.1$ where chosen experimentally. Results for strategy D are also included in Table 3.

Comparing the times in Table 3, we observe that, for problems in the lower half of the table, there can be significant savings by using a restarting strategy. Consider, for example, problem `gupta2` which has initial mean external degree 68.5 and standard deviation 356. This problem is notorious because the classical AMD algorithm is extremely inefficient. Strategies C and D are faster by a factor of about 12—14. Applying strategy D to problem `ins2` results in the time dropping by a factor of about 1500. In general, the larger the value σ/μ is, the more dramatic the improvement is for strategy D. This is as we would expect since a problem with a large value of σ/μ will contain some variables that have much larger initial degrees than the average and, hence, will have a set of variables that are considered as dense. For problem `ins2`, strategy C only detects one of the variables that becomes dense and, hence, performs poorly compared to strategy D.

For the test problems satisfying $\sigma \leq \mu$, we observe that strategies C and D can be slower than classical AMD and the quality of the ordering may also suffer. Consider, for example, problem `G3_circuit` that has $\sigma/\mu = 0.13$. The forecast number of reals in L is more than double for strategy C. Furthermore, because of the large number of restarts, strategy C is significantly slower. This is to be expected since if $\sigma \ll \mu$, then

$$\tau_l = \frac{d_{min} + d_{max}}{2} + 1 \approx \mu$$

and, hence, approximately half of the remaining variables will be classified as quasi dense during each restart.

Problem		AMD	C	D	E
apache1	Time/Restarts	0.24	0.58 / 21	0.28 / 3	0.30 / 2
	$nz(L)$	13.4	24.1	13.2	12.9
G3_circuit	Time/Restarts	4.82	9.49 / 22	5.94 / 4	5.80 / 3
	$nz(L)$	193	517	204	196
ncvxbqp1	Time/Restarts	0.19	0.32 / 16	0.21 / 2	0.21 / 2
	$nz(L)$	4.33	9.29	4.70	4.58
ford1	Time/Restarts	0.02	0.03 / 13	0.03 / 2	0.02 / 2
	$nz(L)$	0.31	0.41	0.31	0.31
tuma1	Time/Restarts	0.02	0.02 / 16	0.02 / 2	0.01 / 1
	$nz(L)$	0.58	0.86	0.64	0.61
bcsstk30	Time/Restarts	0.05	0.08 / 11	0.06 / 1	0.07 / 1
	$nz(L)$	3.79	5.77	4.04	3.87
inline_1	Time/Restarts	1.73	3.13 / 11	2.85 / 2	2.56 / 2
	$nz(L)$	219	362	249	253
lpl1	Time/Restarts	0.19	0.15 / 7	0.13 / 2	0.13 / 2
	$nz(L)$	0.974	1.48	1.18	1.20
gupta3	Time/Restarts	6.93	2.53 / 1	2.40 / 1	2.58 / 1
	$nz(L)$	5.72	6.44	6.52	6.52
mip1	Time/Restarts	6.70	1.38 / 1	1.18 / 1	0.57 / 1
	$nz(L)$	39.0	39.1	39.5	39.7
net4-1	Time/Restarts	1.49	0.64 / 2	0.63 / 1	0.56 / 2
	$nz(L)$	2.38	2.45	2.53	3.09
ckt11752_dc_1	Time/Restarts	0.13	0.09 / 1	0.09 / 1	0.09 / 2
	$nz(L)$	0.56	0.57	0.59	0.63
rajat23	Time/Restarts	0.23	0.26 / 1	0.19 / 1	0.16 / 2
	$nz(L)$	0.46	0.46	0.47	0.47
rajat22	Time/Restarts	0.09	0.06 / 1	0.05 / 1	0.05 / 1
	$nz(L)$	0.15	0.15	0.15	0.15
gupta2	Time/Restarts	70.2	5.65 / 1	5.02 / 1	3.98 / 1
	$nz(L)$	5.89	5.95	5.91	5.98
gupta1	Time/Restarts	23.3	2.48 / 3	1.11 / 1	1.89 / 1
	$nz(L)$	2.06	2.06	2.06	2.06
a0nsdsil	Time/Restarts	2.33	0.06 / 2	0.06 / 1	0.06 / 1
	$nz(L)$	0.34	0.35	0.35	0.35
blockqp1	Time/Restarts	5.27	0.03 / 1	0.05 / 1	0.03 / 1
	$nz(L)$	0.38	0.38	0.38	0.38
trans5	Time/Restarts	67.6	1.82 / 1	0.14 / 1	0.14 / 1
	$nz(L)$	0.68	0.68	0.70	0.70
ins2	Time/Restarts	516	246 / 1	0.34 / 1	0.35 / 1
	$nz(L)$	1.53	1.53	1.59	1.59

Table 3: Comparison of the times, the number of restarts and the number of reals in L ($\times 10^6$) for classical AMD and strategies C, D and E.

Strategy E

The relationship between the initial standard deviation of the variable degrees and their mean value appears to make a significant difference to the performance of the various strategies used so far. Strategy D may be interpreted as a crude attempt at taking the mean and standard deviation into account because a large value of d_{max} relative to the mean μ will generally correspond to a large standard deviation σ and vice versa. We observe that if $\sigma > \mu$, strategy D is usually faster than the other strategies. However, it is generally only performing one restart. We may analyse this as follows: if $d_{max} \gg \mu$, then $\tau_l \approx 0.1 d_{max}$ and we are guaranteed to classify some of the variables as quasi dense. After the restart, the difference between d_{max} and μ will, in general, have dramatically fallen and, hence, $\tau_l \approx 10\mu$. This will result in few, if any, variables being classified as quasi dense.

We might expect that performing more than one restart would be beneficial and we would like a threshold τ_l that allows this. Ideally, the problem size and the difference between the standard deviation and mean of the external degrees should all be reflected in the threshold.

If σ is large relative to μ, our experiments showed that a threshold of the form $\alpha\mu + \beta\sigma + 1$, where α and β are positive constants, was effective for detecting quasi dense rows. As the ratio $\sigma/(\mu+1)$ increased, we observed that larger values of β were more desirable and this led us to try a threshold of the form $\alpha\mu + \beta\sigma \left(\sigma/\left(\mu+1\right)\right)^{1.5}$.

If σ is small relative to μ, then the problem has no quasi dense rows and, hence, the threshold should be very large. For $\gamma > 0$, the expression $\alpha\mu + \gamma\mu^2/\left(\sigma+1\right) + 1$ was found to be very effective, in this case, because $\mu^2/\left(\sigma+1\right)$ will be very large. The same value of α was used as in the case where σ is high relative to μ.

Noting that $\sigma\left(\sigma/\left(\mu+1\right)\right)^{1.5}$ will be small when σ is small relative to μ, and $\mu^2/\left(\sigma+1\right)$ will be small when σ is large relative to μ, we can combine our above observations. Therefore, for strategy E we set

$$\tau_l = \alpha\mu + \beta\sigma\left(\sigma/\left(\mu+1\right)\right)^{1.5} + \gamma\mu^2/\left(\sigma+1\right) + 1, \tag{9}$$

where $\alpha = 9$, $\beta = 0.5$ and $\gamma = 2$ are numerically chosen values. Disappointingly, the results in Table 3 show that we have not been able to consistently improve on the performance of strategy D and have only increased the number of restarts for a small number of problems. Strategy E is faster than strategy D for problems `mip1` and `gupta2`, while strategy D is faster for problem `gupta1`. The quality of the pivot orders are similar for all of these problems apart from `net-4`, which is worse.

So far, the same definition for the threshold τ_l has been used on each restart as was used during the initialization stage. In `MUMPS` and Version 1.0.0 of `MA57`, on restart τ_l is defined to be

$$\tau_l = \max\{2 \times \tau_{l-1}, 0.5\left(d_{min} + d_{max}\right) + 1\}. \tag{10}$$

We will call the strategies that use (7), (8), (9) to compute τ_0 and then (10) to compute τ_l $(l = 1, 2, \ldots)$ on each restart strategies C', D' and E' respectively. Their performances are compared in Table 4. The problems with $\sigma \leq \mu$ have been omitted because the results are very similar to those for strategies C, D and E. We observe that there is little difference between the performance of strategies D and D', and E and E'. We therefore propose using classical AMD for problems with $\sigma \leq \mu$ and strategy E for problems with $\sigma > \mu$: we call this strategy QAMD.

4 Accuracy of the approximate degree $\widehat{d}$

As in Section 3, let F be the set of variables that are currently flagged as full, Q be the set of variables currently flagged as quasi dense, and $S = F \cup Q$,. Suppose that $\widetilde{d}_i$ is the exact external degree of variable i with respect to applying the approximate minimum degree algorithm to A_s, where A_s is the submatrix

Problem		C'	D'	E'
lpl1	Time/Restarts	0.14 / 4	0.12 / 4	0.13 / 4
	$nz(L)$	1.53	1.62	1.46
gupta3	Time/Restarts	2.56 / 1	2.42 / 1	2.58 / 1
	$nz(L)$	6.44	6.52	6.52
mip1	Time/Restarts	1.39 / 1	1.18 / 1	0.57 / 2
	$nz(L)$	39.1	39.5	39.8
net4-1	Time/Restarts	0.64 / 1	0.63 / 1	0.56 / 2
	$nz(L)$	2.45	2.53	3.09
ckt11752_dc_1	Time/Restarts	0.10 / 1	0.10 / 1	0.10 / 2
	$nz(L)$	0.57	0.59	0.63
rajat23	Time/Restarts	0.26 / 1	0.19 / 1	0.16 / 2
	$nz(L)$	0.46	0.47	0.47
rajat22	Time/Restarts	0.07 / 1	0.05 / 1	0.04 / 1
	$nz(L)$	0.15	0.15	0.15
gupta2	Time/Restarts	5.65 / 1	5.03 / 1	3.98 / 1
	$nz(L)$	5.95	5.91	5.98
gupta1	Time/Restarts	2.47 / 1	1.11 / 1	1.88 / 1
	$nz(L)$	2.06	2.06	2.06
a0nsdsil	Time/Restarts	0.06 / 1	0.05 / 1	0.06 / 1
	$nz(L)$	0.35	0.35	0.35
blockqp1	Time/Restarts	0.03 / 1	0.06 / 1	0.03 / 1
	$nz(L)$	0.38	0.38	0.38
trans5	Time/Restarts	1.80 / 1	0.14 / 1	0.14 / 1
	$nz(L)$	0.68	0.70	0.70
ins2	Time/Restarts	248 / 1	0.35 / 1	0.34 / 1
	$nz(L)$	1.53	1.59	1.59

Table 4: Comparison of the times, the number of restarts and the number of reals in L ($\times 10^6$) for strategies C', D' and E'.

of A as defined in Section 3. If p is the kth pivot, we obtain the following upper bound for d_i^k :

$$\begin{aligned} d_i^k &\leq |S| + \widetilde{d}_i^k \\ &= |S| + \left|\mathcal{A}_i^k \setminus (\{i\} \cup S)\right| + \left| \left(\bigcup_{e \in \mathcal{E}_i^k} \mathcal{L}_e \right) \setminus (\{i\} \cup S) \right| \\ &= |S| + \left|\mathcal{A}_i^k \setminus (\{i\} \cup S)\right| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \left| \bigcup_{e \in \mathcal{E}_i^k \setminus \{p\}} (\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)) \right| \\ &\leq |S| + \left|\mathcal{A}_i^k \setminus (\{i\} \cup S)\right| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \sum_{e \in \mathcal{E}_i^k \setminus \{p\}} |\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)| \end{aligned}$$

As in Section 2, $d_i^k \leq n - k + 1$ and $d_i^k \leq d_i^{k-1} + |\mathcal{L}_p \setminus \{i\}|$. Hence, $\widehat{d}_i^k$ is an upper bound for the exact degree d_i^k when

$$\widehat{d}_i^k = \min \left\{ \begin{array}{l} n - k - 1 \\ \widehat{d}_i^{k-1} + |\mathcal{L}_p \setminus \{i\}| \\ |S| + \left|\mathcal{A}_i^k \setminus (\{i\} \cup S)\right| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \sum_{e \in \mathcal{E}_i^k \setminus \{p\}} |\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)|. \end{array} \right.$$

In some cases, this bound will be equal to the exact degree or may be a sharper bound than that used in Algorithm 2. In practice, if there are only a small number of dense rows, the values of $\widehat{d}_i$ and $\overline{d}_i$ will be similar:

Theorem 4.1. *Let $S = S_1 \cup S_2$, where $S_1 = S \cap \left(\mathcal{A}_i \cup \left(\bigcup_{e \in \mathcal{E}_i} \mathcal{L}_e\right)\right)$ and $S_2 = S \setminus \left(\mathcal{A}_i \cup \left(\bigcup_{e \in \mathcal{E}_i} \mathcal{L}_e\right)\right)$. The following inequalities hold:*

- *$d_i = \widehat{d}_i = \overline{d}_i$ holds for all variables $i \notin S$ when $S_2 = \emptyset$ and $|\mathcal{E}_i| \leq 2$;*
- *$d_i + |S_2| \leq \widehat{d}_i \leq \overline{d}_i + |S_2|$ holds for all variables $i \notin S$.*

Proof. Rearranging the expression for $\widehat{d}_i$, where $i \notin S$, and using the fact that $\mathcal{A}_i \cap \mathcal{L}_e = \emptyset$ for all elements $e \in \mathcal{E}_i$, we obtain

$$\begin{aligned}
\widehat{d}_i &= |S| + |\mathcal{A}_i \setminus (\{i\} \cup S)| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)| \\
&= |\mathcal{A}_i \setminus \{i\}| + |S \setminus \mathcal{A}_i| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)| \\
&= |\mathcal{A}_i \setminus \{i\}| + |S \setminus \mathcal{A}_i| + |\mathcal{L}_p \setminus \{i\}| - |\mathcal{L}_p \cap S| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} (|\mathcal{L}_e \setminus \mathcal{L}_p| - |(\mathcal{L}_e \setminus \mathcal{L}_p) \cap S|) \\
&= |\mathcal{A}_i \setminus \{i\}| + |\mathcal{L}_p \setminus \{i\}| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus \mathcal{L}_p| + |S \setminus (\mathcal{A}_i \cup \mathcal{L}_p)| - \sum_{e \in \mathcal{E}_i \setminus \{p\}} |(\mathcal{L}_e \setminus \mathcal{L}_p) \cap S| \\
&= |\mathcal{A}_i \setminus \{i\}| + |\mathcal{L}_p \setminus \{i\}| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus \mathcal{L}_p| + |S_2| + |S_1 \setminus (\mathcal{A}_i \cup \mathcal{L}_p)| - \sum_{e \in \mathcal{E}_i \setminus \{p\}} |(\mathcal{L}_e \setminus \mathcal{L}_p) \cap S_1| \,.
\end{aligned}$$

If $S_2 = \emptyset$ and $|\mathcal{E}_i| \leq 1$, clearly $|S_2| = 0$ and

$$|S_1 \setminus (\mathcal{A}_i \cup \mathcal{L}_p)| - \sum_{e \in \mathcal{E}_i \setminus \{p\}} |(\mathcal{L}_e \setminus \mathcal{L}_p) \cap S_1| = |S_1 \setminus (\mathcal{A}_i \cup \mathcal{L}_p)| = 0.$$

If $S_2 = \emptyset$ and $|\mathcal{E}_i| = 2$ (the current element p and a prior element e, say), $|S_2| = 0$ and

$$\begin{aligned}
S_1 &= (\mathcal{A}_i \cup \mathcal{L}_p \cup (\mathcal{L}_e \setminus \mathcal{L}_p)) \cap S_1 \\
S_1 \setminus (\mathcal{A}_i \cup \mathcal{L}_p) &= (\mathcal{L}_e \setminus \mathcal{L}_p) \cap S_1.
\end{aligned}$$

In both cases we observe that $\widehat{d}_i = \overline{d}_i$ and, hence, $d_i = \widehat{d}_i = \overline{d}_i$ [1, Theorem 4.1].

Considering the general case we obtain

$$\begin{aligned}
S_1 &= \left(\mathcal{A}_i \cup \mathcal{L}_p \cup \left(\bigcup_{e \in \mathcal{E}_i \setminus \{p\}} (\mathcal{L}_e \setminus \mathcal{L}_p)\right)\right) \cap S_1 \\
S_1 \setminus (\mathcal{A}_i \cup \mathcal{L}_p) &= \bigcup_{e \in \mathcal{E}_i \setminus \{p\}} ((\mathcal{L}_e \setminus \mathcal{L}_p) \cap S_1) \\
|S_1 \setminus (\mathcal{A}_i \cup \mathcal{L}_p)| &\leq \sum_{e \in \mathcal{E}_i \setminus \{p\}} |(\mathcal{L}_e \setminus \mathcal{L}_p) \cap S_1| \,.
\end{aligned}$$

Hence, $\widehat{d}_i \leq \overline{d}_i + |S_2|$.

Again, using the fact that $\mathcal{A}_i \cap \mathcal{L}_e = \emptyset$ for all elements $e \in \mathcal{E}_i$, we can rearrange the expression for d_i as

$$\begin{aligned}
d_i &= |\mathcal{A}_i \setminus \{i\}| + \left| \left(\bigcup_{e \in \mathcal{E}_i} \mathcal{L}_e \right) \setminus \{i\} \right| \\
&= |\mathcal{A}_i \setminus \{i\}| + |\mathcal{L}_p \setminus \{i\}| + \left| \bigcup_{e \in \mathcal{E}_i \setminus \{p\}} (\mathcal{L}_e \setminus \mathcal{L}_p) \right| \\
&= |\mathcal{A}_i \cap S| + |(\mathcal{L}_p \setminus \{i\}) \cap S| + \left| \bigcup_{e \in \mathcal{E}_i \setminus \{p\}} ((\mathcal{L}_e \setminus \mathcal{L}_p) \cap S) \right| \\
&\quad + |\mathcal{A}_i \setminus (\{i\} \cup S)| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \left| \bigcup_{e \in \mathcal{E}_i \setminus \{p\}} (\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)) \right| \\
&= |S_1| + |\mathcal{A}_i \setminus (\{i\} \cup S)| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \left| \bigcup_{e \in \mathcal{E}_i \setminus \{p\}} (\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)) \right| \\
&\leq |S| - |S_2| + |\mathcal{A}_i \setminus (\{i\} \cup S)| + |\mathcal{L}_p \setminus (\{i\} \cup S)| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |(\mathcal{L}_e \setminus (\mathcal{L}_p \cup S))| \\
&= \widehat{d_i} - |S_2| \,.
\end{aligned}$$

This completes the proof. □

The first statement in Theorem 4.1 may be used to test whether a variable is full or not. Unfortunately, checking whether $S_2 = \emptyset$ holds for a variable i may be as expensive as calculating the exact degree of the variable. If $S = F$, then we can guarantee that $S_2 = \emptyset$ and, hence, we use $|S| = |F|$ within the test for a full variable rather than $S_2 = \emptyset$.

5 Indistinguishable variables and mass elimination

In the analysis presented in this paper, we have considered each variable separately. However, in the codes used to generate the numerical results we have taken full advantage of the possibility of two variables having the same sparsity pattern in the reduced matrix $A^{(k)}$. Two variables i and j are called *indistinguishable* if they are adjacent and also have identical adjacency lists in the elimination graph corresponding to $A^{(k)}$. Working with a quotient graph means that it is efficient to check whether two sparse variables are indistinguishable [1]. If any other variable is eliminated, then the indistinguishable variables will remain indistinguishable: they will have the same degree until one is selected as a pivot. If i is selected as the current pivot, then j can be selected next without causing any additional fill-in.

We may merge i and j into a *supervariable* containing both i and j, labeled by its *principal* variable (i, say) [7, 9]. Variables that are not principal variables are called *simple* variables. We denote the set of simple variables in a supervariable with principal i as $\mathbf{i}$ and define $\mathbf{i} = \{i\}$ if i is the only variable in the supervariable. When p is selected as a pivot at the k th step, all variables in $\mathbf{p}$ are eliminated. The number of degree calculations performed is, in general, greatly reduced when supervariables are used. Non-principal variables and their incident edges are removed from the quotient graph structure when they are detected. The set notation $\mathcal{A}$ and $\mathcal{L}$ refers to either a set of supervariables or the variables represented by supervariables, depending on the context.

It is possible that some of the variables that are adjacent to a pivot can be eliminated at the same time without causing additional fill. This is known as *mass elimination* [13]. Unfortunately, we cannot test for this unless the pivot is adjacent to all of the quasi dense variables since the adjacency lists of quasi dense variables are only updated when a restart is performed.

Using supervariables, the exact external degree d_i^k, the approximate external degree $\overline{d}_i^k$, and the approximate external degree using dense variables $\widehat{d}_i^k$ are defined by

$$
\begin{aligned}
d_i^k &= |\mathcal{A}_i^k \setminus \mathbf{i}| + \left| \left(\bigcup_{e \in \mathcal{E}_i^k} \mathcal{L}_e \right) \setminus \mathbf{i} \right| \\
\overline{d}_i^k &= \min \begin{cases} n-k-1 \\ \overline{d}_i^{k-1} + |\mathcal{L}_p \setminus \mathbf{i}| \\ |\mathcal{A}_i^k \setminus \mathbf{i}| + |\mathcal{L}_p \setminus \mathbf{i}| + \sum_{e \in \mathcal{E}_i^k \setminus \{p\}} |\mathcal{L}_e \setminus \mathcal{L}_p| \end{cases} \\
\widehat{d}_i^k &= \min \begin{cases} n-k-1 \\ \widehat{d}_i^{k-1} + |\mathcal{L}_p \setminus \mathbf{i}| \\ |S| + \left|\mathcal{A}_i^k \setminus (S \cup \mathbf{i})\right| + |\mathcal{L}_p \setminus (\mathbf{i} \cup S)| + \sum_{e \in \mathcal{E}_i^k \setminus \{p\}} |\mathcal{L}_e \setminus (\mathcal{L}_p \cup S)| \end{cases}
\end{aligned}
$$

In the Appendix, we present a supervariable version of the revised AMD algorithm (Algorithm 5).

6 Conclusions

We have described a new variant of the approximate minimum degree algorithm that aims to efficiently detect and deal with the existence of any dense variable throughout the formation of a pivot order. This work was based on an initial code implementing strategy D' of the algorithm and distributed with the MUMPS package since 2000 [3, 16]. Theoretical results relating the new upper bound for the degrees of the variables to that of Amestoy, Davis and Duff [1] and the exact degree have been provided. They show that the two bounds will be similar if there is only a small number of quasi dense rows.

We have experimentally demonstrated that using a restarting strategy for problems with $\sigma > \mu$ (that is problems with some dense or near dense rows) can make significant savings on the CPU time required to obtain the pivot order: the quality of the ordering is maintained. Strategy QAMD has been implemented within the HSL package `MC47` (version 2.1.0). Version 2.1.0 of `MC47` is available now as part of the 2007 release of the mathematical software library HSL. All use of HSL requires a licence; details of how to obtain a licence and the packages are available at `www.cse.clrc.ac.uk/nag/hsl/`.

References

[1] P. Amestoy, T. Davis, and I. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] ——, *Algorithm 837: AMD, an approximate minimum degree ordering algorithm*, ACM Transactions on Mathematical Software, 30 (2004), pp. 381–388.

[3] P. Amestoy, I. Duff, and J. L'Excellent, *Multifrontal parallel distributed symmetric and unsymmetric solvers*, Comput. Methods in Appl. Mech. Eng, 184 (2000), pp. 501–520.

[4] J. L. Carmen, *An AMD preprocessor for matrices with some dense rows and columns.* http://www.netlib.org/linalg/amd/amdpre.ps.

[5] T. Davis, *The University of Florida Sparse Matrix Collection*, 2007. http://www.cise.ufl.edu/davis/techreports/matrices.pdf.

[6] I. Duff, *MA57– a new code for the solution of sparse symmetric definite and indefinite systems*, ACM Transactions on Mathematical Software, 30 (2004), pp. 118–154.

[7] I. Duff and J. Reid, *A comparison of sparsity orderings for obtaining a pivotal sequence in Gaussian elimination*, J. Inst. Math. Appl., 14 (1974), pp. 281–291.

[8] ——, *MA27 - A set of Fortran subroutines for solving sparse symmetric sets of linear equations*, Tech. Rep. AERE R-10533, HMSO, London, 1982.

[9] ——, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Transactions on Mathematical Software, 9 (1983), pp. 302–325.

[10] I. Duff and J. Scott, *Towards an automatic ordering for a symmetric sparse direct solver*, Tech. Rep. RAL-TR-2006-001, Rutherford Appleton Laboratory, 2006.

[11] A. George, *Nested dissection of a regular finite-element mesh*, SIAM J. Num. Anal., 10 (1973), pp. 345–363.

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

[13] A. George and D. R. McIntyre, *On the application of the minimum degree algorithm to finite element systems*, SIAM J. Numer. Anal., 15 (1978), pp. 90–112.

[14] N. Gould, J. Scott, and Y. Hu, *A numerical evaluation of sparse direct solvers for the solution of large, sparse, symmetric linear systems of equations*, ACM Transactions on Mathematical Software, (2007).

[15] HSL, *A collection of Fortran codes for large-scale scientific computation.* See `http://hsl.rl.ac.uk/`, 2007.

[16] MUMPS, *Multifrontal Massively Parallel Solver.* See `http://mumps.enseiht.fr` or `http://graal.ens-lyon.fr/MUMPS/`.

[17] D. J. Rose, *Symmetric elimination on sparse positive definite systems and the potential flow network problem*, PhD thesis, Harvard University, 1970.

[18] ——, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, 1973, pp. 183–217.

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

[20] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Discrete Meth., 2 (1981), pp. 77–79.

A Appendix

Algorithm 4 The revised approximate minimum degree algorithm

$V = \{1 \dots n\}$, $\overline{V} = \emptyset$
$F = \emptyset$, $Q = \emptyset$
Set $\tau_0 \leq n + 1$
for $i = 1, \dots, n$ **do**
  $\mathcal{A}_i = \{j : a_{ij} \neq 0 \text{ and } i \neq j\}$
  $\mathcal{E}_i = \emptyset$
  $\widehat{d}_i^0 = |\mathcal{A}_i|$
  if $\widehat{d}_i^0 = n - 1$ **then**
    $F = F \cup \{i\}$
  else if $\widehat{d}_i^0 \geq \tau_0 + 1$ **then**
    $Q = Q \cup \{i\}$
  end if
end for
$k = 1$, $l = 0$
while $k \leq n$ **do**
  if $V \setminus (F \cup Q) \neq \emptyset$ **then**
    select variable $p \in V \setminus (F \cup Q)$ that minimizes $\widehat{d}_p^{k-1}$
    $\mathcal{L}_p = \mathcal{A}_p \cup \left(\cup_{e \in \mathcal{E}_p} \mathcal{L}_e\right) \setminus \{p\}$
    for each $i \in \mathcal{L}_p \setminus (F \cup Q)$ **do**
      $\mathcal{A}_i = \mathcal{A}_i \setminus (\mathcal{L}_p \cup \{p\})$
      $\mathcal{E}_i = (\mathcal{E}_i \cup \{p\}) \setminus \mathcal{E}_p$
      $\widehat{d}_i^k = \min \begin{cases} n - k - 1 \\ \widehat{d}_i^{k-1} + |\mathcal{L}_p \setminus \{i\}| \\ |F \cup Q| + |\mathcal{A}_i \setminus (F \cup Q \cup \{i\})| + |\mathcal{L}_p \setminus (F \cup Q \cup \{i\})| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus (F \cup Q \cup \mathcal{L}_p)| \end{cases}$
      if $\left(\widehat{d}_i^k + 1 = n - k\right)$ and $|\mathcal{E}_i| \leq 2$ and $Q = \emptyset$ **then**
        $F = F \cup \{i\}$
      else if $\widehat{d}_i^k \geq \tau_l + 1$ **then**
        $Q = Q \cup \{i\}$
      end if
    end for
    $\overline{V} = \left(\overline{V} \cup \{p\}\right) \setminus \mathcal{E}_p$, $V = V \setminus \{p\}$
    $\mathcal{A}_p = \emptyset$, $\mathcal{E}_p = \emptyset$
    $k = k + 1$
  else if $Q \neq \emptyset$ **then**
    Restart: $l = l + 1$
    for each $i \in Q$ **do**
      $\mathcal{A}_i = \{j : a_{ij} \neq 0 \text{ and } j \in V \text{ and } i \neq j\}$
      $\mathcal{E}_i = \left\{j : a_{ij} \neq 0 \text{ and } j \in \overline{V} \text{ and } i \neq j\right\}$
      $\widehat{d}_i^{k-1} = |\mathcal{A}_i \setminus \{i\}| + \left|\left(\bigcup_{e \in \mathcal{E}_i} \mathcal{L}_e\right) \setminus \{i\}\right|$
      $Q = Q \setminus \{i\}$
      if $\widehat{d}_i^{k-1} = n - k$ **then**
        $F = F \cup \{i\}$
      end if
    end for
    Compute τ_l
  else
    for each $i \in F$ **do**
      $\overline{V} = \overline{V} \cup \{i\}$, $V = V \setminus \{i\}$
      $k = k + 1$
    end for
  end if
end while

Algorithm 5 The approximate minimum degree algorithm with supervariables

$V = \{1 \dots n\}$, $\overline{V} = \emptyset$, $F = \emptyset$, $Q = \emptyset$
Set $\tau_0 < n$
for $i = 1, \dots, n$ **do**
  $\mathcal{A}_i = \{j : a_{ij} \neq 0 \text{ and } i \neq j\}$, $\mathcal{E}_i = \emptyset$, $\mathbf{i} = \{i\}$, $\widehat{d}_i^0 = |\mathcal{A}_i|$
  if $\widehat{d}_i^0 = n - 1$ **then**
    $F = F \cup \mathbf{i}$
  else if $\widehat{d}_i^0 + \tau_0 \geq n - 1$ **then**
    $Q = Q \cup \mathbf{i}$
  end if
end for
$k = 1$, $l = 0$
while $k \leq n$ **do**
  if $V \setminus (F \cup Q) \neq \emptyset$ **then**
    select variable $p \in V \setminus (F \cup Q)$ that minimizes $\widehat{d}_p^{k-1}$
    $\mathcal{L}_p = \mathcal{A}_p \cup \left(\cup_{e \in \mathcal{E}_p} \mathcal{L}_e\right) \setminus \mathbf{p}$
    for each $\mathbf{i} \in \mathcal{L}_p \setminus (F \cup Q)$ **do**
      $\mathcal{A}_i = \mathcal{A}_i \setminus (\mathcal{L}_p \cup \mathbf{p})$, $\mathcal{E}_i = (\mathcal{E}_i \cup \{p\}) \setminus \mathcal{E}_p$
      $\widehat{d}_i^k = \min \begin{cases} n - k - |\mathbf{p}| \\ \widehat{d}_i^{k-1} + |\mathcal{L}_p \setminus \mathbf{i}| \\ |F \cup Q| + |\mathcal{A}_i \setminus (F \cup Q \cup \mathbf{i})| + |\mathcal{L}_p \setminus (F \cup Q \cup \mathbf{i})| + \sum_{e \in \mathcal{E}_i \setminus \{p\}} |\mathcal{L}_e \setminus (F \cup Q \cup \mathcal{L}_p)| \end{cases}$
      if $\left(\widehat{d}_i^k + |\mathbf{i}| = n - k\right)$ and $|\mathcal{E}_i| \leq 2$ and $Q = \emptyset$ **then**
        $F = F \cup \mathbf{i}$
      else if $\widehat{d}_i^k + |\mathbf{i}| \geq \tau_l$ **then**
        $Q = Q \cup \mathbf{i}$
      end if
    end for
    supervariable detection
    for each pair $\mathbf{i}$ and $\mathbf{j} \in \mathcal{L}_p \setminus (F \cup Q)$ **do**
      if $\mathbf{i}$ and $\mathbf{j}$ are indistinguishable **then**
        $\mathbf{i} = \mathbf{i} \cup \mathbf{j}$, $\widehat{d}_i^k = \widehat{d}_i^k - |\mathbf{j}|$, $V = V \setminus \{j\}$, $\mathcal{A}_j = \emptyset$, $\mathcal{E}_j = \emptyset$
      end if
    end for
    $\overline{V} = \left(\overline{V} \cup \{p\}\right) \setminus \mathcal{E}_p$, $V = V \setminus \{p\}$, $\mathcal{A}_p = \emptyset$, $\mathcal{E}_p = \emptyset$, $k = k + |\mathbf{p}|$
  else if $Q \neq \emptyset$ **then**
    Restart: $l = l + 1$
    for each $\mathbf{i} \in Q$ **do**
      $\mathcal{A}_i = \{j : a_{ij} \neq 0 \text{ and } j \in V \text{ and } i \neq j\}$
      $\mathcal{E}_i = \left\{j : a_{ij} \neq 0 \text{ and } j \in \overline{V} \text{ and } i \neq j\right\}$
      $\widehat{d}_i^{k-1} = |\mathcal{A}_i \setminus \mathbf{i}| + \left|\left(\bigcup_{e \in \mathcal{E}_i} \mathcal{L}_e\right) \setminus \mathbf{i}\right|$
      $Q = Q \setminus \mathbf{i}$
      if $\widehat{d}_i^{k-1} + |\mathbf{i}| = n - k + 1$ **then**
        $F = F \cup \mathbf{i}$
      end if
    end for
    Compute τ_l
  else
    for each $i \in F$ **do**
      $\overline{V} = \overline{V} \cup \{i\}$, $V = V \setminus \{i\}$, $k = k + |\mathbf{i}|$
    end for
  end if
end while