ELIMINATION STRUCTURES FOR UNSYMMETRIC SPARSE LU FACTORS

JOHN R. GILBERT* AND JOSEPH W. H. LIU†

Abstract. The elimination tree is central to the study of Cholesky factorization of sparse symmetric positive definite matrices. In this paper, we generalize the elimination tree to a structure appropriate for the sparse LU factorization of unsymmetric matrices. We define a pair of directed acyclic graphs called *elimination dags*, and use them to characterize the zero-nonzero structures of the lower and upper triangular factors. We apply these elimination structures in a new algorithm to compute fill for sparse LU factorization. Our experimental results indicate that the new algorithm is usually faster than earlier methods.

Key words. sparse matrix algorithms, Gaussian elimination, LU factorization, elimination tree, elimination dag.

AMS(MOS) subject classifications. 05C20, 05C75, 65F05, 65F50.

1. Introduction. The *elimination tree* [10, 14] is central to the study of symmetric factorization of sparse positive definite matrices. Liu [11] surveys the use of this tree structure in many aspects of sparse Cholesky factorization, including sparse storage schemes, matrix reordering, symbolic and numerical factorization algorithms, and modeling parallel elimination. In particular, the row and column structures of the Cholesky factor of a symmetric positive definite sparse matrix can be characterized in terms of its elimination tree.

The objective of this paper is to generalize the elimination tree to a structure that can be used to study sparse unsymmetric LU factorization. We seek a structure that characterizes the lower and upper triangular factors in the same way that elimination trees characterize Cholesky factors. Moreover, for the special case when the matrix is symmetric, the generalized structure should be the elimination tree. Our generalization consists of a pair of special directed acyclic graphs (dags), which we call the *elimination dags*, or *edags* for short.

The outline of the paper is as follows. In Section 2, we provide the background material to study sparse LU factorization. We formulate numerical factorization as a sequence of sparse triangular solves, thus motivating the study of the solution of sparse triangular systems. We introduce the relevant graph notions, and quote results from the literature that relate the structure of the solution vector to the structures of the triangular matrix and the right-hand side vector.

In Section 3, we define elimination dags. They are simply the smallest structures that preserve the set of paths in the graphs of the lower and upper triangular factor matrices. In graph-theoretic terms, the edags are *transitive reductions* [1] of the graphs of L and U. We show that if the matrix is symmetric, elimination dags are simply elimination trees. We prove some graph-theoretic results about the path structure of the graph of a matrix, its triangular factors, and its elimination dags.

Section 4 contains the main results, which characterize the structures of sparse LU factors in terms of elimination dags. We demonstrate that elimination dags are

* Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304.

† Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3.
Research was supported in part by the Canadian Natural Sciences and Engineering Research Council under grant A5509.

truly analogs of elimination trees, by showing that the symmetric structure theory of elimination trees can be obtained as special cases of the elimination dag results.

In Section 5, we apply elimination dags to sparse symbolic LU factorization. We briefly review two algorithms FILL1 and FILL2 by Rose and Tarjan [12] that compute fill for sparse unsymmetric matrices. We then formulate a new symbolic factorization scheme based on the pair of elimination dags and compare its performance with that of FILL1 and FILL2. Our experimental results show that the new algorithm performs much better than either on practical sparse problems.

Section 6 contains our concluding remarks. We discuss extensions to these ideas, including the possibility of using elimination dags in numerical sparse LU factorization schemes with partial pivoting.

2. Background.

2.1. LU factorization in terms of triangular solutions. Suppose A is an unsymmetric sparse $n \times n$ matrix that has a factorization $A = LU$ without pivoting. The *bordering method* allows us to compute the factorization as a sequence of triangular solves. Specifically, write

$$A = \begin{pmatrix} A' & h \\ g^T & s \end{pmatrix},$$

where A' is the $(n-1) \times (n-1)$ leading principal submatrix, g and h are $(n-1)$-vectors, and s is a scalar. If A' has been factored as $L'U'$, the LU factorization of A is

$$A = \begin{pmatrix} A' & h \\ g^T & s \end{pmatrix} = \begin{pmatrix} L' & \\ \bar{g}^T & \bar{s} \end{pmatrix} \begin{pmatrix} U' & \bar{h} \\ & 1 \end{pmatrix},$$

where $\bar{g} = {U'}^{-T} g$, $\bar{h} = {L'}^{-1} h$, and $\bar{s} = s - \bar{g}^T \bar{h}$.

Note that the vectors $\bar{g}$ and $\bar{h}$ can be obtained as the solutions of two lower triangular systems, each of size $n-1$:

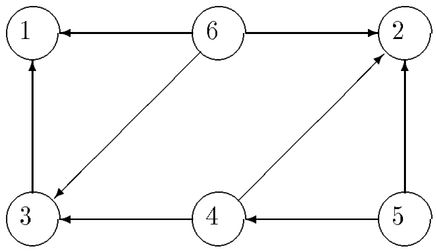
$$L'\bar{h} = h, \qquad {U'}^T \bar{g} = g.$$

The matrix A' can be factored into $L'U'$ in the same way recursively. Thus we can view the entire factorization as a sequence of n pairs of triangular solves of sizes from 0 to $n-1$. Moreover, the structure of the vector $\bar{h}$ depends on that of h and L', and the structure of $\bar{g}$ depends on that of g and U'. This motivates the following study relating the structure of the solution vector with that of the given lower triangular matrix and right hand side.

2.2. Sparse triangular matrices and directed acyclic graphs. Let $A = (a_{ij})$ be a sparse $n \times n$ matrix with nonzero diagonal entries. Define $G(A)$ to be the directed graph of the matrix A as follows. The vertex set of $G(A)$ is $V = \{1, 2, \ldots, n\}$; there is an edge from i to j (for $i \neq j$) if and only if the entry a_{ij} is nonzero. We shall write the directed edge from i to j as $\langle i, j \rangle$. Notice that the edges $\langle i, j \rangle$ and $\langle j, i \rangle$ are different; the former means that $a_{ij} \neq 0$ and the latter means that $a_{ji} \neq 0$.

Many sparse matrix codes [6, 9] use a data structure that lists the nonzeros of a matrix A in column major order. In graph terms, this is the "adjacency list" structure for $G(A^T)$. Representing A by $G(A^T)$ instead of by $G(A)$ seems to be necessary for some efficient algorithms [9]. This is the reason that this paper sometimes states results and algorithms in terms of the graph of the transpose of the matrix in question.

$$L = \begin{pmatrix} 1 & & & & & \\ & 2 & & & & \\ \bullet & & 3 & & & \\ & \bullet & \bullet & 4 & & \\ & \bullet & & \bullet & 5 & \\ \bullet & \bullet & \bullet & & & 6 \end{pmatrix}$$



$G(L)$

FIG. 1. *A lower triangular matrix L and its directed graph $G(L)$.*

The directed graph of a triangular matrix has a special structure. Consider a lower triangular matrix $L = (\ell_{ij})$ and its directed graph $G(L)$. A nonzero off-diagonal entry ℓ_{ij} must have $i > j$, so any directed edge $\langle i, j \rangle$ of $G(L)$ must satisfy $i > j$. Since a directed edge always points from a vertex with a higher subscript to one with a lower subscript, the graph $G(L)$ must be *acyclic*: it cannot have any directed cycles. An acyclic directed graph is often called a *dag* for short [2]. Figure 1 is an example of a lower triangular matrix and its corresponding directed acyclic graph.

2.3. Structural characterization of triangular solution. Consider the solution of the lower triangular system $Lx = b$, where both L and the right-hand side vector b are sparse. Gilbert [7] provides a simple characterization of the sparse structure of the solution vector x in terms of that of L and b. We introduce the following notation: For any row or column vector $w = (w_1, \ldots, w_n)$ or $w = (w_1, \ldots, w_n)^T$, define its *vector structure* as the vertex subset

$$\mathrm{Struct}(w) = \{i \in V \mid w_i \neq 0\}.$$

Note that in this definition, i is a vertex in the graph $G(L)$ and w_i is an entry in the vector w.

THEOREM 2.1. **[7]** *The structure* Struct(x) *of the solution vector x to $Lx = b$ is given by the set of vertices reachable from vertices of the right-hand side structure* Struct(b) *by paths in the directed graph $G(L^T)$.* □

Strictly speaking, this result gives only an upper bound on the structure of x, because coincidental cancellation might produce more zeros. This is the tightest upper bound on Struct(x) that is possible given only the nonzero structures of L and b. To avoid cluttering up theorem statements in the rest of the paper, we will not keep mentioning the possibility of coincidental cancellation. Thus when we make a statement of the form "The structure of X is Y," it should be understood to mean all of the following: "If the nonzero values of A are chosen independently at random, then the structure of X is Y with probability one"; "If the nonzero values of A are algebraically independent, then the structure of X is Y"; and "Regardless of the nonzero values of A, the structure of X is a subset of Y." See Gilbert [7] for a detailed discussion of this point.

To illustrate Theorem 2.1, consider the solution of $Lx = b$ for the matrix L of Figure 1. The graph $G(L^T)$ is the graph $G(L)$ with all its edges reversed. If b_1 is the only nonzero in the right hand side vector b, then its structure vertex set is Struct$(b) = \{1\}$. The vertices that are reachable from vertex 1 through directed paths in $G(L^T)$ are those in the set $\{1, 3, 4, 5, 6\}$. On the other hand, if Struct$(b) = \{2, 4\}$,

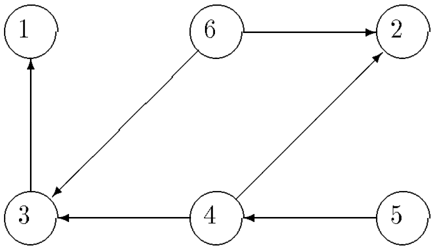


FIG. 2. *The transitive reduction of the graph in Figure 1.*

then the solution structure Struct(x) is $\{2, 4, 5, 6\}$.

Theorem 2.1 is actually true (including the remarks about coincidental cancellation) for any nonsingular matrix with nonzero diagonal, whether triangular or not. However we will only use the triangular case.

3. Elimination dags for sparse LU factorization.

3.1. Transitive reduction of a directed graph. Theorem 2.1 relates the structure of the solution vector x to information about paths in the directed graph $G(L)$. One economical way to represent path information for a directed graph is by its *transitive reduction* [1]. The idea is to find another directed graph with fewer edges than the given graph $G(L)$, but with the same path structure. In such a graph, fewer edges would need to be traversed to generate Struct(x) for a given Struct(b).

A graph G° is a *transitive reduction* of a given directed graph G if G° satisfies the following two conditions:

(a) G° has a directed path from u to v if and only if G has a directed path from u to v.
(b) No graph with fewer edges than G° satisfies condition (a).

An arbitrary graph may have many different transitive reductions, but Aho, Garey and Ullman [1] show that a dag has only one.

THEOREM 3.1. **[1]** *If G is a directed acyclic graph, then the transitive reduction of G is unique, and it is a subgraph of G.* □

Figure 2 shows the transitive reduction of the graph in Figure 1. Since there is a directed path $\langle 6, 3, 1\rangle$, the directed edge $\langle 6, 1\rangle$ is redundant. Similarly $\langle 5, 2\rangle$ is redundant because of the path $\langle 5, 4, 2\rangle$.

Since the transitive reduction preserves paths, we can restate Theorem 2.1 in terms of it.

COROLLARY 3.2. *The structure* Struct(x) *of the solution vector x to $Lx = b$ is given by the set of vertices reachable from vertices of the right-hand side structure* Struct(b) *by paths in the transitive reduction $G^\circ(L^T)$ of the directed graph $G(L^T)$.* □

A notion related to transitive reduction is the *transitive closure* G^* of a directed graph G, which is the graph that has an edge $\langle u, v\rangle$ whenever G has a directed path from u to v. The transitive closure is the least compact representation of the path information in G; the transitive reduction G° can also be defined as the smallest graph with the same transitive closure as G. Gilbert [7] shows that the transitive closure captures the structure of the inverse of a matrix:

THEOREM 3.3. **[7]** *Let A be a nonsingular matrix, not necessarily triangular, with nonzero diagonal elements. Ignoring coincidental cancellation, $G(A^{-1}) = G^*(A)$.* □

$$A = \begin{pmatrix} 1 & & \bullet & & & \\ & 2 & & & & \\ \bullet & & 3 & & & \bullet \\ & \bullet & \bullet & 4 & \bullet & \\ & \bullet & & \bullet & 5 & \\ \bullet & \bullet & & & & 6 \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ & 2 & & & & \\ \bullet & & 3 & & & \\ & \bullet & \bullet & 4 & & \\ & \bullet & & \bullet & 5 & \\ \bullet & \bullet & \circ & & & 6 \end{pmatrix} \begin{pmatrix} 1 & & \bullet & & & \\ & 2 & & & & \\ & & 3 & & & \bullet \\ & & & 4 & \bullet & \circ \\ & & & & 5 & \circ \\ & & & & & 6 \end{pmatrix}$$

FIG. 3. *LU factorization of a matrix example.*

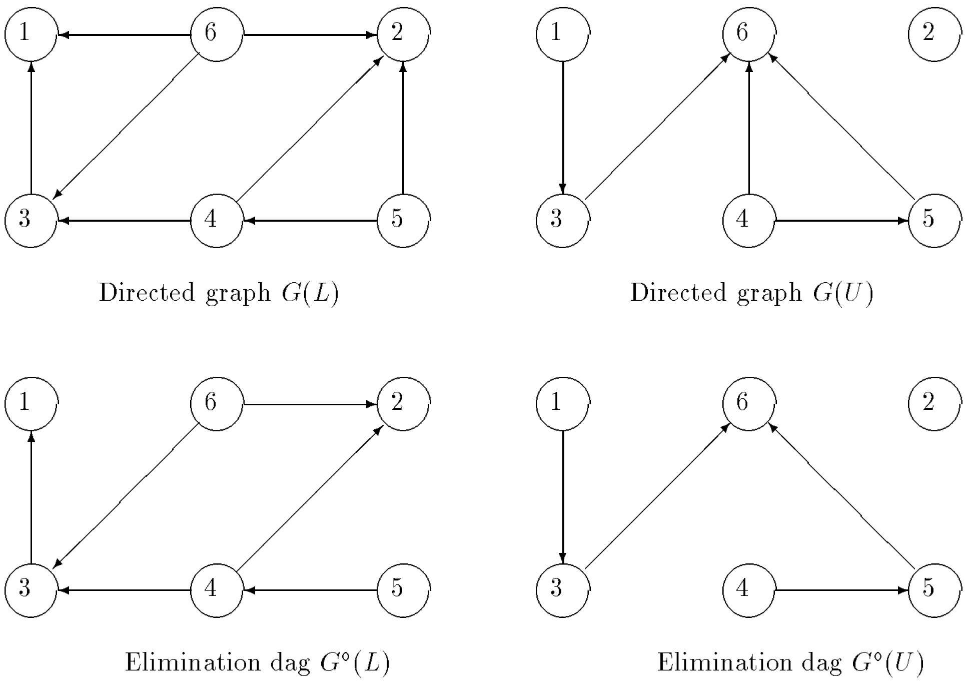


FIG. 4. *Elimination dags of the matrix in Figure 3.*

3.2. Definition of elimination dags. As before, let A be an unsymmetric $n \times n$ matrix that can be factored as $A = LU$ without pivoting. The lower triangular matrix L has a directed acyclic graph $G(L)$, which has a unique transitive reduction $G^\circ(L)$ by Theorem 3.1.

Similarly, the upper triangular matrix U has a directed acyclic graph $G(U)$, which in turn has a unique transitive reduction $G^\circ(U)$. We call the two reduced directed acyclic graphs $G^\circ(L)$ and $G^\circ(U)$ the *lower* and *upper elimination directed acyclic graphs* respectively. We shall also refer to them collectively as the *elimination dags* of the matrix A.

As an illustration, consider the unsymmetric matrix in Figure 3. An LU factorization of this matrix creates three fills, one in L and two in U. The fills are represented in the figure by "$\circ$". Figure 4 displays the two elimination dags corresponding to this matrix example.

3.3. The symmetric case: Elimination trees. The situation is simpler if the matrix A is symmetric. Still assuming no pivoting, Gaussian elimination yields a symmetric factorization $A = LL^T$. The *elimination tree* [10, 14] has been used extensively to study symmetric Gaussian elimination.

Let L be the Cholesky factor of a symmetric positive definite matrix A. Formally, the elimination tree $T(A)$ of A has n vertices $\{1, \ldots, n\}$, and $\langle j, k \rangle$ is an edge if and only if

$$k = \min\{r > j \mid \ell_{rj} \neq 0\}.$$

This structure is a tree rooted at vertex n if A is irreducible, or a forest with one tree for each irreducible block if A is reducible. Liu's survey paper [11] contains a comprehensive exposition of this important structure in the context of sparse factorization. We quote one of its properties in our terminology.

THEOREM 3.4. **[11]** *The elimination tree $T(A)$ is the transitive reduction of the directed graph $G(L^T)$.* □

Since the transitive reduction of a dag is unique, it follows that for a symmetric matrix the elimination dags $G^\circ(L^T)$ and $G^\circ(U)$ are both equal to the elimination tree $T(A)$. In Section 4, we shall show that the structural characterization of Cholesky factors by elimination trees can be generalized to unsymmetric factors using elimination dags.

3.4. Elimination dags and path structure. Elimination dags capture the path structure of the LU factors of a sparse matrix. In this section we explore some of the connections between edags, the path structures of L and U, and the path structure of the original matrix A. We also answer the question, "which pairs of directed graphs can be the elimination dags of some matrix?" Recall that we consider only matrices A with nonzero diagonal elements, and recall the discussion of coincidental cancellation in Section 2.3.

We begin by defining notation for two directed graphs that correspond to addition and multiplication of matrices.

Definition. If B and C are two $n \times n$ matrices with nonzero diagonal elements, then $G(B) + G(C)$ is the *union* of the graphs of B and C, defined as the graph whose vertex set is $\{1, \ldots, n\}$ and whose edge set is the union of those of $G(B)$ and $G(C)$.

Definition. If B and C are two $n \times n$ matrices with nonzero diagonal elements, then $G(B) \cdot G(C)$ is the *product* of the graphs of B and C, defined as the graph whose vertex set is $\{1, \ldots, n\}$, with an edge $\langle i, j \rangle$ exactly in case $\langle i, j \rangle$ is an edge of $G(B)$, or $\langle i, j \rangle$ is an edge of $G(C)$, or there is some k such that $\langle i, k \rangle$ is an edge of $G(B)$ and $\langle k, j \rangle$ is an edge of $G(C)$.

We will write $G \subseteq H$ to mean that G is a subgraph of H.

LEMMA 3.5. *If B and C are matrices with nonzero diagonals, then*

$$G(B + C) \subseteq G(B) + G(C),$$

with equality unless there is coincidental cancellation in $B + C$.

Proof. Immediate. □

LEMMA 3.6. *If B and C are matrices with nonzero diagonals, then*

$$G(BC) \subseteq G(B) \cdot G(C),$$

with equality unless there is coincidental cancellation in BC.

Proof. If $A = BC$, then $a_{ij} \neq 0$ only if there exists k such that $b_{ik} \neq 0$ and $c_{kj} \neq 0$. Conversely, if such k exists then a_{ij} is a sum of nonzeros and hence can be zero only if there is cancellation. □

If $A = LU$ is an LU factorization, Lemma 3.6 implies that $G(A)$ is a subgraph of $G(L) \cdot G(U)$, that is, that an edge in A corresponds to an edge in L followed by an edge in U. However, $G(A)$ is not usually equal to $G(L) \cdot G(U)$; any fill in L and U cancels out when they are multiplied back together.

Our first result relates paths in A to paths in L and U. It says that a path in A always corresponds to a path in U followed by a path in L. We allow paths of length zero.

THEOREM 3.7. *If $A = LU$ and there is a path in the directed graph $G(A)$ from vertex i to vertex j, then there is some $1 \leq k \leq n$ such that the directed graph $G(U)$ has a path from vertex i to vertex k, and the directed graph $G(L)$ has a path from vertex k to vertex j.*

Proof. This can be proved by induction using the "path lemma" of Rose and Tarjan [12], which characterizes $G(L+U)$ in terms of $G(A)$. Another proof, however, is to interpret the equation $A^{-1} = U^{-1}L^{-1}$ in terms of paths and transitive closures. Theorem 3.3 implies that

$$\begin{aligned} G^*(A) &= G(A^{-1}) \\ &= G(U^{-1}L^{-1}) \\ &\subseteq G(U^{-1}) \cdot G(L^{-1}) \\ &= G^*(U) \cdot G^*(L), \end{aligned}$$

where the third step is by Lemma 3.6. In English, this equation says that every path in A corresponds to a path in U followed by a path in L. □

Since the elimination dags of A preserve the path information in $G(L)$ and $G(U)$, the same conclusion holds for them.

COROLLARY 3.8. *If $A = LU$ and there is a path in the directed graph $G(A)$ from vertex i to vertex j, then there is some $1 \leq k \leq n$ such that the elimination dag $G^\circ(U)$ has a path from vertex i to vertex k, and the elimination dag $G^\circ(L)$ has a path from vertex k to vertex j. In other words,*

$$G^*(A) \subseteq G^{\circ *}(U) \cdot G^{\circ *}(L). \quad \square$$

This says that, while the elimination dags have very simple (i.e., acyclic) path structure themselves, they preserve all the path structure of A. Any path in $G(A)$ corresponds to an "ascending" path in $G^\circ(U)$ followed by a "descending" path in $G^\circ(L)$.

We turn now to the question of characterizing the graphs that can be elimination dags of some matrix. Again we consider $G(L)$ and $G(U)$ first. Any directed graph G with vertices $\{1, \ldots, n\}$ whose edges $\langle i, j\rangle$ all satisfy $i > j$ is $G(L)$ for some A. We merely choose A to be the matrix whose entries are $a_{ij} = 1$ whenever $i = j$ or $\langle i, j\rangle$ is an edge of G, and zeros elsewhere. Then $A = LU$ with $L = A$ and $U = I$, and $G(L) = G(A) = G$. Similarly any graph with edges directed from lower to higher numbered vertices is the graph of U for some A.

However, not every *pair* of directed acyclic graphs corresponds to a factorization $A = LU$ without cancellation. Rose and Tarjan [12] show that (in the absence of coincidental cancellation) $G(L+U)$ is always a so-called *perfect elimination digraph* in perfect elimination order, which means that whenever $k < \min(i, j)$ and $\langle i, k\rangle$ and $\langle k, j\rangle$ are edges, then $\langle i, j\rangle$ is also an edge. This condition is both necessary and sufficient. We can restate it in terms of our definitions as follows.

THEOREM 3.9. **[12]** *Suppose there is no cancellation in the factorization $A = LU$. Then*

$$G(L) \cdot G(U) = G(L) + G(U). \tag{1}$$

Furthermore, a specified lower and upper triangular structure $G(L)$ and $G(U)$ correspond to the LU factorization without cancellation of some matrix if and only if they satisfy Equation 1. □

The situation for structures of elimination dags is similar: Any transitively reduced dag can be the structure of the elimination dag $G^\circ(L)$ or $G^\circ(U)$, but a specified pair of reduced dags might not be $G^\circ(L)$ and $G^\circ(U)$ for any single matrix $A = LU$. We can characterize the possible pairs in a way similar to Theorem 3.9.

THEOREM 3.10. *Suppose there is no cancellation in the factorization $A = LU$. Then*

$$G^{\circ *}(L) \cdot G^{\circ *}(U) = G^{\circ *}(L) + G^{\circ *}(U). \tag{2}$$

Furthermore, a reduced structure is the pair of edags of some matrix if and only if it satisfies Equation 2.

REMARK. An equivalent way to state the theorem is: Let two transitively reduced acyclic directed graphs G_ℓ and G_u be given, with edges oriented from higher to lower numbered vertices in G_ℓ and vice versa in G_u. Then there exists a matrix A that factors without cancellation as LU with $G^\circ(L) = G_\ell$ and $G^\circ(U) = G_u$ if and only if $G_\ell^* \cdot G_u^* = G_\ell^* + G_u^*$.

Proof. First, let $A = LU$ be given. The containment $G^{\circ *}(L) + G^{\circ *}(U) \subseteq G^{\circ *}(L) \cdot G^{\circ *}(U)$ follows from the definition of the product graph. We prove the opposite containment. Note that $G^{\circ *}(L) = G^*(L)$ and $G^{\circ *}(U) = G^*(U)$.

Let $\langle i, j\rangle$ be an edge of $G^{\circ *}(L) \cdot G^{\circ *}(U)$. Then there exists k such that there are paths $\langle i = i_t, i_{t-1}, \ldots, i_0 = k\rangle$ in $G(L)$ and $\langle k = j_0, j_1, \ldots, j_s = j\rangle$ in $G(U)$. We induct on $s + t$, the total length of these paths. If either path has length zero, then the concatenation of the paths is in $G(L)$ or $G(U)$ and $\langle i, j\rangle$ is an edge of $G^*(L) + G^*(U) = G^{\circ *}(L) + G^{\circ *}(U)$. If both paths have length at least one and

$i_1 = j_1$, then we can shorten the paths by deleting $i_0 = j_0$. Otherwise, the edges $\langle i_1, k\rangle$ in $G(L)$ and $\langle k, j_1\rangle$ in $G(U)$ make $\langle i_1, j_1\rangle$ an edge of $G(L)\cdot G(U)$. Theorem 3.9 then implies that $\langle i_1, j_1\rangle$ is an edge of $G(L) + G(U)$; that is, it is either an edge of $G(L)$ or an edge of $G(U)$. If $i_1 > j_1$ then $\langle i_1, j_1\rangle$ is an edge of $G(L)$. Then the paths $\langle i = i_t, i_{t-1}, \ldots, i_1, j_1\rangle$ in $G(L)$ and $\langle j_1, j_2 \ldots, j_s = i\rangle$ in $G(U)$ have smaller total length than the original paths, so the inductive hypothesis applies. Similarly, if $i_1 < j_1$ then we shorten the path in $G(L)$.

Second, let G_ℓ and G_u be given. The "only if" part of the theorem is what we have just proved. To prove the "if" part, we assume that $G_\ell^* \cdot G_u^* = G_\ell^* + G_u^*$ and proceed to show the existence of a suitable A. Now G_ℓ^* and G_u^* satisfy Equation 1, so Theorem 3.9 says that there exists $A = LU$ (without cancellation) such that $G(L) = G_\ell^*$ and $G(U) = G_u^*$. But since G_ℓ and G_u are transitively reduced and the transitive reduction of a dag is unique, this implies $G^\circ(L) = G_\ell$ and $G^\circ(U) = G_u$ as well. □

All these results reduce to fairly simple facts about elimination trees in the symmetric case. In the symmetric version of Theorem 3.7, an irreducible A gives a complete graph $G^*(A)$ and a complete lower triangular structure $G^*(L)$. Theorem 3.9 says in the symmetric case that $G(L)\cdot G(L^T) = G(L)+G(L^T)$, which restates that $G(L)$ is a perfect elimination or *chordal* graph [13]. The symmetric case of Theorem 3.10 is an unusual characterization of a forest (that is, of a graph whose connected components are trees): $G^{\circ *}(L) \cdot G^{\circ *}(L^T) = G^{\circ *}(L) + G^{\circ *}(L^T)$ says that a "descending" path in $T(A)$ followed by an "ascending" path in $T(A)$ can be replaced by a pure descending or ascending path. This is indeed true of forests, and forests are the only transitively reduced directed acyclic graphs of which it is true.

The latter observation is a roundabout proof that the elimination tree of a symmetric matrix really is a tree (or, rather, a forest). One important fact about an elimination tree of an $n \times n$ matrix is that it encodes all the path information about $G(L)$ in a structure of linear size—with n vertices and at most $n - 1$ edges. Thus $G^\circ(L)$ is likely to be much smaller than L.

An elimination dag, on the other hand, may have $\Theta(n^2)$ edges in the worst case. Matrix EG-3 in Figure 10 is an example where both $G(U)$ and $G^\circ(U)$ have about $n^2/4$ edges. Nevertheless, in practice we observe that the elimination dags of a matrix are almost always much smaller than its triangular factors. This fact explains the results of Section 5.3, in which an algorithm based on edags is much faster than two earlier algorithms.

4. Structural characterization of LU factors.

In this section we use the elimination dags $G^\circ(L)$ and $G^\circ(U)$ to characterize the row and column structures of the triangular factors. Throughout the section, $A = LU$ is the factorization without pivoting of a square matrix A with nonzero diagonal. Recall the discussion in Section 2.3 of our assumptions about cancellation.

4.1. Row structure of L.

We first focus our attention on the row structure of the lower triangular factor L. The definitions immediately imply the following proposition.

PROPOSITION 4.1. *If $\ell_{ij} \neq 0$, then there exists a path from vertex i to vertex j in the elimination dag $G^\circ(L)$.* □

This condition is necessary but clearly not sufficient for an entry in L to be nonzero. For example, the entry ℓ_{41} is zero in Figure 3, but there is a path from vertex 4 to 1 via 3. We now provide a necessary and sufficient condition for ℓ_{ij} to be nonzero in terms of paths in the upper elimination dag $G^\circ(U)$.

THEOREM 4.2. *Let $i > j$. Then $\ell_{ij} \neq 0$ if and only if there is some $k \leq j$ such that $a_{ik} \neq 0$ and there is a directed path in the elimination dag $G^\circ(U)$ from vertex k to vertex j.*

Proof. Let A_i be the $i \times i$ leading principal submatrix of A, with factors $A_i = L_i U_i$. Then the i-th row of L (without the diagonal entry) is given by

$$(\ell_{i1}, \ldots, \ell_{i\,i-1})^T = \begin{pmatrix} \ell_{i1} \\ \vdots \\ \ell_{i\,i-1} \end{pmatrix} = U_{i-1}^{-T} \begin{pmatrix} a_{i1} \\ \vdots \\ a_{i\,i-1} \end{pmatrix} = U_{i-1}^{-T}(a_{i1}, \ldots, a_{i\,i-1})^T.$$

Therefore, by Theorem 2.1, the vector structure of $(\ell_{i1}, \ldots, \ell_{i,i-1})$ is given by the set of vertices reachable from the vertices in the vector structure of $(a_{i1}, \ldots, a_{i,i-1})$ in the directed graph $G(U_{i-1})$. Thus $\ell_{ij} \neq 0$ iff there exists a nonzero a_{ik} such that vertex j is reachable from vertex k in the graph $G(U_{i-1})$; in other words, iff there exists k such that $a_{ik} \neq 0$ and $G(U_{i-1})$ contains a directed path from k to j. Such a k must be less than or equal to j because all edges of $G(U)$ are directed from lower- to higher-numbered vertices.

Now $G(U_{i-1})$ is a subgraph of $G(U)$, so a path in $G(U_{i-1})$ is a path in $G(U)$ as well. Conversely (again because edges are directed from lower- to higher-numbered vertices) any directed path ending at vertex j in $G(U)$ must lie entirely within $G(U_{i-1})$. Thus $\ell_{ij} \neq 0$ iff there exists k such that $a_{ik} \neq 0$ and there is a directed path from k to j in $G(U)$. The result now follows from the fact that the elimination dag $G^\circ(U)$ preserves the set of paths in $G(U)$. $\square$

Theorem 4.2 characterizes the structure of L by rows: the structure of the i-th row of L is given by the subset of $\{1, \ldots, i\}$ reachable in the upper elimination dag $G^\circ(U)$ from the vertices of the structure of the i-th row in the lower triangular part of A. The following section characterizes the structure of L by columns.

4.2. Column structure of L. Using the notation $\mathrm{Struct}(L_{*j}) = \{i \mid \ell_{ij} \neq 0\}$ for the nonzero structure of a column of L, we restate a result of Rose and Tarjan [12] that describes the structure of L by columns in terms of A and U.

THEOREM 4.3. **[12]**

$$\begin{aligned} &\mathrm{Struct}(L_{*j}) = \\ &\quad \mathrm{Struct}(A_{*j}) \cup \bigcup \{\mathrm{Struct}(L_{*k}) \mid k < j, u_{kj} \neq 0\} - \{1, \ldots, j-1\}. \quad \square \end{aligned}$$

COROLLARY 4.4. *Let $k < j$ and $u_{kj} \neq 0$. Then*

$$\mathrm{Struct}(L_{*k}) - \{1, \ldots, j-1\} \subseteq \mathrm{Struct}(L_{*j}). \quad \square$$

In matrix terms, we can determine the column structure of L_{*j} by taking the union of the column structure of A_{*j} with a number of columns of L before j. These columns are given exactly by the structure of column j of U, the upper triangular factor. Figure 5 expresses the result of Theorem 4.3 diagrammatically.

Note that all columns L_{*k} such that $u_{kj} \neq 0$ are used to determine the structure of L_{*j} in Theorem 4.3. The following theorem shows that we need only consider a subset of those columns, and this subset is determined by the structure of the upper elimination dag $G^\circ(U)$.

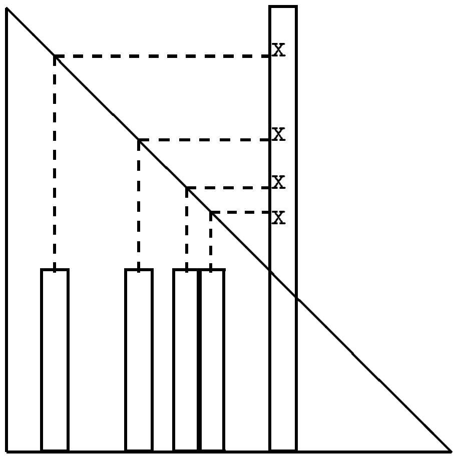


FIG. 5. *Union of column structures.*

THEOREM 4.5.

$$\begin{aligned}&\mathrm{Struct}(L_{*j}) = \\ &\quad \mathrm{Struct}(A_{*j}) \ \cup \ \bigcup \{\mathrm{Struct}(L_{*k}) \mid \langle k, j\rangle \text{ is an edge of } G^{\circ}(U)\} - \{1, \ldots, j-1\}.\end{aligned}$$

Proof. The right-hand side in Theorem 4.5 is a subset of the right-hand side of Theorem 4.3 by the definition of $G^{\circ}(U)$.

To prove the converse, consider any nonzero entry u_{kj} where $\langle k, j\rangle$ is not an edge of $G^{\circ}(U)$. Then $G^{\circ}(U)$ must contain a path $\langle k = i_0, i_1, \ldots, i_t, i_{t+1} = j\rangle$, with $u_{i_s i_{s+1}} \neq 0$ for $0 \leq s \leq t$. Corollary 4.4 says that

$$\begin{aligned}\mathrm{Struct}(L_{*k}) - \{1, \ldots, i_1 - 1\} &\subseteq \mathrm{Struct}(L_{*i_1}),\\ \mathrm{Struct}(L_{*i_1}) - \{1, \ldots, i_2 - 1\} &\subseteq \mathrm{Struct}(L_{*i_2}),\\ &\vdots\\ \mathrm{Struct}(L_{*i_{t-1}}) - \{1, \ldots, i_t - 1\} &\subseteq \mathrm{Struct}(L_{*i_t}).\end{aligned}$$

Combining, we have

$$\mathrm{Struct}(L_{*k}) - \{1, \ldots, j-1\} \subseteq \mathrm{Struct}(L_{*i_t}) - \{1, \ldots, j-1\},$$

where $\langle i_t, j\rangle$ is an edge of $G^{\circ}(U)$. We have thus proved that the right hand side in Theorem 4.3 is a subset of that in Theorem 4.5. $\square$

4.3. Row and column structures of U. By the same argument as in Theorems 4.2 and 4.5, we can relate the nonzero structure of the upper triangular factor U to the lower elimination dag $G^{\circ}(L)$. Theorem 4.6 characterizes the column structure and Theorem 4.7 the row structure of U.

THEOREM 4.6. *Let $i < j$. Then $u_{ij} \neq 0$ if and only if there is some $k \leq i$ such that $a_{kj} \neq 0$ and there is a directed path in the elimination dag $G^{\circ}(L)$ from vertex i to vertex k.* $\square$

THEOREM 4.7.

$$\text{Struct}(U_{i*}) =$$
$$\text{Struct}(A_{i*}) \cup \bigcup \{\text{Struct}(U_{k*}) \mid \langle i, k\rangle \text{ is an edge of } G^\circ(L)\} - \{1, \ldots, i-1\}. \quad \square$$

We now use the matrix example in Figure 3 to illustrate the results of Theorems 4.2 and 4.6 on the row structure of L and the column structure of U. Consider row and column 6 of the matrix in Figure 3. The structure of row A_{6*} is $\{1, 2, 6\}$. The set of vertices reachable from this set in the upper elimination dag $G^\circ(U)$ in Figure 4 is $\{1, 2, 3, 6\}$, which is precisely the structure of L_{6*}. On the other hand, the structure of column A_{*6} is $\{3, 6\}$. These two vertices in the lower elimination dag $G^\circ(L)$ are reachable exactly from vertices in the set $\{3, 4, 5, 6\}$. As Theorem 4.6 predicts, this is the structure of U_{*6}.

Using the same matrix, we illustrate the results of Theorems 4.5 and 4.7 on the column structure of L and the row structure of U. Consider row and column 5 of the matrix in Figure 3. Since $\langle 4, 5\rangle$ is the only edge into vertex 5 in the upper elimination dag $G^\circ(U)$, Theorem 4.5 says that Struct(L_{*5}) can be obtained from Struct(A_{*5}) and Struct(L_{*4}). On the other hand, of the two edges $\langle 5, 2\rangle$ and $\langle 5, 4\rangle$ in the graph $G(L)$, only $\langle 5, 4\rangle$ is in the lower elimination dag $G^\circ(L)$. Therefore, by Theorem 4.7, it is sufficient to consider Struct(A_{5*}) and Struct(U_{4*}) to determine Struct(U_{5*}).

4.4. Edags as unsymmetric analogs of elimination trees. In Section 3.3 we saw that when the matrix A is symmetric, the elimination dag $G^\circ(L^T) = G^\circ(U)$ is identical to the elimination tree $T(A)$. Here we show that the results from the literature relating elimination trees to the structure of sparse Cholesky factors are special cases of the results of Sections 4.1 through 4.3. Thus elimination dags are truly an unsymmetric analog of elimination trees.

In this subsection, we take A to be symmetric and positive definite, and L to be its Cholesky factor. The following result characterizes the column structure of L using the elimination tree, and forms the basis of an efficient symbolic Cholesky factorization scheme.

THEOREM 4.8. **[11]**

$$\text{Struct}(L_{*j}) =$$
$$\text{Struct}(A_{*j}) \cup \bigcup \{\text{Struct}(L_{*k}) \mid k \text{ is a child of } j \text{ in } T(A)\} - \{1, \ldots, j-1\}. \quad \square$$

Since $G^\circ(L^T) = T(A)$ in the symmetric case, this is a special case of Theorem 4.5 or Theorem 4.7.

The row structure of the Cholesky factor L can also be characterized in terms of the elimination tree. Schreiber [14] shows that the row structure of L_{i*} is a pruned subtree rooted at vertex i of the elimination tree. Liu [10] gives a complete characterization, which we quote here.

THEOREM 4.9. **[10]** *Let $i > j$. Then $\ell_{ij} \neq 0$ if and only if vertex j is an ancestor of some vertex k in the elimination tree such that $a_{ik} \neq 0$.* $\square$

We can restate this in terms of paths in the elimination tree as follows.

COROLLARY 4.10. *Let $i > j$. Then $\ell_{ij} \neq 0$ if and only if there is some $k \leq j$ such that $a_{ik} \neq 0$ and there is a directed path in the elimination tree $T(A)$ from vertex k to vertex j.* $\square$

This is a special case of Theorem 4.2 or Theorem 4.6.

5. An application: Fill computation. In this section we apply elimination dags to computing the symbolic factorization of an unsymmetric matrix. For this problem, we are given the nonzero structure of an unsymmetric matrix A and asked to compute the nonzero structures of its LU factors, under the assumption that A has an LU factorization without pivoting. Equivalently, we want to compute the result of playing the vertex elimination game [12] on a specified directed graph. We first review two algorithms from the literature briefly. Then we present a new algorithm, analyze it, and give experimental results.

In this section we use the notation $\eta(X)$ for the number of nonzeros in the matrix or vector X.

5.1. Algorithms FILL1 **and** FILL2**.** Rose and Tarjan [12] present two algorithms, which they call FILL1 and FILL2, to determine fill. Both algorithms compute the structures of L and U row by row.

Algorithm FILL1 determines the row structures of L_{i*} and U_{i*} by using the structure of A_{i*} together with the first $i-1$ computed row structures of U. It basically performs a symbolic simulation of the numerical row elimination scheme. The amount of work required is proportional to that required for numerical factorization. This is also proportional to the time required to multiply L and U back together again, which we write informally as flops(LU).

On the other hand, algorithm FILL2 computes the row structures by using the "path lemma" for fills [12, Theorem 1], which says that $\langle i, j\rangle$ is an edge of $G(L+U)$ if and only if the graph $G(A)$ contains a directed path from i to j through vertices with numbers smaller than both i and j. The structures of L_{i*} and U_{i*} are determined using only the structures of the first i rows of the original matrix A. Rose and Tarjan show that the time complexity of FILL2 is bounded by $O(n\eta(A))$.

The reader is referred to the paper by Rose and Tarjan [12] for detailed descriptions of these algorithms. Rose and Tarjan do not provide code; our implementations, which we used to obtain the experimental results reported in this paper, are based on their descriptions.

5.2. A symbolic LU algorithm using elimination dags. The structural characterization results of Theorems 4.2, 4.5, 4.6, and 4.7 for sparse LU factors can be used to formulate a symbolic factorization algorithm. The algorithm in Figure 6 computes the structures of L and U row by row. We use the notation L_i and U_i to denote the $i \times i$ leading principal submatrix of L and U. Their graphs and transitive reductions are defined in the same way as for L and U.

The algorithm uses Theorem 4.2 to determine the structure of L_{i*}, and Theorem 4.7 to determine the structure of U_{i*}. At the end of the i-th iteration, the structures of the first i rows of L and U are known together with the structures of the elimination dags $G^\circ(L_i)$ and $G^\circ(U_i)$. The data structure represents L and A by columns, and U by rows; that is, it stores the adjacency lists of the various graphs $G(L^T)$, $G(U)$, $G^\circ(L^T)$, and $G^\circ(U)$. (The adjacency lists are not actually linked lists, but arrays of row numbers as in the standard column-oriented sparse matrix data structure [6].)

We now analyze the time complexity of this algorithm. To get simple expressions for the complexity, we will use the fact [9] that if there is no cancellation in solving $Lx = b$, then the number of nonzero arithmetic operations is the same as the number of nonzero arithmetic operations needed to multiply L by x, and is on the order of the time needed to traverse the part of the graph $G(L^T)$ reachable from b.

algorithm EDAGS
for row $i := 1$ **to** n **do**
1: Compute Struct(L_{i*}) by traversing $G^\circ(U_{i-1})$
from A_{i*} (using Theorem 4.2);
2: Transitively reduce Struct(L_{i*}) using $G^\circ(L_{i-1})$,
thus extending the edag to $G^\circ(L_i)$;
3: Compute Struct(U_{i*}) as a union of earlier rows of U
(using Theorem 4.7);
4: Transitively reduce Struct(U_{*i}) using $G^\circ(U_{i-1})$,
thus extending the edag to $G^\circ(U_i)$;
end for

FIG. 6. *Symbolic LU factorization using elimination dags.*

In Step 1, at row i, this fact says that the traversal of $G^\circ(U_{i-1})$ takes time on the order of the number of operations to multiply a (hypothetical) matrix whose structure is that of $G^\circ(U_{i-1}^T)$ by the vector L_{i*}^T. Added up over all rows, this is bounded by the number of operations to multiply a matrix whose structure is that of $G^\circ(U^T)$ by L^T; we will write this number informally as flops($U^{\circ T} L^T$) or flops(LU°). Similarly, the time for Step 3 is bounded by flops($L^\circ U$).

Step 2 can be implemented in at least two ways. One method is to search in $G^\circ(L_{i-1})$ from the vertices corresponding to nonzeros in L_{i*}, reducing L_{i*} by discarding element ℓ_{ij} if vertex j is reached during the search from some ℓ_{ik} with $j < k$. The search at row i can avoid looking at any edge twice by searching in decreasing order of k and marking vertices that have already been reached. The time for this reduction search is on the order of the operations to solve for y in $L_{i-1}^{\circ T} y = L_{i*}^T$. If we add vertex i to get the graph $G^\circ(L_i)$, we see that the reduction search is the same as the search that would be done in solving $L_i^{\circ T} y = e_i$ (where e_i is the i-th unit vector), except that from vertex i the reduction search examines all the nonzeros of L_{i*}^T instead of just those of $L_{i*}^{\circ T}$. Thus for row i, the search takes time on the order of operations to multiply $L_i^{\circ T}$ by column i of its inverse, plus $\eta(L_{i*}^T) - \eta(L_{i*}^{\circ T})$. Adding this up over all i, and noting that since $G(L)$ and $G^\circ(L)$ have the same transitive closures their inverses have the same structure, we get flops($L^{-1}L^\circ$) $+ \eta(L) - \eta(L^\circ)$. The last two terms are dominated by the first, so the total time for Step 2 is on the order of flops($L^{-1}L^\circ$). Similarly the total time for Step 4 is on the order of flops($U^\circ U^{-1}$) by this method.

The second method for implementing Step 2 is to search "backwards" (that is, in $G^\circ(L_{i-1}^T)$), discarding nonzero ℓ_{ij} if vertex i can be reached from some ℓ_{ik} with $k < j$. Again the search can avoid looking at an edge twice by marking vertices that have been reached. The search at row i by this method is on the order of operations to solve $L_{i-1}^\circ y = L_{i*}^T$. It seems harder to get a simple expression for the total cost over all rows. To get an upper bound, we note that the search for row i at worst examines all of $G^\circ(L_{i-1})$, and thus the total time is no more than the order of $\sum_i \eta(L_i^\circ)$. The total time for Step 4 is bounded above by a similar sum for U.

One way to express this upper bound is as follows. Let F be a full n by n "skew upper triangular" matrix, that is $f_{ij} \neq 0$ if $i+j \leq n+1$. Then $\sum_i \eta(L_i^\circ)$ is on the order of flops(FL°), and $\sum_i \eta(U_i^\circ)$ is on the order of flops($U^\circ F$). Thus the total time for

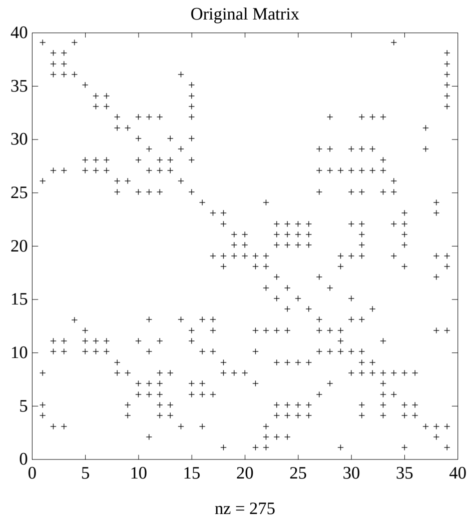


FIG. 7. *The nonzero structure of matrix A.*

the algorithm (using the second method for Steps 2 and 4) is bounded by the order of flops($FL^\circ + U^\circ F + LU^\circ + L^\circ U$), which in turn is bounded above by flops($F(E+E^T)$), where $E = L^\circ + U^\circ$ is a matrix with the structure of the two edags.

This upper bound actually applies to both methods for Steps 2 and 4. In our implementation, we used the second method, since it uses $G^\circ(L^T)$ and thus fits more naturally with the column-oriented data structure for L.

Theoretically, none of the three algorithms FILL1, FILL2, or EDAGS dominates any of the others; any of them could be the fastest on some matrices. The next section describes three matrices, each of which takes $O(n^3)$ time by one algorithm and $O(n^2)$ time by the other two.

5.3. Experimental results. We compared the performance of the the symbolic LU algorithm using elimination dags with the FILL1 and FILL2 algorithms of Rose and Tarjan. The programs were written in Fortran, using many of the routines from SPARSPAK [6]. We used sparse MATLAB [8] for the reorderings and to plot results. Figures 7, 8, and 9 show silhouette plots of the structure of a small sample matrix,

TABLE 1
List of test problems.

Problem	n	Nonzeros	Description
SHL400	663	1712	Permuted triangular LP basis
FS7603	760	5976	Unsymmetric facsimile convergence matrix
MCFE	765	24382	Astrophysics problem
BP1600	822	4841	Unsymmetric LP basis
YOUNG3C	841	3988	Matrix from Young
DWT918	918	7384	Beam with cutouts (symmetric)
GRE1107	1107	5664	Problem from Grenoble collection
WEST2021	2021	7353	Chemical engineering: Column section

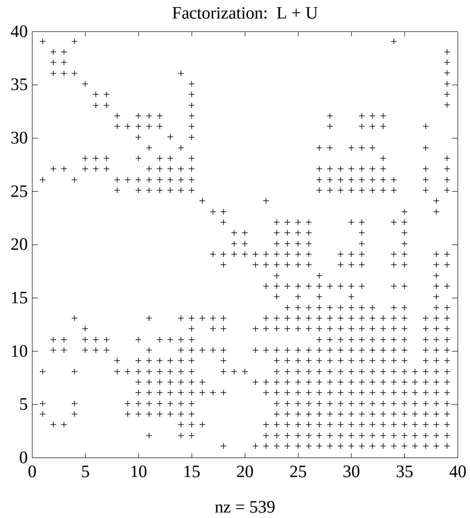


FIG. 8. *The triangular factors of* $A = LU$.

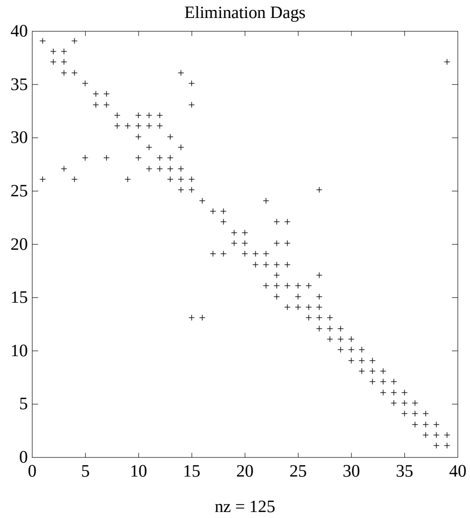


FIG. 9. *The elimination dags of* A, $G^{\circ}(L) + G^{\circ}(U)$.

$$
\begin{pmatrix}
1 & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & 2 & & & & & & \\
\bullet & & 3 & & & & & \\
\bullet & & & 4 & & & & \\
\bullet & & & & 5 & & & \\
\bullet & & & & & 6 & & \\
\bullet & & & & & & 7 & \\
\bullet & & & & & & & 8
\end{pmatrix}
\quad
\begin{pmatrix}
1 & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & 2 & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
 & \bullet & 3 & \bullet & \bullet & \bullet & \bullet & \bullet \\
 & & \bullet & 4 & \bullet & \bullet & \bullet & \bullet \\
 & & & \bullet & 5 & \bullet & \bullet & \bullet \\
 & & & & \bullet & 6 & \bullet & \bullet \\
 & & & & & \bullet & 7 & \bullet \\
 & & & & & & \bullet & 8
\end{pmatrix}
\quad
\begin{pmatrix}
1 & & & & \bullet & \bullet & \bullet & \bullet \\
\bullet & 2 & & & & & & \\
\bullet & & 3 & & & & & \\
\bullet & & & 4 & & & & \\
 & & & & 5 & & & \\
 & & & & & 6 & & \\
 & & & & & & 7 & \\
 & & & & & & & 8
\end{pmatrix}
$$

EG-1 (Arrow) | EG-2 (Hessenberg) | EG-3 (Worst Case)

FIG. 10. *Three contrived matrix examples.*

its factors, and its elimination dags.

All but three of our test problems are from the Harwell-Boeing Sparse Matrix Collection [3]; Table 1 describes these problems. These problems all have unsymmetric nonzero structures, except for DWT0918, which is a symmetric problem that we included to demonstrate that EDAGS works well in that case too. We also included examples contrived to make each of the three algorithms perform poorly. Figure 10 shows 8×8 versions of these artificial problems; in the experiments we used 500×500 versions.

Since the algorithms are all non-numerical and we are addressing only the case when no pivoting for numerical stability is necessary, the question of numerical error does not arise in the comparisons between the algorithms. We did, however, compute a numerical LU factorization for each matrix (after modifying the diagonal to make it dominant) in order to check that the combinatorial algorithms were producing the correct result.

Table 2 presents timing statistics for the three fill computation algorithms on the set of test problems. The experiments were performed on a Sun Sparcstation IPC workstation; times are reported in seconds as returned by calls to the system timer. For the Harwell-Boeing problems, FILL2 is consistently better than FILL1, whereas EDAGS outperforms the other two.

It is interesting to compare the performance on the three contrived examples. The analysis in the previous section can be used to show that algorithm FILL1 takes $\Theta(n^3)$ time on EG-1 and $\Theta(n^2)$ time on EG-2 and EG-3; algorithm FILL2 takes $\Theta(n^3)$ time on EG-2 and $\Theta(n^2)$ time on EG-1 and EG-3; and algorithm EDAGS takes $\Theta(n^3)$ time on EG-3 and $\Theta(n^2)$ time on EG-1 and EG-2. This analysis is reflected in the measured times.

Table 3 compares the sizes of the factor matrices and their elimination dags, both determined by EDAGS. (In the table, $|G|$ means the number of edges in the graph G.) We see that in the worst-case example EG-3 there is no difference, but in all the practical problems the edags are substantially smaller than the corresponding triangular factors. We also note that the symmetric matrix DWT0918, which is of order 918, has edags with 917 edges apiece; these are in fact both equal to its elimination tree.

Finally, for curiosity's sake, we present the result of one more experiment on the

Table 2
Time to compute $G(L+U)$ from A.

Problem	n	Nonzeros	Fill1	Fill2	Edags
SHL400	663	1712	0.65	0.65	0.48
FS7603	760	5976	122.50	5.63	5.51
MCFE	765	24382	3.41	12.81	1.15
BP1600	822	4841	14.47	2.85	2.05
YOUNG3C	841	3988	1.05	4.49	0.60
DWT918	918	7384	25.59	7.69	2.84
GRE1107	1107	5664	14.89	6.93	4.34
WEST2021	2021	7353	39.80	14.93	7.66
EG-1	500	1498	68.69	1.64	3.12
EG-2	500	125749	0.60	68.78	2.12
EG-3	500	999	0.19	0.34	21.85

Table 3
Structural statistics for algorithm Edags *in Figure 6.*

Problem	n	Nonzeros	$\|G(L)\|$	$\|G^\circ(L)\|$	$\|G(U)\|$	$\|G^\circ(U)\|$
SHL400	663	1712	9191	892	8181	1355
FS7603	760	5976	200187	778	205817	779
MCFE	765	24382	23779	761	55911	761
BP1600	822	4841	66688	1459	68078	1431
YOUNG3C	841	3988	22066	845	22188	846
DWT918	918	7384	107347	917	107347	917
GRE1107	1107	5664	61087	3135	141887	1442
WEST2021	2021	7353	148368	10730	231052	4430
EG-1	500	1498	124750	499	124750	499
EG-1	500	125749	499	499	124750	499
EG-1	500	999	249	249	62500	62500

Table 4
Timings for largest irreducible blocks, ordered by minimum degree.

Problem	n	Nonzeros	Fill1	Fill2	Edags
FS7603 B	740	5878	0.80	1.43	0.40
MCFE B	697	23186	6.20	6.77	1.12
BP1600 B	217	1155	0.02	0.10	0.04
YOUNG3C B	841	3988	0.23	1.08	0.23
DWT918 B	918	7384	0.71	1.87	0.43
GRE1107 B	1107	5664	5.88	3.45	1.25
WEST2021 B	1500	5495	0.08	1.89	0.18

Harwell-Boeing matrices. Many of these matrices are highly reducible. Reducible linear systems are often solved by permuting to block triangular form and then factoring only the irreducible diagonal blocks. For each matrix in the test set, we first applied a row permutation to place nonzeros on the diagonal, then permuted the matrix to block upper triangular form, and then extracted the largest diagonal block B. Then we reordered that block by a symmetric minimum degree permutation (on the structure of $B + B^T$). Table 4 presents the timings for the three fill algorithms on the resulting matrices. (There is no entry for SHL400, because that matrix is actually a permutation of a triangular matrix, so its largest irreducible diagonal block is 1×1.) We see that EDAGS is often fastest by a large margin, but FILL1 wins in some cases. The reason may be that these matrices, when reordered, have factors that are already so sparse that there is little to gain from transitive reduction.

6. Concluding remarks. This paper has defined a generalization of the elimination tree, which is an important structure in sparse Cholesky factorization, to an "elimination dag" useful for sparse unsymmetric LU factorization. The elimination dag arises by generalizing one particular property of the elimination tree, namely that it is the unique transitive reduction of the graph of the triangular factor. We have shown that elimination dags model the path structure of LU factors in a way that parallels, and generalizes, the elimination tree model of the Cholesky factor. We presented a new algorithm for sparse symbolic LU factorization, using edags, and showed experimentally that it compares favorably with existing symbolic schemes.

Eisenstat and Liu [5] have recently extended this work by defining a pair of dags they call the *symmetric reductions* of a matrix. These dags are intermediate in size between the edags and the triangular factors, but are easier to find than edags; Eisenstat and Liu's experiments show that a symbolic factorization algorithm based on them is even faster than our Algorithm EDAGS.

In this paper, we have used elimination dags only to study sparse LU factorization with no pivoting. When numerical pivoting is required for stability, we believe that edags can still be useful. The sparse partial pivoting scheme of Gilbert and Peierls [9] performs an LU factorization with pivoting in time proportional to arithmetic operations. Each column of the factors L and U is computed by a sparse triangular solve with the already-computed columns of L, of the form

$$\begin{pmatrix} L_j & 0 \\ L_j' & I \end{pmatrix} \begin{pmatrix} u_j \\ b_j \end{pmatrix} = a_j .$$

The triangular solve begins with a symbolic phase that uses the graph of the known part of L (actually, of L^T) to predict the nonzero structure of the result, in order to limit the numerical factorization to nonzero arithmetic. The symbolic phase for each column could be improved by using the elimination dag $G^\circ(L^T)$ instead of the graph $G(L^T)$. We expect this to be especially important when using vector machines, since the numerical phase can be speeded up by vectorization but the symbolic phase cannot. Eisenstat and Liu [4] have recently reported positive results using symmetric reductions in the symbolic phase of sparse partial pivoting on workstations.

REFERENCES

[1] A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1:131–137, 1972.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[3] I. S. Duff, R. Grimes, and J. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15:1–14, 1989.

[4] Stanley C. Eisenstat and Joseph W. H. Liu. Exploiting structural symmetry in a sparse partial pivoting code. Presented at the Fourth SIAM Conference on Applied Linear Algebra, Minneapolis, 1991.

[5] Stanley C. Eisenstat and Joseph W. H. Liu. Exploiting structural symmetry in unsymmetric sparse symbolic factorization. Technical Report CS–90–12, York University, 1990.

[6] Alan George and Joseph W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[7] John R. Gilbert. Predicting structure in sparse matrix computations. Technical Report 86–750, Cornell University, 1986. To appear in *SIAM Journal on Matrix Analysis and Applications*.

[8] John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in Matlab: Design and implementation. Technical Report CSL 91–4, Xerox Palo Alto Research Center, 1991. To appear in the *SIAM Journal on Matrix Analysis and Applications*.

[9] John R. Gilbert and Tim Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM Journal on Scientific and Statistical Computing*, 9:862–874, 1988.

[10] J. W. H. Liu. A compact row storage scheme for Cholesky factors using elimination trees. *ACM Transactions on Mathematical Software*, 12:127–148, 1986.

[11] J. W. H. Liu. The role of elimination trees in sparse factorization. *SIAM Journal on Matrix Analysis and Applications*, 11:134–172, 1990.

[12] D. J. Rose and R. E. Tarjan. Algorithmic aspects of vertex elimination of directed graphs. *SIAM Journal on Applied Mathematics*, 34:176–197, 1978.

[13] Donald J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32:597–609, 1970.

[14] R. Schreiber. A new implementation of sparse Gaussian elimination. *ACM Transactions on Mathematical Software*, 8:256–276, 1982.